

UCLA

UCLA Electronic Theses and Dissertations

Title

Multi-scale Human Behavior Modeling with Heterogeneous Data

Permalink

<https://escholarship.org/uc/item/1xh2s6ns>

Author

Jiang, Jyun-Yu

Publication Date

2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Multi-scale Human Behavior Modeling with Heterogeneous Data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Jyun-Yu Jiang

2021

© Copyright by
Jyun-Yu Jiang
2021

ABSTRACT OF THE DISSERTATION

Multi-scale Human Behavior Modeling with Heterogeneous Data

by

Jyun-Yu Jiang

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2021

Professor Wei Wang, Chair

In this era of big data, massive data are generated from heterogeneous resources every day, which provides an unprecedented opportunity for deepening our understanding of complex human behaviors. Modeling human behaviors requires robust computational methods that can not only capture semantics and useful insights from sparse and heterogeneous data, but also unravel sophisticated human behaviors at different scales. In addition, the enormous data velocity and the unparalleled scale of deep models also pose significant challenges to efficiency.

In this dissertation, we demonstrate a collection of research results that systematically improve the ecosystem of human behavior modeling based on representation learning. For heterogeneous data in various settings, we present practical representation learning methods to effectively and efficiently capture their semantics. Moreover, these representation learning methods can actually fill a niche to comfortably model different behaviors with atomic, compositional, and explainable operations, thereby modeling human behaviors at different scales. As a result, our proposed approaches not only address various real-world challenges in diverse domains, but also present the potentials to adopt valuable domain knowledge into machine learning.

The dissertation of Jyun-Yu Jiang is approved.

Yizhou Sun

Cho-Jui Hsieh

Kai-Wei Chang

Wei Wang, Committee Chair

University of California, Los Angeles

2021

To my dear family.

TABLE OF CONTENTS

List of Figures	xv
List of Tables	xxii
Acknowledgments	xxvii
Vita	xxx
1 Introduction	1
1.1 Heterogeneous Data Harnessing with Structural Semantics	2
1.2 Interdisciplinary Knowledge Discovery	5
1.3 Compositional and Explainable Behavior Modeling	6
1.4 Thesis Contributions	7
1.5 Thesis Overview	9
2 Related Work	10
2.1 Neural Language Models for Information Retrieval	10
2.2 Circular RNA Prediction	11
2.3 Network Representation Learning	13
2.3.1 Homogeneous Network Embedding Models	13
2.3.2 Heterogeneous Network Embedding Models	14
2.4 Collaborative Filtering for Recommender Systems	15
2.5 Maximum Inner Product Search	16
2.6 k-mer Signiture Profiling	17
2.7 Modeling Users behind Shared Accounts	18

2.8	Sponsorship Disclosure in Influencer Marketing	19
2.9	Query Suggestion and Reformulation	20
2.10	Conversation Disentanglement	21
3	Long Document Modeling for Information Retrieval	22
3.1	Introduction	22
3.2	Preliminaries on Document Ranking	24
3.3	QDS-Transformer	26
3.3.1	Query-Directed Sparse Attention	26
3.3.2	Summary	28
3.3.3	Efficient Sparsity Implementation	29
3.4	Experimental Methodologies	29
3.5	Evaluation Results	31
3.5.1	Retrieval Effectiveness	31
3.5.2	Effectiveness of Attention Patterns	33
3.5.3	Model Efficiency	35
3.5.4	Learned Attention Weights	36
3.5.5	Case Study on Learned Attention Weights	37
3.6	Conclusion	39
4	Sequence Modeling for Circular RNA Prediction	40
4.1	Introduction	40
4.2	Materials and Methods	43
4.2.1	Preliminary and Problem Statement	43
4.2.2	Framework Overview	44

4.2.3	Attentive Junction Encoders	45
4.2.4	Cross-attention for Modeling Deep Interaction	48
4.2.5	Circular RNA Prediction	49
4.2.6	Learning and Optimization	50
4.2.7	Remarks on the Interpretability of JEDI	50
4.3	Experiments	51
4.3.1	Datasets	51
4.3.2	Experimental Settings	52
4.3.3	Isoform-level Circular RNA Prediction	54
4.3.4	Gene-level Circular RNA Prediction	55
4.3.5	Independent Study on Mouse circRNAs	56
4.3.6	Zero-shot Backsplicing Discovery	57
4.3.7	Analysis and Discussions	58
4.4	Conclusions	61
5	Heterogeneous Network Representation Learning with Meta-context Aware	
	Random Walk	62
5.1	Introduction	62
5.2	Problem Statement	66
5.2.1	Heterogeneous Network	66
5.2.2	Problem Definition	66
5.3	<i>MARU</i> for Heterogeneous Network Embedding	67
5.3.1	Framework Overview	68
5.3.2	Bidirectional Extended Random Walks	69
5.3.3	Meta-context Aware Skip-gram Model	70

5.3.4	Complexity Analysis	73
5.4	Experiments	74
5.4.1	Datasets and General Experimental Settings	74
5.4.2	Task 1: Multi-label Node Classification	76
5.4.3	Tasks 2: Node Clustering	78
5.4.4	Task 3: Link Prediction	79
5.4.5	Analysis and Discussions	81
5.5	Conclusion	83
6	Learning User Coresets to Accelerate Large-scale Top-K Recommender Systems	85
6.1	Introduction	85
6.2	Problem Statement	88
6.3	Constructing User Coresets for Top-K Recommender Systems	89
6.3.1	Preliminary	89
6.3.2	Framework Overview	90
6.3.3	Preparation Stage	91
6.3.4	Prediction Stage	98
6.4	Experiments	98
6.4.1	Experimental Settings	98
6.4.2	Performance Comparison	101
6.4.3	Number of Sub-Sampled User Latent Vectors	103
6.4.4	Trade-off in Proximity Graph Construction	103
6.4.5	Number of Affinity Groups	104
6.4.6	Effectiveness of Adaptive Representative Selection	105

6.5	Conclusions	106
7	Efficient Signature Profiling in Genomic Data	107
7.1	Introduction	107
7.2	Materials and Methods	110
7.2.1	Problem Statement	111
7.2.2	Aho-Corasick Automaton	112
7.2.3	Thinned Automaton with Binarized Pattern Matching	113
7.2.4	Acceleration by Rolling Hash	116
7.2.5	Implementation Details	117
7.2.6	Software Adaptation	118
7.2.7	Synthetic <i>K</i> -mers Generation	118
7.2.8	Synthetic Reads	119
7.2.9	Real Datasets	119
7.3	Results and Discussion	119
7.3.1	Automaton Construction	119
7.3.2	Pilot Study of 13 Approaches	120
7.3.3	Extensive Study on Synthetic Datasets with Both Single and Multiple Threads	122
7.3.4	Real Datasets from Different Sequencing Platforms	125
7.4	Conclusion	127
8	Identifying Users behind Shared Accounts	129
8.1	Introduction	129
8.2	Problem Statement	132
8.3	Session-based Heterogeneous graph Embedding for User Identification	133

8.3.1	Framework Overview	133
8.3.2	Node Embedding in Heterogeneous Graph	134
8.3.3	Learning Node Features	135
8.3.4	Normalized Random Walk	136
8.3.5	Item-based Session Embedding	137
8.3.6	User Identification	138
8.3.7	Complexity Analysis	140
8.4	Experiments	141
8.4.1	Datasets and Experimental Settings	141
8.4.2	User Identification Performance	144
8.4.3	Study of Parameter Sensitivity	146
8.4.4	Item Recommendation with SHE-UI	148
8.5	Conclusions and Discussions	150
9	Modeling Heterogeneous Data in Social Media Posts for Sponsorship De-	
	tection	151
9.1	Introduction	151
9.2	Problem Statement	154
9.3	Methodology	155
9.3.1	Framework Overview	155
9.3.2	Aspect-Attentive Heterogeneous Post Encoder	156
9.3.3	Sponsorship Estimation and Ranking	161
9.3.4	List-wise Learning to Rank	162
9.3.5	Temporal Regularization	163
9.4	Experiments	163

9.4.1	Experimental Dataset	163
9.4.2	Experimental Settings	165
9.4.3	Comparative Baseline Methods	165
9.4.4	Experimental Results	166
9.4.5	Analysis and Discussions	170
9.5	Conclusion	173
10	Social Media User Geolocation via Hybrid Attention	175
10.1	Introduction	175
10.2	Hybrid-attentive User Geolocation	177
10.2.1	Multi-head Graph Attention Network	178
10.2.2	Language Attention Network	179
10.2.3	Hybrid Attention	180
10.2.4	Location-regularized User Geolocation	180
10.3	Experiments	181
10.3.1	Experimental Setup	181
10.3.2	Experimental Results	183
10.4	Conclusion	185
11	Reformulation Inference Network for Context-Aware Query Suggestion	186
11.1	Introduction	186
11.2	Problem Statement	189
11.3	Reformulation Inference Network	190
11.3.1	Framework Overview	190
11.3.2	Distributed Reformulation Representation	191

11.3.3	Query Session Encoder	193
11.3.4	Reformulation Inferencer	194
11.3.5	Query Discrimination and Generation	195
11.3.6	Learning and Optimization	196
11.4	Experiments	198
11.4.1	Datasets and Experimental Settings	198
11.4.2	Experimental Results	200
11.4.3	Analysis and Discussions	203
11.5	Conclusion	207
12	Learning to Disentangle Interleaved Conversations	208
12.1	Introduction	208
12.2	Conversation Disentanglement	211
12.2.1	Problem Statement	211
12.2.2	Framework Overview	212
12.2.3	Message Pair Selection	213
12.2.4	Similarity Estimation with the Siamese Hierarchical CNN (SHCNN)	214
12.2.5	Conversation Identification by SIMilarity Ranking (CISIR)	218
12.3	Experiments	220
12.3.1	Datasets and Experimental Settings	220
12.3.2	Pairwise Similarity Estimation	222
12.3.3	Conversation Identification	224
12.4	Conclusion	225
13	Enhancing Air Quality Prediction with Online Community Behaviors in Social Media	227

13.1	Introduction	227
13.2	Air Quality Prediction with Social Media and NLP	229
13.2.1	Stage 1: Tweet Filtering	230
13.2.2	Stage 2: Feature Extraction	231
13.2.3	Stage 3: Air Quality Prediction	232
13.3	Experiments	232
13.3.1	Experimental Settings	232
13.3.2	Experimental Results	234
13.4	Conclusion and Discussions	235
14	Conclusion	237
A	Appendix in QDS-Transformer	239
A.1	Experimental Details	239
A.1.1	Experimental Datasets	239
A.1.2	Experimental Settings	240
A.1.3	Evaluation Scripts	241
A.2	Baseline Methods	241
B	Appendix in MARU	244
B.1	The proof of Corollary 5.1	244
C	Appendix in CANTOR	245
C.1	Proof of Theorem 6.1	245
C.2	Proof of Theorem 6.2	245
D	Appendix in TahcoRoll	247

D.1 Proof of Proposition 7.1	247
D.2 Proof of Proposition 7.2	248
D.3 Proof of Proposition 7.3	249

LIST OF FIGURES

1.1	Illustration of the three-layer approach for human behavior modeling with multidisciplinary and heterogeneous data.	2
1.2	Illustration of representation learning for heterogeneous data.	3
1.3	Examples with incorporated structural semantics in heterogeneous data.	4
1.4	Illustration of the universal latent space and the aspect attention.	5
1.5	Illustration of the atomic blackboxes of representation learning and compositional operations for human behavior modeling.	7
3.1	An example illustration of the attention mechanism used in Query-Directed Sparse Transformer.	23
3.2	The overall schema of our proposed QDS-Transformer.	28
3.3	The performance of QDS-Transformer on TREC-19 DL track dataset with different local attention window sizes w	34
3.4	The average maximum attention scores to different types of tokens over Transformer layers.	37
3.5	The average entropy scores of attention distributions for different token types over Transformer layers.	37
4.1	The schema of the proposed framework, <i>Junction Encoder with Deep Interaction</i> (JEDI), using the gene NM_001080433 with six exons as an example, where the second exon forms backsplicing. A_i and D_j represent the i -th and j -th potential acceptors and donors.	44
4.2	The illustration of the attentive encoder for acceptor junctions. Note that the donor junction encoder shares the same model structure with different model parameters.	45

4.3	The ROC curves for zero-shot backsplicing discovery based on the 5-fold cross-validation and JEDI trained for gene-level circular RNA prediction.	57
4.4	The isoform-level circular RNA prediction performance of JEDI with different flanking region lengths L based on the 5-fold cross validation. We report the mean for each metric and apply error bars to indicate standard deviations.	58
4.5	The isoform-level circular RNA prediction performance of JEDI with different k -mer sizes K based on the 5-fold cross validation. We report the mean for each metric and apply error bars to indicate standard deviations.	59
5.1	The schema of the proposed framework Meta-context Aware Random Walks (MARU).	67
5.2	The illustrations of classical random walks and our proposed bidirectional extended random walks for learning network representations. The yellow nodes are the starting nodes of random walks while the white nodes with dotted strokes are the extended nodes. The lines are the sliding windows for the corresponding nodes for optimization.	69
5.3	Performance of different methods for the multi-label node classification task in three datasets. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test. Note that the Micro-F1 scores do not increase with more labeled nodes in some cases because of the imbalance of class distribution.	77
5.4	Performance of different methods for the node clustering task in three datasets. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.	79
5.5	The macro-F1 scores of MARU with classical random walks and our proposed bidirectional extended random walks with 50% of training labeled nodes in the task of multi-label node classification in DBIS and MovieLens.	81

5.6	The Macro-F1 scores of MARU as a function of percentage of labeled training data and sliding window size for meta-contexts in Yelp.	82
6.1	The general framework of the proposed <u>clustering</u> and <u>navigating</u> for <u>top-K</u> recommenders (CANTOR).	91
6.2	The distributions of users and items over different degrees in the Amazon dataset.	94
6.3	The ratios of speedup and the P@1 scores of CANTOR over different numbers of sampled users u in six datasets.	103
6.4	The ratios of speedup and the P@1 scores of CANTOR over different sizes of the hyperparameter efs in six datasets.	104
6.5	The ratios of speedup and the P@1 scores of CANTOR over different numbers of affinity groups r in six datasets.	105
6.6	The preparation time of CANTOR with (w/) and without (w/o) the adaptive representative selection method (ARS) in six datasets.	106
7.1	An example with five signatures and two sequencing reads in signature profiling. Each segment represents an occurrence of the corresponding signature in the read.	111
7.2	The automaton of AC with five signatures. Black solid links are trie links, and red dashed links are failure links. Colored nodes and thicker links are traversed while profiling the read ATTTG.	112
7.3	The binarized representations for five patterns and two sequencing reads. Two signatures GA and TC share the same binarized pattern (red). A substring in a sequencing read ATCG has the identical binarized form to the signature AGAT (blue).	115
7.4	An example of the thinned automaton of AC with five signatures. Black solid links are trie links, and red dashed links are failure links. The <i>yellow</i> node represents two signatures GA and TC.	115

7.5	Run-time (left) and memory (right) for constructing the automaton given 16 sets of k -mer patterns. A lower value represents a more efficient approach. PlainAC.C++ maxes out the memory capacity while constructing the “large” batch of 24 million k -mers (large_240), and thus its recorded time and memory are truncated. TahcoRoll consistently requires less time and memory than PlainAC.Py	120
7.6	Run-time (x-axis) and memory (y-axis) of counting <i>small</i> batches of k -mers on synthetic reads of 75bp. Each point represents a pair of measurements (run-time and memory). Points located on the lower left corner of each plot indicate more efficient approaches. The top panel examines three different read sets with 1.2 million k -mers; the bottom panel examines three different k -mer sets with 34,497,448 reads.	121
7.7	Multi-threads evaluation of MSBWT, KMC3, Jellyfish and TahcoRoll on profiling 86,976,737 synthetic reads of 180bp with <i>wide</i> batches k -mers.	124
8.1	The framework overview of SHE-UI.	133
8.2	An example of heterogeneous graph construction with a session that begins with three songs by Lady Gaga and ends with a song by Taylor Swift.	134
8.3	An illustration for the procedure of normalized random walks. The walk is going to be transited from p and evaluating transition probabilities of neighbor nodes.	137
8.4	The percentage of accounts over different numbers of account users in the real-word KKBOX dataset.	142
8.5	The percentage of session pairs sharing items or metadata within accounts in the KKBOX dataset. A pair of sessions can be issued by the same user or two different users.	142
8.6	The performance of determining the cluster number. The lower MAE and RMSE indicate the better performance. All improvements are statistically significant at 95% confidence level by a paired t-test.	144
8.7	Results of Parameter Sensitivity Study.	147

8.8	Results of recommendation by varying k and α	149
9.1	An example of paid media that fails to disclose sponsorship. Despite the influencer advertises a product and mentions a certain brand name, no sponsorship is disclosed.	152
9.2	The overall framework of the SPoD. The aspect-attentive heterogeneous post encoder generates post representations. The estimated sponsorship scores are optimized by conducting temporal regularization for detecting sponsored posts. .	155
9.3	The illustration of the post encoder using heterogeneous information.	156
9.4	Precision at retrieval rates of the proposed model and the baseline methods. SPoD shows efficient and robust ranking performance compared to the baseline methods.	168
9.5	Examples of successfully detected sponsored posts with absence of sponsorship disclosure, and highly ranked non-sponsored posts.	170
9.6	The post node features and the text features play an important role in detecting sponsorship of posts. SPoD significantly improves the performance in the short post set.	171
9.7	The MAP scores of methods with different caption lengths of posts.	172
10.1	The overview of the proposed framework, HUG.	177
10.2	Attention weight analysis. (a)-(b) Documents of users A and B. (c)-(d) Geolocations and attention weights of users C and D with their one-hop neighbors. . . .	184
11.1	The schema of the proposed Reformulation Inference Network (RIN).	190
11.2	An example of the heterogeneous network constructed by a search session of four queries for deriving term embeddings. Note that the queries in the graph are auxiliary nodes connecting the domains of terms and websites.	192
11.3	The MRR improvement of four methods over the MPS baseline method with different context lengths.	202

11.4	The MRR performance of RIN with either or both of homomorphic query and reformulation embeddings. Q denotes the homomorphic query embeddings while R indicates the embeddings of reformulations.	203
11.5	The MRR performance of RIN with homomorphic and non-homomorphic embeddings. Note that RIN _{NH} replaces the original homomorphic query embeddings with non-homomorphic query embeddings from node2vec.	204
11.6	The MRR performance of RIN with and without the reformulation inferencer. Note that RIN (-I) removes the reformulation inferencer.	205
11.7	The MRR performance of RIN over different numbers of embedding dimensions.	206
12.1	A segment of real-world conversations involving six users and five (annotated) threads from the IRC dataset.	209
12.2	Illustration of our proposed two stage method. In the first stage, (1a) message pairs are selected for (1b) estimating pairwise similarity with a Siamese hierarchical CNN (SHCNN). In the second stage, (2) the algorithm of conversation identification by similarity ranking (CISIR) constructs a graph with strong relationships among messages and finds conversations as connected components. . .	212
12.3	The percentage of messages in the same conversation as a given message with elapsed time between messages no greater than i hours for four experimental datasets.	213
12.4	Illustration of hierarchical CNN (HCNN) for message representation. The labels with a larger font size indicate the corresponding tensors, and the labels with a smaller font size explain the operations between tensors.	215
12.5	The siamese hierarchical CNN (SHCNN) for similarity estimation. Note that the model structure of an HCNN is shown in Figure 12.4.	216

12.6	An example message pair in two different conversations from IRC shows how SHCNN discriminates between messages on different topics. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 0.67% of being in the same conversation for this pair while DeepQA with single-layer CNNs predicts 69.81%.	223
12.7	An example message pair in a conversation from IRC shows how SHCNN captures similarity in local information. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 70.41% for this pair while ABCNN with multiple-layer CNNs predicts 36.50%.	224
13.1	The framework of the proposed approach.	229
13.2	Micro F1 scores with four-class categorization. All of the improvements of our approach over the baseline method are significant with a paired t-test at a 99% significance level.	235
13.3	Macro F1 scores with four-class categorization. All of the improvements of our approach over the baseline method are significant with a paired t-test at a 99% significance level.	235

LIST OF TABLES

3.1	The statistics of the experimental datasets.	30
3.2	The ad-hoc retrieval performance of our approach and baseline methods on the TREC-19 DL track benchmark. Note that those baselines with higher MAP scores are all full retrieval and benefited from additional data engineering like query expansion.	32
3.3	The few-shot learning retrieval performance of different methods on three benchmark datasets. NDCG and ERR are at cut-off 20.	33
3.4	The retrieval performance of different models on the TREC-19 DL track benchmark dataset with different global attention patterns. Q and S indicate the usage of query and sentence global attention. Note that QDS-Transformer with no global attention is equivalent to Sparse-Transformer.	33
3.5	Efficiency Quantification. The local attention window size is shown in parentheses. Q and S indicate the usage of only query and sentence attention. Sparsity is compared with fully attention at same text length.	35
3.6	Case study of two queries on the sentences with the highest attention weights in the last transformer layer over different heads for the [CLS] token.	37
3.7	Case study of the query 1112341 on the sentences in the document D1641978 with the highest attention weights among all heads from two query tokens. Note that we use attention weights in the third transformer layer.	38
3.8	Case study of the query 833860 with the query tokens with the highest attention weights in the 10-th transformer layer among all heads from the [SOS] tokens of sentences in the document D2944963.	38
4.1	Evaluation of isoform-level circular RNA prediction based on the 5-fold cross-validation. We report the mean and standard deviation for each metric.	54

4.2	Evaluation of gene-level circular RNA prediction based on the 5-fold cross-validation. We report the mean and standard deviation for each metric.	55
4.3	Independent study of isoform-level circular RNA prediction for mouse circRNAs based on the models trained on human circRNAs.	56
4.4	Run-time analysis on isoform-level circular RNA prediction in seconds (s), minutes (m), and hours (h), based on the 5-fold cross-validation. We report the mean of the training time (over five folds).	60
5.1	Summary of notations and their descriptions.	68
5.2	The statistics of three experimental datasets of heterogeneous networks.	74
5.3	The statistics of three datasets for the task of multi-label node classification.	77
5.4	The <i>AUC</i> scores of different methods with four operators for link prediction in three datasets.	80
5.5	The classification performance of MARU over different walk lengths l of bidirectional extended random walks with 50% of training labeled nodes in Yelp. Note that the length of generated random walks is $2 \times l + 1$ because MARU conducts random walks bidirectionally.	83
5.6	The classification performance of MARU over different sizes of embedding dimensions d with 50% of training labeled nodes in Yelp.	84
6.1	The statistics of six experimental datasets. Note that the personalized link prediction problem can be mapped to an item recommendation problem by treating each user as an item and recommending other users to a user in a similar way to that of recommending items to a user, and in this case the numbers of users and items are equal.	99

6.2	Comparisons of top- K recommendation results on six datasets in two tasks. Note that $P@K$ measures the precision of approximating the top- K recommendations of full inner-product computations. SU indicates the ratio of <u>speedup</u> based on the original full inner product time of inferring top- K recommendations. For example, 9.4x means the computation time of the method is 9.4 times faster than the full inner product computation time. PT means the preparation time and IT represents the inference time in prediction process. The time units of <i>seconds</i> , <i>minutes</i> , and <i>hours</i> are represented as s, m, and h, respectively. Computation time of the full inner product method for each dataset is 71s (MovieLens), 1,017s (Last.fm), 92,828s (Amazon), 56,824s (Youtube), 71,653s (Flickr), and 72,723s (Wikipedia).	102
7.1	Time (Hour) and memory (GB) of synthetic signatures over different read sets. Dagger (†) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.	123
7.2	Time (Hour) and memory (GB) of synthetic reads over different k -mer sets. Dagger (†) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.	123
7.3	Time (Hour) and memory (GB) of profiling synthetic reads over different <i>wide</i> batch of k -mer sets. Dagger(†) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.	125
7.4	Evaluation of real datasets across different sequencing platforms. MSBWT, KMC3, and Jellyfish are run with eight threads. Fold-change is relative to the measurements reported by eight-thread TahcoRoll.	126
7.5	Evaluation of different binarized representations. Time is reported in hour and memory is reported in gigabyte. Nucleotides can be divided into balanced or unbalanced partitions. The p -values are computed through paired t -tests on time against the default setting: $[\{A, G\}, \{C, T\}]$, and adjusted by Bonferroni correction.	126

8.1	Statistics of Two Datasets.	142
8.2	The results of user identification behind shared accounts. The higher values in metrics indicate the better performance. All improvements of SHE-UI against DW [264] are statistically significant at 95% confidence level by a paired t-test.	145
8.3	Results of item recommendation with AUREc.	149
9.1	Node features that represent characteristics of each type of node in GCNs.	160
9.2	Performance comparison with the baseline methods. SPoD significantly outperforms all types of baseline methods. The temporal regularization and aspect-attentive components improve the ranking performance.	167
9.3	Precision results on the top-ranked unknown posts. SPoD outperforms other baseline methods in detecting undisclosed sponsorship of the unknown posts.	169
10.1	Dataset statistics.	182
10.2	Twitter user geolocation prediction performance.	183
10.3	Ablation study on TWITTER-US.	183
11.1	The statistics of queries with different context lengths in the training and testing datasets.	199
11.2	The MRR performance of different methods in the testing sets with different context lengths for the task of discriminative query suggestion. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.	201
11.3	The PER performance of three methods. The improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.	203
12.1	Statistics of four datasets after pre-processing.	222

12.2	Performance of pairwise similarity estimation in four datasets. Our approach is denoted as SHCNN. The performance with only low-level or high-level representations are denoted as SHCNN (L) and SHCNN (H). All improvements of SHCNN against the best baseline are significant at the 1% level of significance in a paired <i>t</i> -test.	222
12.3	Performance of conversation disentanglement in four datasets. Our approach is denoted as CISIR. “Oracle” indicates the optimal performance if CISIR correctly retrieves all message pairs in identical conversations. All improvements of CISIR against the best baseline are significant at the 1% level of significance in a paired <i>t</i> -test.	225
13.1	Categorization of AQI from EPA.	230
13.2	Statistics of five experimental datasets. The relevant tweets refer to the remaining tweets after the stage of tweet filtering.	233
13.3	The overall classification performance of the baseline methods and our approach. All of the improvements of our approach (ours) over PAQI are significant with a paired <i>t</i> -test at a 99% significance level.	234
A.1	Number of parameters for methods.	241

ACKNOWLEDGMENTS

The achievement of my Ph.D. journey and the work presented in this thesis could not have been possible without the help of my advisor and committee members. First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Wei Wang, for her continuous support and guidance throughout my entire Ph.D. study. She always fully supports all of my decisions and gives me lots of freedom to pursue my research interests with sufficient resources and grow into an independent researcher. Besides research, she also teaches me mind-sets and skill-sets to be more enthusiastic and professional in academia. I will also always remember her encouragement that takes me to overcome every frustration and painful moment in my Ph.D.

In addition to my advisor, I want to thank my committee members, Prof. Kai-Wei Chang, Prof. Cho-Jui Hsieh, and Prof. Yizhou Sun for their advice and suggestions. They not only provide valuable insights into many of my research projects, but also suggest various helpful comments for me to pursue a successful academic career. The collaborations and discussions with them are some of the most exciting parts of my Ph.D. study.

Outside the computer science department at UCLA, I am fortunate to collaborate with many faculty members from diverse domains in grant projects, including Prof. Sean Young, Prof. Chen Li, Prof. Peipei Ping, and Prof. Andrea Bertozzi. These opportunities inspire me to conduct more interdisciplinary research and pursue broader impacts to be a force for social good in our world. Besides, these collaborations also enable me to learn great leadership skills to manage teams at different scales.

In the industry, I am also fortunate to have many mentors. Our collaborations result in great impacts on my research and contribute to several characters in this thesis. When I was at FXPAL, Dr. Francine Chen and Dr. Yan-Ying Chen first introduced me to research problems about modeling human interactions. At Google Research, Dr. Mingyang Zhang, Dr. Cheng Li, and Dr. Michael Bendersky opened the line of my research on long-form docu-

ment modeling. Dr. Tao Wu and Dr. Georgios Roumpos showed me how to build a pipeline for industry-scale experiments. At Microsoft Research, Dr. Chia-Jung Lee, Dr. Longqi Yang, Dr. Brent Hecht, and Dr. Jaime Teevan encouraged me to combine computer science and psychology to more deeply model human behaviors. Dr. Chenyan Xiong gave me various helpful insights and guidance in deep information retrieval, especially for text modeling and setting up benchmark environments.

During my Ph.D., I am grateful to have opportunities to collaborate on diverse research topics with many talented colleagues in ScAI Lab and other research groups at UCLA, including Ruirui Li, Chelsea Ju, Seungbae Kim, Patrick Chen, Zeyu Li, Joey Zhou, Guangyu Zhou, Junheng Hao, Cheng Zheng, Xiusi Chen, Jieyu Zhao, Masaki Nakada, and Zehan Chao. These collaborations result in several publications and also contribute to some chapters in this thesis. I enjoyed the time working with them and learned a lot from our collaborations.

I also want to express great appreciation to my advisors at National Taiwan University and Academic Sinica before joining UCLA, Prof. Pu-Jen Cheng and Prof. Cheng-Te Li. Pu-Jen brought me into the world of research when I was a junior undergraduate. He always believes in my potential and inspires me with his research enthusiasm. Cheng-Te broadened my research into more diverse domains. I learned a lot from him about conducting research with real-world impacts. The collaborations with them also have provided me a strong capability and basis for accomplishing my Ph.D. study.

I would like to thank all of my friends for their friendship and help over these years. I give my gratitude to my roommates at UCLA, Bo-Jhang Ho, Michael Tu, and Junheng Hao. Bo-Jhang and Michael helped me have a wonderful start to begin my Ph.D. life in Los Angeles. I always enjoyed our chats every night about interesting discoveries from three researchers in different domains. My second half of my Ph.D. is also amazing with Junheng by exploring new culinary delights and fun places to go. As colleagues in the same lab, we also shared many important moments in our Ph.D. journeys. I want to thank Shyr-Shea Chang and Chi-Chi Huang for having many gatherings and meals together. I am grateful to be roommates of Hsin-Yu Liu, Shu-Hao Yu, Chun-Sung Ferng, and Hsiu-

Wen Hsueh during my internships. I would not have those joyful summers in the bay area without hanging out and traveling with them. I also want to thank many friends from Taiwan who continuously offer their warm encouragement during my Ph.D. journey, including Chun-Pai Yang, Jyun-Jhe Chou, Pi-Hsun Shih, Chin-Hua Tsai, Shang-En Huang, Chun-Sung Ferng, Hsiu-Wen Hsueh, Chung-Yi Li, Jie-Wei Wu, Chih-Chung Kuo, Chen-Yu Yang, Shao-Chuan Lee, Hanting Chiang, Wenfeng Cheng, Ju-Cheng Chou, Robert Wang, Wei-Che Chang, Chia-Jung Hsu, and *Friends of Semi* (Hsu-Feng Cheng, Lun-Kai Hsu, Bir-Die Huang, Zi-Wei Liao, and Zhi-Ye Yu). From another perspective of encouragement, I am especially thankful to Yui Horie and Aqours for their songs and concerts that have accompanied me through these years.

Last and most importantly, I owe the highest debt of gratitude to my loving family, my mom Shu-Chen Huang, my dad Ruey-Hung Jiang, and my sister Jia-Yun Jiang, for their unconditional and endless love and faith. I could not have accomplished this journey without their unstinting support and never-ending encouragement as they have always given me the courage to overcome every challenge.

VITA

- 2009 – 2013 B.Sc. (Computer Science and Information Engineering),
National Taiwan University, Taipei, Taiwan
- 2013 – 2015 M.S. (Computer Science and Information Engineering),
National Taiwan University, Taipei, Taiwan
- 2017 Research Intern, FX Palo Alto Laboratory, Palo Alto, California
- 2018, 2019 Research Intern, Google Research, Mountain View, California
- 2020 Research Intern, Microsoft Research, Redmond, Washington
- 2017 – 2020 Teaching Assistant and Associate, Computer Science Department, UCLA
- 2016 – 2021 Graduate Student Researcher, Computer Science Department, UCLA

PUBLICATIONS

Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee and Wei Wang. “Long Document Ranking with Query-Directed Sparse Transformer,” in *Proceedings of 2020 Conference on Empirical Methods in Natural Language Processing* (EMNLP 2020).

Jyun-Yu Jiang, Chelsea J.-T. Ju, Junheng Hao, Muhao Chen and Wei Wang. “JEDI: Circular RNA Prediction based on Junction Encoders and Deep Interaction among Splice Sites,” in *Proceedings of The 29th annual international conference on Intelligent Systems for Molecular Biology and The 20th annual European Conference on Computational Biology* (ISMB/ECCB 2021). Bioinformatics, OUP, 2021.

Jyun-Yu Jiang, Zeyu Li, Chelsea J.-T. Ju and Wei Wang. “MARU: Representation Learn-

ing for Heterogeneous Networks with Meta-context Aware Random Walks,” in *Proceedings of The 29th ACM International Conference on Information and Knowledge Management (CIKM 2020)*.

Jyun-Yu Jiang*, Patrick H. Chen*, Cho-Jui Hsieh and Wei Wang. “Clustering and Constructing User Coresets to Accelerate Large-scale Top-K Recommender Systems,” in *Proceedings of the 2020 World Wide Web Conference (WWW 2020)*.

Jyun-Yu Jiang, Cheng-Te Li, Yian Chen and Wei Wang. “Identifying Users behind Shared Accounts in Online Streaming Services,” in *Proceedings of The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018)*.

Jyun-Yu Jiang and Wei Wang. “RIN: Reformulation Inference Network for Context-Aware Query Suggestion,” in *Proceedings of The 27th ACM International Conference on Information and Knowledge Management (CIKM 2018)*.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen and Wei Wang. “Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking,” in *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*.

Jyun-Yu Jiang, Xue Sun, Wei Wang and Sean Young. “Enhancing Air Quality Prediction with Social Media and Natural Language Processing,” in *Proceedings of The 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*.

Cheng Zheng*, **Jyun-Yu Jiang***, Yichao Zhou, Sean Young and Wei Wang. “Social Media User Geolocation via Hybrid Attention,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*.
(*Equal contribution)

Seungbae Kim, **Jyun-Yu Jiang** and Wei Wang. “Discovering Undisclosed Paid Partnership on Social Media via Aspect-Attentive Sponsored Post Learning,” in *Proceedings of The 14th ACM International Conference on Web Search and Data Mining (WSDM 2021)*.

CHAPTER 1

Introduction

We, humans, are indeed the most sophisticated species on this planet so that understanding complex human behaviors remains one of the most critical challenges in many domains, including web applications, social science, medicine, health care, and biology. Fortunately, technology development in recent decades not only boosts our capability to collect immense human data from various resources but also fuels unprecedented computational power to manage and analyze massive datasets. The massive amount of heterogeneous data generated from myriad motherlodes every day creates an excellent opportunity to unravel complex and ambiguous behaviors at multiple scales from different angles.

Although there are several forces in both academia and industry for modeling human behaviors in many fields, existing work has often focused on certain types of data and/or specific behaviors, and therefore lacks generalizability to a broader range of data and behaviors. Moreover, most existing methods are too sophisticated to integrate with one another, making it impossible to assemble a unified system that can handle data from heterogeneous sources and behaviors across multiple scales. These deficiencies substantially limit the utilities and potential impacts of human behavior modeling. Indeed, there is an urgent need to develop a robust end-to-end system that is capable of effectively and efficiently modeling human behaviors at different scales using multidisciplinary data.

In this dissertation, we aim at improving mining of broad multidisciplinary data to remove these barriers in learning complex human behaviors for real-world applications, thereby enhancing human productivity and well-being. Specifically, our research advances the whole vertical ecosystem of human behavior modeling in three research layers as shown in Fig-

ure 1.1, including heterogeneous data harnessing, interdisciplinary knowledge discovery, and human behavior modeling. We first develop novel computational methods to surmount obstacles in deriving machine-friendly representations of heterogeneous and multidisciplinary data for both effectiveness and efficiency. We then leverage these semantic-rich representations to model complicated human behaviors by mining knowledge from interdisciplinary resources, thereby benefiting real-world applications for improving human productivity and well-being.

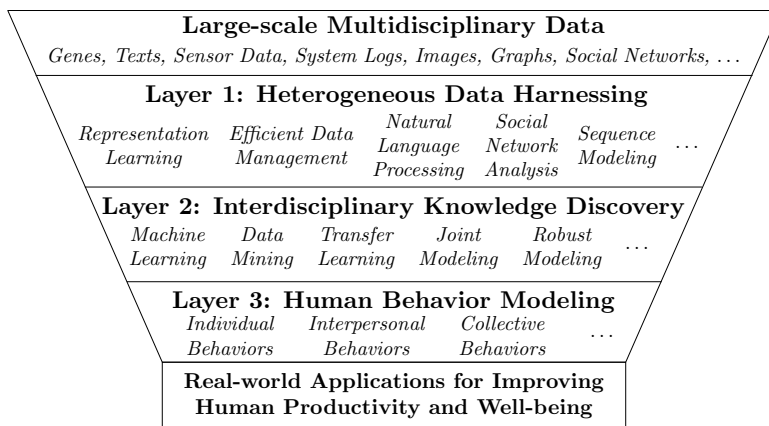


Figure 1.1: Illustration of the three-layer approach for human behavior modeling with multidisciplinary and heterogeneous data.

1.1 Heterogeneous Data Harnessing with Structural Semantics

To tackle heterogeneous data with distinct formats and complex semantics, one of the feasible approaches in the fields of modern machine learning is representation learning [32]. As shown in Figure 1.2, representation learning methods can be developed to encode heterogeneous data from various resources into machine-readable representations. These machine-readable representations are usually derived as continuous vectors so that machine learning methods can conveniently treat the representations as numerical features to learn appropriate hypotheses for downstream applications. In other words, developing effective and efficient representation learning frameworks has become one of the most key challenges to understand complicated heterogeneous data. Moreover, these frameworks can be also considered

as universal backbone components to address various real-world applications.

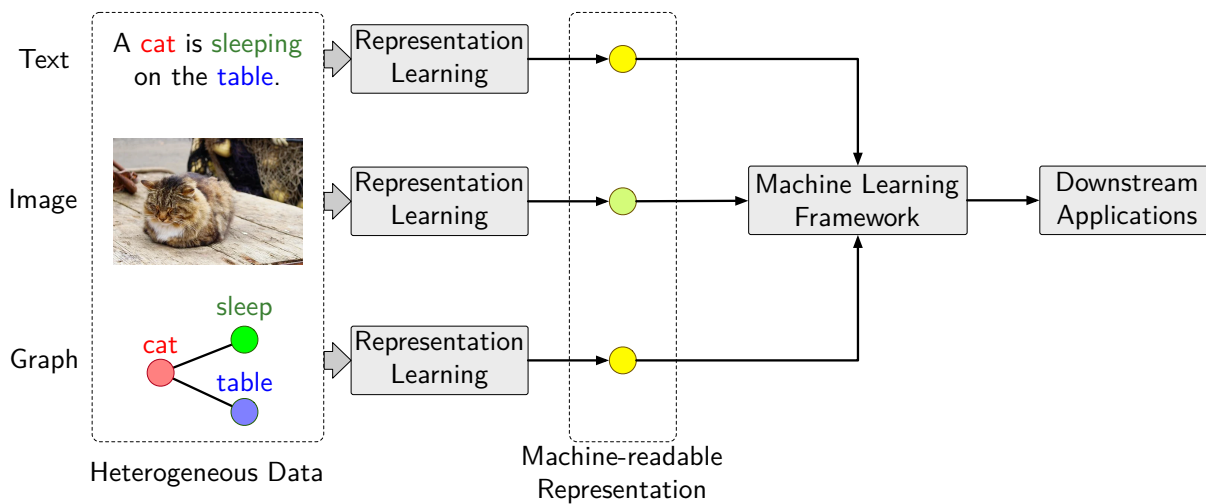
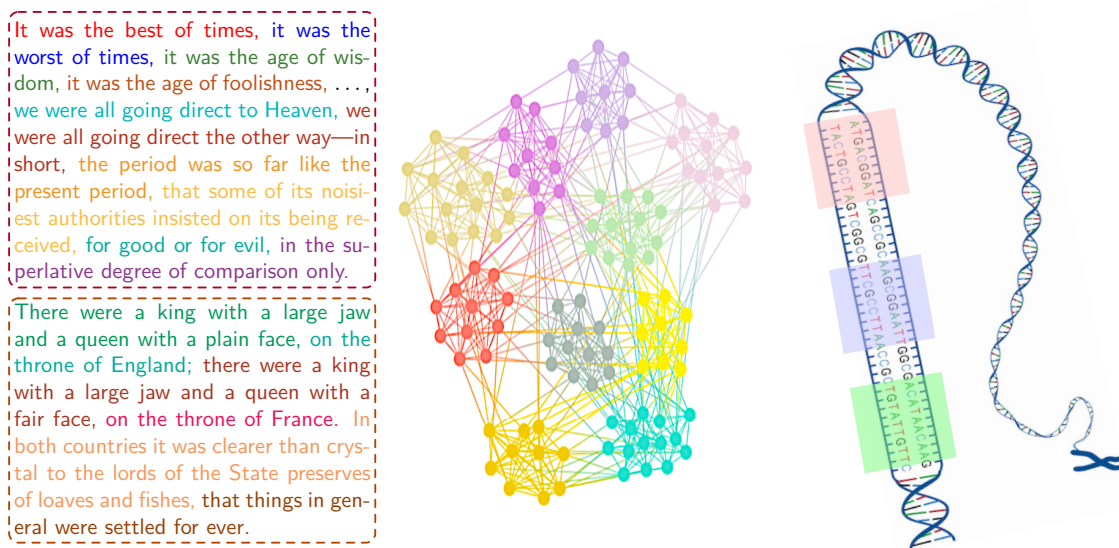


Figure 1.2: Illustration of representation learning for heterogeneous data.

One of the most significant common issues for modeling heterogeneous data is the sparsity. More specifically, large-scale data tend to have an extremely large data space, but the available data are usually limited. For instance, natural languages have extensive vocabulary lists and result in sparse corpora in many domains, such as natural language processing and information retrieval. Large-scale graphs usually have only limited links between nodes with low density. Although there are four types of nucleotides, genome sequences are intensely long so that phenotypic expressions can still be sparse. As a result, conventional representation learning methods can have a hard time deriving satisfactory and effective representations for sparse data. To address this issue, in this thesis, we propose to leverage structural semantics for representation learning. Specifically, we aim to encode structures with semantic meanings into derived representations. As shown in Figure 1.3, most of the heterogeneous data types could incorporate structural semantics. For example, text documents consist of paragraphs and sentences while graphs could have several communities of nodes with tighter connections. There are also some functional exons in genome sequences, which provide critical biological evidences and their relations to RNA splicing. With structural semantics, representation learning methods can not only absorb more knowledge into derived representations but also

leverage information across different data with similar semantic structures.



(a) Paragraph and sentence structure in texts. (b) Community structure in large-scale graphs. (c) Exon and intron structure in genome sequences.

Figure 1.3: Examples with incorporated structural semantics in heterogeneous data.

Besides the effectiveness, there are several efficiency issues when it comes to the era of big data. For example, the volume size of real-world data would reach more than 25ZB by 2025 with a significant growth rate [281]. Large-scale modern deep learning models also introduce the difficulty of training representation learning models. For instance, Switch Transformer [113] as a contextualized language model has 1.6T model parameters and requires 355 years to be appropriately pre-trained using one NVidia V100 GPU while ELMo [265] published in 2018 with 94M model parameters only needs an hour under the same situation. In addition, the high velocity of popular online applications leads to an enormous amount of incoming data generated every day [166] so that model efficiency becomes extensively important for rapidly training models and inferring results with repetitively updated data. To address the efficiency issues, in this thesis, we propose to leverage the structural semantics and navigate computations through the important and essential structure, thereby reducing redundant computations and improving the model efficiency.

1.2 Interdisciplinary Knowledge Discovery

Based on satisfactory representations derived from heterogeneous data, the next step of this thesis is to develop novel machine learning and data science approaches for discovering precious knowledge in data from different domains. This task is challenging because heterogeneous data can be generated in completely distinct formats. For instance, a social media post can simultaneously contain images, texts, and information about social networks while each data type has different appropriate representation learning methods. In other words, representations can be in different embedding spaces so that valuable knowledge cannot be leveraged across heterogeneous data. In addition, resources can also have different importance over diverse circumstances. To address the above issues, we propose to learn representations in a universal latent space and estimate the importance of each resource with the aspect attention.

As shown in Figure 1.4, we first project heterogeneous data into a shared embedding space so that representations can be comparable to each other. Specifically, representations in the universal latent space can be derived by extending individual representation learning methods for heterogeneous data with non-linear projections [193, 375] and regularization [177].

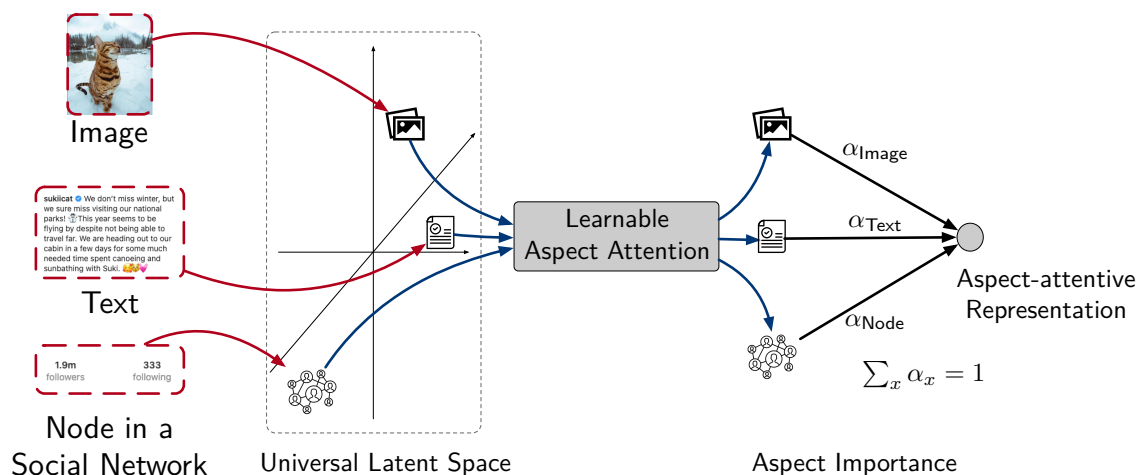


Figure 1.4: Illustration of the universal latent space and the aspect attention.

To integrate knowledge across heterogeneous resources, we propose the concept of aspect attention. More precisely, a well-designed estimator function can appropriately compute

the corresponding importance scores α_x for all resources x according to their representations \mathbf{r}_x . For example, a context vector \mathbf{r}^c can be trained to estimate importance scores as:

$$\alpha_x = \frac{\exp(\langle \mathbf{r}_x, \mathbf{r}^c \rangle)}{\sum_y \exp(\langle \mathbf{r}_y, \mathbf{r}^c \rangle)},$$

where the importance of each resource is calculated by a dot product and normalized by a softmax function. The overall aspect attentive-representation can be further derived as $\sum_x \alpha_x \cdot \mathbf{r}_x$ for downstream machine learning applications.

1.3 Compositional and Explainable Behavior Modeling

With learned representations for heterogeneous data, there still could be a gap from these features to real-world applications in different domains. Moreover, various practical applications rely on domain knowledge of experts in the fields to reach more satisfactory results. To provide a general and ubiquitous solution for diverse real-world applications, we propose to treat representation learning methods as atomic blackboxes so that domain experts can conduct compositional and explainable operations to establish machine learning models for downstream applications.

As shown in Figure 1.5, different heterogeneous data with sparse and discrete formats can be encoded by corresponding representation learning methods into continuous vectors as informative and machine-readable representations. As a result, domain experts can treat representation learning methods as blackboxes that proceed atomic processes of obtaining useful representations as the basis for developing machine learning solutions. More precisely, these machine-readable and computable representations can be considered as puzzles for researchers in different fields to model complex human behaviors with compositional and explainable operations. For example, query reformulation in information retrieval leads to transitions between submitted queries so that reformulation behaviors can be modeled by compositional operations of computing directed Euclidean vectors between query representations [173]. With atomic blackboxes of representation learning and the concepts of

compositional operations, domain experts can not only more conveniently utilize the state-of-the-art representation learning techniques, but also have more interpretable models based on explainable operations.

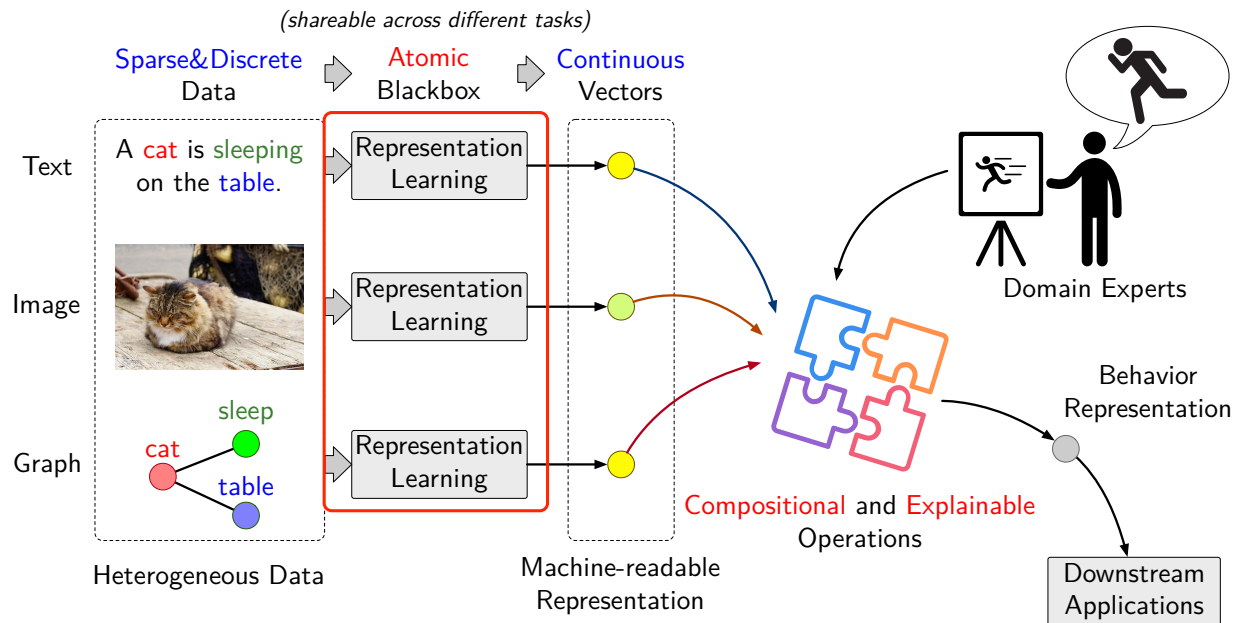


Figure 1.5: Illustration of the atomic blackboxes of representation learning and compositional operations for human behavior modeling.

1.4 Thesis Contributions

In this dissertation, we emphasize the importance of modeling complex human behaviors with multidisciplinary and heterogeneous data. Accordingly, we propose a series of machine learning methods that can fit in the three-layer framework as shown in Figure 1.1.

To effectively and efficiently model heterogeneous data and derive robust representations, we propose various deep learning approaches to tackle data from diverse domains. More specifically, we focus on leveraging *structural semantics* to not only encode more valuable knowledge but also reduce redundant computations. QDS-Transformer enhances the state-of-the-art transformer-based retrieval models by introducing the navigational global attention through the document structure, thereby improving relevance ranking and the ef-

efficiency in both training and inference stages. JEDI is the first work to model biological sequences with cross-attention layers and encode junctions in DNA sequences as the structural knowledge for circular RNA prediction. MARU leverages the node communities with similar type distributions in neighbors by considering meta-contexts during representation learning. CANTOR learns the coresets of user affinity groups with similar interests to accelerate the inference of latent factor models for top- K recommender systems. TahcoRoll indexes DNA sequences with a thinned Aho-Corasick automaton based on a binary structure for efficient k -mer signature profiling.

To jointly model heterogeneous data, we propose to learn representations in a universal latent space and conduct the aspect attention to integrate knowledge from different resources based their importance. SHE-UI constructs a heterogeneous graph of different entities so that graph representation learning methods can derive universal representations for identifying users behind shared accounts in online streaming services. SPoD and HUG apply the aspect attention to estimate the importance scores of heterogeneous resources on social media posts, such as texts, social networks, and images for sponsorship detection and user geolocation.

To model complex user behaviors at different scales, we present the general framework to treat representation learning methods as atomic blackboxes and demonstrate behaviors with compositional and explainable operations based on the valuable knowledge of domain experts. For individual behaviors, RIN is the first work to utilize homomorphic query embedding to derive the representations of user reformulation behaviors in web search with explainable Euclidean vector computations while heterogeneous graph embedding plays a role of the atomic blackbox to learn the base representations of queries, terms, and websites. For interpersonal behaviors between people, SHCNN conducts link operations between text messages based on the semantic similarity estimated by convolutional neural networks for disentangling interleaved conversational threads. For community behaviors that reflect environmental situations, we also propose to model text data to enhance air quality prediction for improving human well-being.

1.5 Thesis Overview

The rest of this dissertation can be logically categorized into the following segments.

Part I: Background and Survey. Chapter 2 provides the background survey and summarizes related work for each research problem in this dissertation.

Part II: Effective and Efficient Heterogeneous Data Harnessing. Based on the structural semantics, Chapters 3, 4, 5 describe our approach to improve the effectiveness of representation learning for multi-disciplinary data, including text documents, biological sequences, and heterogeneous networks. Chapters 6 and 7 further show how we reduce redundant computations to accelerate computational inference for recommender systems and signature profiling for biological sequences.

Part III: Interdisciplinary Knowledge Discovery. In this part, we demonstrate the use cases of the universal latent space and the aspect attention. Chapter 8 presents our work to learn universal representations based on heterogeneous graph embedding for identifying user behind shared accounts. Chapters 9 and 10 propose to apply the aspect attention to model heterogeneous resources in social media posts for sponsorship detection and user geolocation.

Part IV: Multi-scale Human Behavior Modeling. In this part, we present some examples of treating representation learning methods as atomic blackboxes and utilizing compositional and explainable operations to modeling human behaviors at different scales. Chapter 11 focuses on individual behaviors of query reformulation in web search. Chapter 12 discusses interpersonal behaviors in multi-party conversations for conversation disentanglement. Chapter 13 introduces our study to model community behaviors on social media for enhancing air quality prediction.

Part V: Conclusion. As the final part of this dissertation, Chapter 14 concludes our contributions with a summary of our works.

CHAPTER 2

Related Work

In this chapter, we present previous studies related to our work as the background on various applications of human behavior modeling with heterogeneous data.

2.1 Neural Language Models for Information Retrieval

Neural models have demonstrated significant advances across various ranking tasks [138]. Early approaches investigated diverse ways to capture relevance between queries and documents [84, 137, 162, 350]. And recently the state-of-the-art in many text ranking tasks has been taken by BERT or other pretrained language models [80, 83, 95, 250, 251, 353], when sufficient relevance labels are available for fine-tuning (e.g., on MS MARCO [25]).

The improved effectiveness comes with the cost of computing efficiency with deep pre-trained transformers, especially on long documents. This stimulates studies investigating ways to retrofit long documents to BERT’s maximum sequence length limits (512). A vanilla strategy is to truncate or split the documents: Dai and Callan [83] applied BERT ranking on each passage segmented from the document independently and explored different ways to combine the passage ranking scores, using the score of the first passage (BERT-FirstP), the best passage (BERT-MaxP) (also studied in Yan et al. [351]), or the sum of all passage scores (BERT-SumP).

More sophisticated approaches have also been developed to introduce structures to transformer attentions. Transformer-XL employs recurrence on a sequence of text pieces [85], Transformer-XH [372] models a group of text sequences by linking them with eXtra Hop attention paths, and Transformer Kernel Long (TKL) [155] uses a sliding window over the

document terms and matches them with the query terms using matching kernels [350].

On the efficiency front, Kitaev et al. [201] proposed Reformer that employed locality-sensitive hashing and reversible residual layers to improve the efficiency of Transformers. Child et al. [65] introduced sparse transformers to reduce the quadratic complexity to $O(L\sqrt{L})$ by applying sparse factorizations to the attention matrix, making the use of self-attention possible for extremely long sequences. Subsequent work [79, 304] leverage a similar idea in a more adaptive way. Combining local windowed attention with a task motivated global attention, Beltagy et al. [31] presented Longformer with an attention mechanism that scales linearly with sequence length.

2.2 Circular RNA Prediction

Current works to discover circular RNA can be divided into two categories: one relies on detecting back-spliced junction reads from RNA-Seq data; the other examines features directly from transcript sequences.

The first category aims at detecting circRNA from expression data, specifically from RNA-Seq reads. It is mainly achieved by searching for chimeric reads that join the 3'-end to the upstream 5'-end with respect to a transcript sequence [30]. Existing algorithms include **MapSplice** [331], **CIRCexplorer** [369], **KNIFE** [309], **find-circ** [237], and **CIRI** [123, 124]. These algorithms can be quite sensitive to the expression abundance, as circRNAs are often lowly expressed and fail to be captured with low sequencing coverage [30]. In the comparison conducted by Hansen et al. [145], the findings suggest dramatic differences among these algorithms in terms of sensitivity and specificity. Other caveats are reflected in long duration, high RAM usage, and/or complicated pipeline.

The second category focuses on predicting the circRNA based on transcript sequences. Methods in this category leverage different features and learning algorithms to distinguish circRNA from other lncRNAs. **PredicircRNA** [258] and **H-ELM** [60] develop different strategies to extract discriminative features, and employ conventional statistical learning al-

gorithms, i.e. multiple kernel learning for PredicircRNA and hierarchical extreme learning machine for H-ELM, to build a classifier. Statistical learning approaches require explicit feature engineering and selection. However, the extracted features are dedicated to specific facets of the sequence properties and present a limited coverage on the interaction information between the donor and acceptor sites. **circDeep** [54] and **DeepCirCode** [330] are two pioneering methods that employ deep learning architectures to automatically learn complex patterns from the raw sequence without extensive feature engineering. **circDeep** uses convolution neural networks (CNNs) with the bi-directional long short term memory network (Bi-LSTM) to encode the entire sequence, whereas **DeepCirCode** uses CNNs with max-pooling to capture only the flanking sequences of the back-splicing sites. Although **circDeep** has claimed to be an end-to-end framework, it requires external resources and strategies to capture the reverse complement matching (RCM) features at the flanking sequence and the conservation level of the sequence. In addition, the RCM features only measure the match scores between sites on the nucleotide-level, and neglect the complicated interaction between two sites. CNNs with max-pooling aim at preserving important local patterns within the flanking sequences. As a result, **DeepCirCode** fails to retain the positional information of nucleotides and their corresponding convoluted results.

Besides sequence information, a few conventional lncRNA prediction methods also present the potential of discovering circRNA through the secondary structure. **nRC** [114] extracts features from the secondary structures of non-coding RNAs and adopts CNNs framework to classify different types of non-coding RNA. **lncFinder** [143] integrates both the sequence composition and structural information as features and employs random forests. The learning process can be further optimized to predict different types of lncRNA. Nevertheless, none of these methods factor in the information specific to the formation of circRNAs, particularly the interaction information between splicing sites.

2.3 Network Representation Learning

Networks can be categorized into two types, including *homogeneous* and *heterogeneous* networks. Homogeneous networks contain a single node type, e.g., social networks of users, whereas heterogeneous networks involve multiple types of nodes, such as citation networks of authors, papers, and venues. Network representation learning for both categories aims at mapping nodes in graphs to low-dimensional continuous vectors. These low-dimensional vectors are learned to capture the essential information of the nodes, and consequently, better preserve the structure and semantic similarity among nodes.

A range of network representation learning algorithms has been proposed for homogeneous network embedding learning [59, 135, 264, 313, 325] and heterogeneous network embedding learning [57, 98, 122, 164, 312].

2.3.1 Homogeneous Network Embedding Models

DeepWalk [264] is a pioneering representation learning approach for homogeneous networks. It explores the network structure through the *random walks* sampled from the network. Mapping to the concepts in *word2vec* [241], nodes and random walks are treated as words and sentences, respectively. The node representations can be learned by using the vanilla skip-gram model [241] on the random walks. The paradigm of *DeepWalk* has inspired many studies [98, 135, 255, 335] that are applied to diverse types of networks. *node2vec* [135] is one of the examples that extend *DeepWalk* by relaxing the definition of network neighborhood and designing a biased random walk procedure to explore more diverse node representations. However, previous literature has demonstrated that such walk generation methods introduce a bias towards the nodes with higher degrees [306]. Therefore, the structural and semantic information of the isolated or less connected nodes becomes difficult to be captured by the model, which eventually leads to the inefficiency of the training procedure and poor accuracy of the trained representations of nodes with lower degree numbers. Most importantly, the model is prone to preserve only the global structure [325], assuming that nodes with more

common neighbors yield similar representations.

To better capturing the complicated underlying network structure, *LINE* [313] and *SDNE* [325] use edge-sampling algorithms to preserve both the local and global network structure. They model both the first-order proximity, defined as the proximity between directly connected nodes, and the second-order proximity, defined as the proximity between nodes that share common neighbors.

2.3.2 Heterogeneous Network Embedding Models

In order to comprehensively capture the rich semantics in edges and to better understand the different interactions between multi-typed nodes, heterogeneous information network embedding models are proposed. These methods either construct the embeddings for each modality defined beforehand, or learn all node embeddings together in the same latent space.

Most of the approaches that use predefined modality learn the node embeddings by minimizing the loss over each modality. HNE [57] presents a deep embedding framework that leverages a highly nonlinear multi-layered embedding function to capture the complex interactions. Each modality, such as image and text, is constructed separately. The embeddings of different modalities are then mapped to the same embedding space. Zhao et al. [374] specifically model the network structure of Wikipedia data that consists of three types of nodes: entities, words, and categories. It uses the coordinate matrix factorization technique to jointly learn the representations of these three types of nodes. PTE [312] is a semi-supervised representation learning method designed for text data. Based on the edge types, it decomposes the heterogeneous network into a set of bipartite networks. The method learns the embeddings of each node according to its one-hop neighbors, i.e. directly connected nodes, of the resulting bipartite networks.

To address the caveat of explicit node types, several approaches have been proposed to incorporate meta-paths, which are sequences of node types, for heterogeneous graph embeddings. For instance, metapath2vec [98] is another extension of DeepWalk that uses meta-paths to capture the relationships between different node types. More specifically, a

strategy for sampling random walks from heterogeneous networks is proposed to restrict random walks to follow particularly predefined transitions of node types. However, the set of meta-paths needs to be predefined while the selection of meta-paths significantly affects the performance. To avoid the requirement of meta-paths, Fu et al. [122] propose HIN2Vec to learn node representations by predicting the meta-paths as relations between nodes while Hussein et al. [164] manipulate the procedure of sampling random walks.

2.4 Collaborative Filtering for Recommender Systems

Collaborative filtering (CF) [93] is one of the most popular solutions for recommendation problems, including the task of top- K recommender systems. Moreover, the low-rank assumption in CF further leads to the prominence of latent factor models or matrix factorization (MF) [203]. For example, Kang et al. [187] exploited MF models to optimize numerical ratings for top- K recommenders while Rendle et al. [278] observed pairwise implicit feedback in a one-class preference matrix and enhanced the personalized ranking performance of MF models. However, MF models can be time-consuming in inferring recommendations. More specifically, although MF models can be trained efficiently with sparse preference matrices, the number of possible recommendations can be enormous when the numbers of users and items are massive. To tackle this problem, Duan et al. [102] proposed to separately compute dot-product results in each dimension so that some items can be discarded if their dot-product values are below a threshold for specific dimensions. However, separately processing different dimensions and discarding certain entries not only lead to inaccuracy, but also give up the opportunity to take advantage of low-level runtime optimization like BLAS [33] as shown in our experiments. Moreover, this approach does not reduce the number of possible recommendations. On the other hand, although some of the previous studies [189, 252] achieve acceleration by group recommendation, users in a certain group would receive identical recommendations that can be unsatisfactory for individual users.

2.5 Maximum Inner Product Search

Maximum inner product search (MIPS) can be treated as a closely related problem to MF based top- K recommender systems. Shrivastava and Li [295] and Neyshabur and Srebro [249] proposed to reduce MIPS to nearest neighbor search (NNS) and then solve NNS by Locality Sensitive Hashing (LSH) [167]. PCA tree [299] partitions the space according to the directions of principal components and shows better performance in practice. Bachrach et al. [20] showed tree-based approaches can be used for solving MIPS but the performance is poor for high dimensional data. Malkov et al. [225], Malkov and Yashunin [226] recently developed an NNS algorithm based on small world graph. Zhang et al. [366] applied the MIPS-to-NNS reduction and showed that graph-based approach performs well on neural language model prediction. Some algorithms were proposed to directly tackle MIPS problem instead of transforming to NNS. For example, Yu et al. [359] proposed Greedy-MIPS and showed a significant improvement over LSH and tree-based approaches. Another branch of research exploited sampling techniques with guaranteed approximation precision. Liu et al. [221] applied a bandit framework to iteratively query each dimension of the item vector; Ding et al. [96] proposed a 2-stage entry-wise sampling scheme and constructed an alias table to accelerate the sampling process. Despite having theoretical guarantee of approximation precision, in practice these methods suffer from slow entry-wise computation and the speedup is thus limited or even worse than the naive computation. Among all previous works, learning to screen (L2S) proposed by Chen et al. [61] is most similar to our method. L2S also leverages the clustering architecture to accelerate MIPS computation of multiple NLP tasks. However, L2S takes a long preparation time as it finds the clustering by end-to-end training and constructs a reduced search space by naive computation. In addition, L2S does not use representative vectors which differs from our proposed method. In our experiments, hierarchical graphical models [366], Greedy-MIPS [359] and L2S [61] are selected as the state-of-the-art MIPS methods for comparison.

2.6 k -mer Signiture Profiling

Existing k -mer counters index the reads into a compact and searchable structure, such as a hash table, a burst trie, or a compact suffix array. The occurrences of a specific k -mer can be retrieved by querying these data structures. Most of these counters are designed to process reads with fixed-size k -mers; several of them restrict the choice of k to fall within a threshold. These algorithms can be adapted to count k -mers of different sizes by repeating the process with different k 's. Here, we discuss these approaches.

Thread-Safe Shared Memory Hashing. *Jellyfish* [228] exploits the CAS (compare-and-swap) assembly instruction to update a memory location in a multi-threaded environment, and uses the “quotienting technique” and bit-packed data structure to reduce wasted memory. *Squeakr* [259] builds an off-the-shelf data structure based on counting quotient filter (CQF). It maintains both global and local CQFs to facilitate updates of each thread.

Disk-Based Hashing. Disk-based hashing reduces memory with complementary disk space. In general, this approach splits k -mers into bins, and stores them in files. Each bin is then loaded into the memory for counting. *DSK* [17] divides k -mers using a specific hash function based on the targeted memory and disk space. *MSPKmerCounter (MSPKC)* [216] proposes Minimum Substring Partitioning to reduce the memory usage of storing k -mers. Observing the fact that consecutive k -mers in a read often share a shorter substring, these consecutive k -mers can be compressed and stored in one bin. *KMC* [90], *KMC2* [91], and *KMC3* [202] are serial developments of parallel counters. These methods scan reads one block at a time and use a number of splitter threads to process the blocks. *KMC2* leverages the concept of minimizer to further reduce disk usage. *KMC3* accelerates the running time and optimizes the memory by taking a larger part of input data and better balancing the bin sizes.

Probabilistic Hashing. Probabilistic hashing approaches avoid counting the k -mers that are likely to arise from sequencing errors. *BFCOUNTER* [236] uses Bloom filter to identify all k -mers that are present more frequently than a threshold with a low false positive rate.

khmer [367] uses a streaming-based probabilistic data structure, CountMin Sketch [78].

Suffix-Arrays. Suffix-arrays present the potential of searching arbitrary k -mers without any restriction of k on a single scan. However, constructing a suffix-array on read data can be computationally expensive. *Tallymer* [205] is tailored to detect *de novo* repetitive elements ranging from 10 to 500bp in the genome. The algorithm first constructs an enhanced suffix array (`gt suffixerator`), and indexes k -mers one k at a time. *MSBWT* [156] compresses raw reads via a multi-string variant of Burrows-Wheeler Transform (BWT). Instead of concatenating all reads and sorting, it builds a BWT on each string and merges these multi-string BWTs through a small interleave array. The final structure allows a fast query of k -mers of arbitrary k .

Burst Tries. *KCMBT* [227] uses a cache efficient burst trie to store compact k -mers. The trie structure stores k -mers that share the same prefix in the same container. When a container is full, k -mers are sorted and burst. A good balance between the container size and the tree depth is essential to avoid constant sorting and bursting. As a result, *KCMBT* uses hundreds of trees.

2.7 Modeling Users behind Shared Accounts

Several studies have attempted to *model* user behaviors from session logs [4, 24, 323, 337, 355, 373]. They improve the performance of item recommendation according to the (latent) preferences of individual users. Diverse types of items have been investigated, including TV [4, 337, 355], movie [323, 363], and flight ticket [373]. The common approach to model user preferences is to de-convolute a high dimensional feature space that characterizes the relationships among accounts, items, and time [337, 355]. Techniques, such as subspace clustering [363], graph partition [337], collaborative filtering [323], topic model [373], and latent factor model with LDA [24], are used to obtain latent features so that the user preferences can be captured. Although the performance of the recommender systems can be improved, these studies assume each account is associated with one user and thus do not distinguish

individual users sharing an account. We argue that identifying individual users can bring additional values, as it allows for recovering lost revenue, better targeted marketing, designing new service plans, among many other useful applications. In addition, a sequence of items from a user can be regarded as a Markov process, so modeling interleaved Markov processes [207] can be also treated as a related work for modeling user preferences in shared accounts.

To the best of our knowledge, Zhang et al. [363] is the first and only attempt that can report whether an account is shared by multiple users and explicitly identify these users. They focus on item ratings and show how conventional methods such as expectation maximization and principal component analysis can be used for user identification via a specialized subspace clustering.

Fraud detection can be treated as a special case of user identification. Previous work detects malicious users based on context information such as social network structures [8, 51] and unusual behavior patterns [157, 181]. However, all of them identify the only one user in an account and cannot deal with the situations with multiple users.

2.8 Sponsorship Disclosure in Influencer Marketing

As influencer marketing has become a popular advertising method in recent years [26, 192, 223], several previous studies show the effect of disclosing sponsorship. Evans et al. [110] find that sponsorship disclosure helps audiences recognize paid partnerships but lowers purchase intention. Stubb and Colliander [302] find that impartiality disclosure, e.g., adding “This is not sponsored post”, helps generate high influencer credibility. Moreover, Evans et al. [111] investigate the effects of sponsorship text disclosure and sponsor pre-roll video advertising on YouTube. They find that the sponsor pre-roll advertising help audiences to understand sponsorship transparency. Yang et al. [354] reveal that distinct characteristics of sponsored posts, e.g., less number of usertags, longer caption than non-sponsored posts, that help exclusive promotion in advertising posts. Wojdyski et al. [342] present a metric to measure sponsorship transparency based on consumers’ perceptions.

2.9 Query Suggestion and Reformulation

To understand search intents behind queries, the context information including previous queries and click-through data is usually employed for query suggestion. Most of the existing studies rely on query association and query similarity in the search session. For example, association rules [115] and co-occurrence [116, 160] can be mined and calculated for query suggestion. The connections between consecutive queries can be also learned by a query-flow graph [37] or Markov models [50, 148]. The cosine similarity [29, 160] and the edit distance [58] are popular metrics to recommend queries that are similar to the context. To deal with the problems of data sparsity, some works attempt to cluster queries into denser groups. For instance, a bipartite graph based on click-through data can be built for discovering queries with similar concepts [49, 218, 234, 347]. The word distributions of queries can be also utilized for EM clustering [136]. In addition to clustering, machine learning frameworks with statistical features can also partially alleviate the sparsity problem [257, 287, 294]. To learn how users reformulate queries, Jiang et al. [174] model syntactic reformulations based on predefined reformulation strategies. Well-established ontologies can also be leveraged to learn semantic reformulations [172]. Recently, Sordoni et al. [298] propose to suggest queries with a hierarchical RNN as the first study of query suggestion with deep learning. Dehghani et al. [88] then improve the approach by decomposing the generation process into two reformulation strategies. Wu et al. [345] take the implicit user feedback into account to better rank queries for suggestion.

Query reformulation is the process that users refine the preceding queries in order to obtain more satisfactory search results. Previous studies focus on determining and predicting reformulation strategies [86, 161]. These reformulation strategies can be analyzed in two aspects, including *syntactic* and *semantic* reformulations. The syntactic reformulations are the changes of terms between queries, such as adding and removing terms [38]. The semantic reformulations address the changes of topics behind queries, such as generalization and specialization of the concepts [6]. Most of the works attempt to manually define reformulation strategies based on the above two aspects. For example, Boldi et al. [37] design four prede-

finer strategies of query transition while Huang and Efthimiadis [161] classify reformulations into 15 different types.

In addition to query suggestion [174], understanding reformulations is also beneficial to many other applications. Lee et al. [211] determine the term effectiveness for improving the search quality. Based on reformulations in a search session, the search results can be further personalized [171]. Ren et al. [276] leverage the concept of query reformulation to understand conversation logs. All of these works demonstrate not only the effectiveness but also the robustness of learning reformulations.

2.10 Conversation Disentanglement

Methods for conversation disentanglement can be simply categorized into unsupervised and supervised approaches. Unsupervised approaches [333] estimate the relationship between messages through unsupervised similarity functions such as cosine similarity, and assign messages to conversations based on a predefined threshold. In contrast, supervised methods exploit a set of user annotations [101, 106, 230, 233, 290] to adapt to different datasets.

In addition to conversations, some studies predict the partial structure of threaded data, especially for online forums [18, 328, 332]. These studies merely classify parent-child relationships in disentangled, independent threads. Moreover, they focus only on comments to the same post. Indeed, conversation disentanglement is a more difficult task.

Estimating the similarity of text pairs is an essential part in our approach. Many studies also focus on similar tasks aside from conversation disentanglement, such as entailment prediction [245, 334] and question-answering [11, 289, 357].

CHAPTER 3

Long Document Modeling for Information Retrieval

Languages are the most widely utilized media for both communication and documentation. Hence, text data are inherently one of the most popular and common data types in real-world applications. In this chapter, we present a novel deep learning approach to simultaneously enhance the effectiveness and efficiency in modeling text data by leveraging the structural semantics in text documents. Moreover, we conduct extensive experiments to demonstrate the improvements over state-of-the-art methods on the tasks of ad-hoc and few-shot document retrieval with multiple benchmark datasets.

3.1 Introduction

Pre-trained Transformers such as BERT [95] effectively transfer language understanding to better relevance estimation in many search ranking tasks [250, 251, 353]. Nevertheless, the effectiveness comes at the quadratic cost $O(n^2)$ in computing complexity corresponds to the text length n , prohibiting its direct application to long documents. Prior work adopts quick workarounds such as document truncation or splitting-and-pooling to retrofit the document ranking task to pretrained transformers. Whilst there have been successes with careful architecture design, those bandit-solutions inevitably introduce information loss and create complicated system pipelines.

Intuitively, effective document ranking does not require fully connected self-attention between all query and document terms. The relevance matching between queries and documents often takes place at text segments as opposed to individual tokens [48, 178], suggesting that a document term may not need information thousands of words away [65, 238], and

that not all document terms are useful to calculate the relevance to the query [350]. The fully connected attention matrix includes many unlikely connections that create efficiency debt in computing, inference time, parameter size, and training convergence.

We present Query-Directed Sparse Transformer (QDS-Transformer) for long document ranking. In contrast to retrofitted solutions, QDS-Transformer fundamentally considers the desirable properties for assessing relevance by focusing on attention paths that matter. Using sparse local attention [65], our model removes unnecessary connections between distant document tokens. Using global attention upon sentence boundaries, our model further incorporates the hierarchical structures within documents. Last but not the least, we use global attention on all query terms that direct the focus to the relevance matches between query-document term pairs. These three attention patterns in our Query-Directed Sparse attention, as illustrated in Figure 3.1, permit global dissemination of IR-axiomatic information while keeping computation compact and essential.

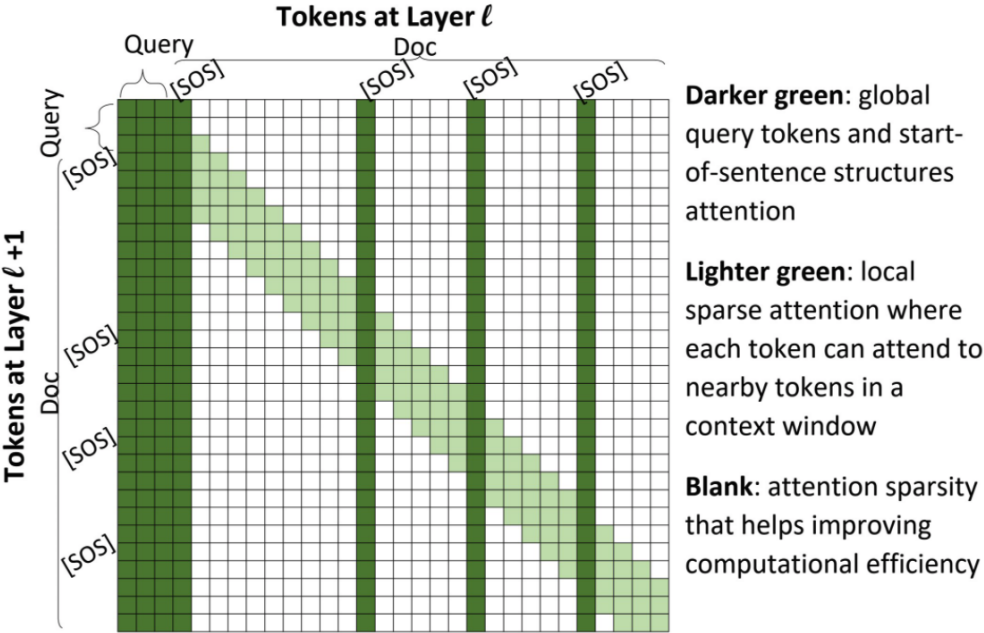


Figure 3.1: An example illustration of the attention mechanism used in Query-Directed Sparse Transformer.

In our experiments with TREC Deep Learning Track [80] and three more few-shot document ranking benchmarks [365], QDS-Transformer consistently improves the standard

retrofitting BERT ranking baselines (e.g., max-pooling on paragraphs) by 5% NDCG. It also shows gains over more recent transformer architectures that induces various sparse structures, including Sparse Transformer, Longformer, and Transformer-XH, as they were not designed to incorporate the essential information required in document ranking. In the meantime, we also thoroughly quantify the efficiency improvement from our query-directed sparsity, showing that with TVM support [62], different sparse attention patterns lead to variant training and inference speed up, and in general QDS-Transformer enjoys 200%+ speed up compared to vanilla BERT on long documents.

Our visualization also shows interesting learned attention patterns in QDS-Transformer. Similar to the observation on BERT in NLP pipeline [316], in lower QDS-Transformer levels, the attention focuses more on learning the local interactions and document hierarchies, while in higher layers the model focuses more on relevance matching with the query terms. We also show examples that QDS attention may center on the sole sentence that directly answers the query, or may span across several sentences that cover different aspects of the query, depending on the scope of the intent; this brings the advantage of better interpretability based on sparse attention.

3.2 Preliminaries on Document Ranking

Given a query q and a set of candidate documents $D = \{d\}$, the document ranking task is to produce the ranking score $f(q, d)$ for each candidate document based on their relevance to the query.

BERT Ranker. The standard way to leverage pretrained BERT in document ranking is to concatenate the query and the document into one text sequence, feed it into BERT layers, and then use a linear layer on top of the last layer’s [CLS] token [250]:

$$f(q, d) = \text{Linear}(\text{BERT}([\text{CLS}] \circ q \circ [\text{SEP}] \circ d)).$$

This BERT ranker can be fine-tuned using relevance labels on (q, d) pairs, as simple as a classification task, and has achieved strong performances in various text ranking benchmarks [25, 80].

Transformer Layer. More specifically, let $\{t_0, t_1, \dots, t_i, \dots, t_n\}$ be the tokens in the concatenated q - d sequence, with query tokens $t_{1:|q|} \in q$ and document tokens $t_{|q|+1:n} \in |d|$, considering special tokens being part of q or d . The l -th transformer layer in BERT takes the hidden representations of previous layer (H^{l-1}), which is embedding for $l = 1$, and produces a new H^l as follows [321].

$$H^l = W^F(\hat{H}^l), \quad (3.1)$$

$$\hat{H}^l = A \cdot M \cdot V^T, \quad (3.2)$$

$$A = \mathbf{1}, \quad (3.3)$$

$$M = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right), \quad (3.4)$$

$$(Q^T; K^T; V^T) = (W^q; W^k; W^v) \cdot H^{l-1}. \quad (3.5)$$

It first passes the previous representations through the self-attention mechanism, using three projections (Eqn. 3.5), and then calculates the attention matrix between all token pairs using their query-key similarity (Eqn. 3.4, as in single-head formation). The attention matrix M then is used to fuse all other tokens' representation V , to obtain the updated representation for each position (Eqn. 3.2). In the end, another feed-forward layer is used to obtain the final representation of this layer H^l (Eqn. 3.1).

The matrix A is the n^2 ‘‘adjacency’’ matrix in which each entry is one if there is an attention path between corresponding positions: $A_{ij} = 1$ means t_i queries the value of t_j using the key of t_j . In standard transformer and BERT, the attention paths are fully connected thus $A = \mathbf{1}$.

Computation Complexity. In each of the BERT layers, all the feed-forward operations (Eqn. 3.1 and 3.5) are applied to each individual token, leading to linear complexity w.r.t. text length n and the square of the hidden dimension size dim . The self-attention operation

in Eqn. 3.2 and 3.4 calculates the attention strengths upon all token pairs, leading to squared complexity w.r.t text length but linear of the hidden dimension size.

The complexity of one transformer layer in BERT thus includes two components:

$$\underbrace{\mathcal{O}(\text{dim}^2 n)}_{\text{Feedforward}} + \underbrace{\mathcal{O}(n^2 \text{dim})}_{\text{Self-Attention}}. \quad (3.6)$$

The hidden dimension size (dim) is 768 in BERT Base and 1024 in BERT Large [95]. When the text sequence is longer than 1000 or 2000 tokens, which is often the case in document ranking [80], the self-attention part becomes the main bottleneck in both computation and GPU memory. This leads to various retrofitted solutions that adapted the document ranking tasks to standard BERT which takes at most 512 tokens per sequence [83, 251, 351, 353].

3.3 QDS-Transformer

Recent research has shown that with sufficient training and fully-connected self-attention, BERT learns attention patterns that capture meaningful structures in language [71] or for specific tasks [372]. However, this is not yet the case in long document ranking as computing becomes the bottleneck.

This section first presents how we overcome this bottleneck by injecting IR-specific inductive bias as sparse attention patterns. Then we discuss the efficient implementation of sparse attention.

3.3.1 Query-Directed Sparse Attention

Mathematically, inducing sparsity in self-attention is to modify the attention adjacency matrix A by only keeping connections that are meaningful for the task. For document retrieval, we include two groups of informative connections as sparse adjacency matrices: *local attention* and *query-directed global attention*.

3.3.1.1 Local Attention

Intuitively, it is unlikely that a token needs to see another token thousands of positions away to learn its contextual representation, especially in the lower transformer layers which are more about syntactic and less about long-range dependencies [316]. We follow this intuition used in the Sparse Transformer [65] and define the following local attention paths:

$$A_{\text{local}}[i, j] = 1, \text{ iff } |i - j| \leq w/2. \quad (3.7)$$

It only allows a token to see another token in each transformer layer if the two are $w/2$ position away, with w the window size. The local attention serves as the backbone for many sparse transformer variations as it provides the basic local contextual information [31, 79, 304].

3.3.1.2 Query-Directed Global Attention

The local attention itself does not fully capture the relevance matches between the query and documents. We introduce several query-directed attention patterns to incorporate inductive biases widely used in document representation and ranking.

Hierarchical Document Structures. A common intuition in document representation is to leverage the hierarchical structures within documents, for example, words, sentences, paragraphs, and sections, and compose them into hierarchical attention networks [356]. We use a two-level word-sentence-document hierarchy and inject this hierarchical structure by adding fully connected attention paths to all the sentences.

Specifically, we first prepend a special token [SOS] (start-of-sentence) to each sentence in the document, and form the following attention connections:

$$A_{\text{sent}}[i, j] = 1, \text{ iff } t_j = [\text{SOS}]. \quad (3.8)$$

Matching with the Query. For retrieval tasks, arguably the most important principle is

to capture the semantic matching between queries and documents. Inducing this information is as simple as adding dedicated attention paths on query terms:

$$A_{\text{query}}[i, j] = 1, \text{ iff } t_i \in q. \quad (3.9)$$

It allows each token to see all query terms so as to learn query-dependent representations.

3.3.2 Summary

The three attention patterns together form the query-directed attention in QDS-Transformer:

$$A_{\text{QDS}} = A_{\text{local}} \cup A_{\text{sent}} \cup A_{\text{query}} \cup A_{[\text{CLS}]}. \quad (3.10)$$

We also add the global attention between all other tokens and [CLS]. Keeping everything else standard in BERT and using this query-directed sparse attention (A_{QDS}) in place of the fully-connected self-attention (A), we obtain our QDS-Transformer architecture as illustrated in Figure 3.2.

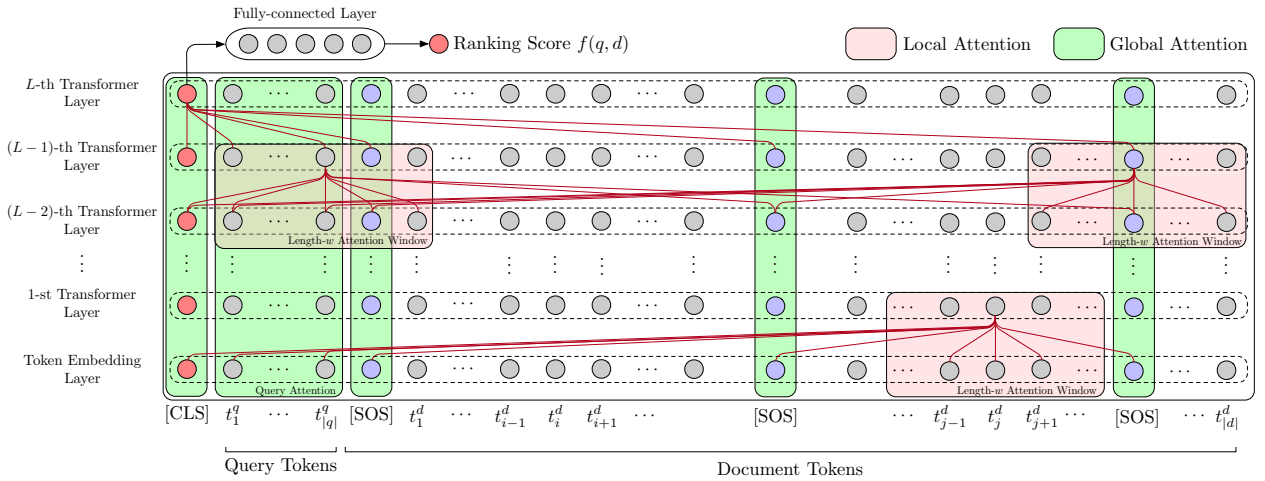


Figure 3.2: The overall schema of our proposed QDS-Transformer.

Interestingly, QDS-Transformer also resembles various effective IR-Axioms developed in past decades. For example, in QDS attention, a query term mainly focuses on the [SOS] token through A_{Sent} , while the [SOS] token recaps the proximity [48] matches locally around

it through A_{Local} . The local attention in the query part also resembles the effective phrase matches [238] as the query term representations are contextualized using other query terms through A_{Local} .

3.3.3 Efficient Sparsity Implementation

Our query-directed sparse attention reduces the self-attention complexity from $\mathcal{O}(n^2 \text{dim})$ to $\mathcal{O}(n \cdot \text{dim} \cdot (w + |q| + |s|))$, where the local window size w and query length $|q|$ are constant to document length, and the number of sentences is orders of magnitude smaller.

However, to implement this sparsity efficiently on GPU is not that straightforward. Naively using for-loops or masking the adjacency matrix A may result in even worse efficiency than the full self-attention in common deep learning frameworks. An efficient implementation of sparse operations often requires customized CUDA kernels, which are inconvenient and require expertise in low-level GPU operations [65]. Inspired by Longformer [31], we address this issue by implementing QDS-Transformer with Tensor Virtual Machine (TVM) [62]. Precisely, we implement custom CUDA kernels using TVM to dynamically compile our attention map A_{QDS} into efficiency-optimized CUDA codes.

3.4 Experimental Methodologies

This section discusses our experimental settings.

TREC 2019 Deep Learning Track Benchmark. We evaluate QDS-Transformer based on the document ranking task from this recent TREC benchmark [80], specifically using the reranking subtask to rerank top-100 BM25 retrieved documents. The official evaluation metric is NDCG@10 on the test set. We also report MAP on test and MRR@10 on the development set.

Few-shot Document Ranking Benchmarks. We then evaluate the generalization ability of QDS-Transformer in the few-shot setting [365] using TREC datasets Robust04 (RB04), ClueWeb09-B (CW09), and ClueWeb12-B13 (CW12), in which labels are much fewer than

	Ad-hoc	Few-shot (avg. over 5 folds)		
	TREC19 DL	RB04	CW09-B	CW12-B13
Train queries	367,013	150	120	60
Train qrels	384,597	186,846	28,278	17,343
Dev queries	5,193	50	40	20
Dev qrels	519,300	62,282	9,426	5,781
Test queries	43	50	40	20
Test qrels	16,258	62,282	9,426	5,781

Table 3.1: The statistics of the experimental datasets.

DL track. Our experimental settings are consistent with prior work [365] in using the “MS MARCO Human Labels”. Specifically, neural rankers trained with MARCO labels are used as feature extractors to enrich TREC documents, which are then tested with five-fold cross-validation [84].

Table 3.1 summarizes the statistics of four datasets. We describe more details about datasets and experimental settings in Appendix A.1.1.

Baselines. Our baselines include multiple neural IR models and the best official TREC runs of single models. The main baselines cover:

- Relying on BERT models, RoBERTa (FirstP) only considers the first paragraph, while RoBERTa (MaxP) encodes short paragraphs with BERT and combines them with a max-pooling layer [83].
- Transformer-XH [372] retrofits data pipelines to create independent sentences which are fed into BERT models, and aggregates them with an extra-hop attention layer.
- TK [154] and TKL [155] apply BERT-based kernels to estimate the relevance over document tokens with full attention.
- Sparse-Transformer [65] applies length- w sparse local attention windows without considering query tokens.
- Longformer also uses sparse local attention and adds global attention by prepending one special token respectively to the query and document, same as in their [31] QA setup.

For ad-hoc retrieval, we also consider CO-PACRR [163] which employs CNNs without using pretrained NLM (non-PLM). Note that IDST [351] is not comparable because it exploits

external generators for document expansion. For the few-shot learning task, we additionally compare with SDM, RankSVM, Coor-Ascent, and Conv-KNRM as reported in previous studies [84, 350]. More details of the baselines can be found in Appendix A.2.

Implementation Details. We implement all methods with PyTorch [260] and the Hugging Face transformer library [343], excluding the baselines that have previously reported their scores. For sparse attention, we implement it using TVM with a custom CUDA kernel in PyTorch [62]. Models are optimized by the Adam optimizer [196] with a learning rate 10^{-5} , $(\beta_1, \beta_2) = (0.9, 0.999)$, and a dropout rate 0.1. The dev set is used for hyperparameter tuning to decide the best model, which is then applied to the test set. We set the maximum length of input sequences as 2,048. The dimension of the dense layer $\mathcal{F}_{\text{dense}}(\cdot)$ in relevance estimation is 768, while the local attention window size w is 128. All experiments are conducted on an Nvidia DGX-1 server with 512 GB memory and 8 Tesla V100 GPUs. Each method is limited to access only one GPU for fair comparisons.

3.5 Evaluation Results

This section evaluates QDS-Transformer in its effectiveness, attention patterns, and efficiency. We also analyze the learned query-directed attention weights and show case studies.

3.5.1 Retrieval Effectiveness

Table 3.2 summarizes the retrieval effectiveness on the TREC-19 DL benchmark. Table 3.3 shows the few-shot performance on the three TREC datasets.

QDS-Transformer consistently outperforms baseline methods on all datasets in both experimental settings. Note that the higher MAP scores from some methods in TREC-19 DL is because they have better first stage retrieval and are not using the same reranking setting. QDS-Transformer outperforms the best BERT-based TREC run by 3.25% in NDCG@10 and is more effective than the concurrent sliding window approach, TKL. Moreover, QDS-Transformer outperforms RoBERTa (MaxP), which is the standard retrofitted method for

TREC Deep Learning Track Document Ranking			
Method	Test Set		Dev Set
	NDCG@10	MAP	MRR@10
BM25	0.488	0.234	0.252
TREC Best Models			
BM25 (bm25tuned_prf)	0.528	0.386	0.318
Trad (srchvrs_run1)	0.561	0.349	0.306
Non-PLM (TUW19-d3-re)	0.644	0.271	0.401
BERT (bm25exp_marcomb)	0.646	0.424	0.352
Baseline Models			
CO-PACRR	0.550	0.231	0.284
TK	0.594	0.252	0.312
TKL	0.644	0.277	0.329
RoBERTa (FirstP)	0.588	0.233	0.278
RoBERTa (MaxP)	0.630	0.246	0.320
Sparse Attention based Models			
Sparse-Transformer	0.634	0.257	0.328
Longformer-QA	0.627	0.255	0.326
Transformer-XH	0.646	0.256	0.347
QDS-Transformer	0.667	0.278	0.360

Table 3.2: The ad-hoc retrieval performance of our approach and baseline methods on the TREC-19 DL track benchmark. Note that those baselines with higher MAP scores are all full retrieval and benefited from additional data engineering like query expansion.

BERT, by 6% in NDCG@10 while also being a unified framework.

Compared with Sparse Transformers and Longformer-QA, QDS-Transformer provides more than 5% improvement in nearly all datasets. The best baseline is Transformer-XH, which creates structural sparsity by breaking a document into segments and introduces effective eXtra-hop attentions to jointly model the relevance of those segments. While these methods show competitive effectiveness especially with our TVM implementation, QDS-Transformer is consistently more accurate through the query-directed sparse attention patterns in all evaluation settings.

Method	RB04		CW09		CW12	
	NDCG	ERR	NDCG	ERR	NDCG	ERR
Classical IR; Cross Validated						
SDM	0.427	0.117	0.277	0.138	0.108	0.091
RankSVM	0.420	n.a.	0.289	n.a.	0.121	0.092
Coor-Ascent	0.427	n.a.	0.295	n.a.	0.121	0.095
Neural IR; Trained on MS MARCO and then Cross Validated.						
Conv-KNRM	0.427	0.117	0.287	0.160	0.112	0.092
RoBERTa (FirstP)	0.437	0.110	0.262	0.161	0.111	0.086
RoBERTa (MaxP)	0.439	0.114	0.264	0.162	0.092	0.074
Sparse-Transformer	0.449	0.119	0.274	0.173	0.119	0.094
Longformer-QA	0.448	0.113	0.276	0.179	0.111	0.085
Transformer-XH	0.450	0.123	0.283	0.179	0.107	0.080
QDS-Transformer	0.457	0.126	0.308	0.191	0.131	0.112

Table 3.3: The few-shot learning retrieval performance of different methods on three benchmark datasets. NDCG and ERR are at cut-off 20.

Method	Attention		TREC-19 DL Track	
	Q	Sent	NDCG@10	MAP
RoBERTa (MaxP)	✓	✗	0.630	0.246
Sparse Transformer	✗	✗	0.634	0.257
LongFormer-QA	✗	✗	0.627	0.255
Transformer-XH	✓	✓	0.646	0.256
QDS-Transformer (S)	✗	✓	0.633	0.244
QDS-Transformer (Q)	✓	✗	0.658	0.263
QDS-Transformer	✓	✓	0.667	0.278

Table 3.4: The retrieval performance of different models on the TREC-19 DL track benchmark dataset with different global attention patterns. Q and S indicate the usage of query and sentence global attention. Note that QDS-Transformer with no global attention is equivalent to Sparse-Transformer.

3.5.2 Effectiveness of Attention Patterns

This experiment studies the contribution of our query-directed sparse attention patterns to QDS-Transformer’s effectiveness.

Table 3.4 shows the ablation results of the three attention patterns in TREC-19 DL benchmark: local attention only (A_{local} , Sparse Transformer), hierarchical attention on sentence only (A_{sent} , QDS-Transformer (S)), and query-oriented attention only (A_{query} , QDS-

Transformer (Q)). All three sparse attention patterns contribute. As expected, query-oriented attention is most effective to capture the relevance match between query and documents. Note that the RoBERTa (MaxP) and Transformer-XH also attend to queries, but the attention is more localized as the document is broke into separated text pieces and the query is concatenated with each of them. In comparison, QDS-Transformer mimics the proximity matches and captures the global hierarchical structures in the document using dedicated attention from query terms to sentences.

Figure 3.3 depicts the change in retrieval effectiveness by varying the local attention window size. Both NDCG@10 and MAP@10 grow at a steady pace starting from a window size of 32 and peak at 128, but no additional gain is observed with bigger window sizes. The information from a term 512 tokens away does not provide many signals in relevance matching and is safely pruned in QDS-Transformer. Note that the dip at attention size 1024 is because our model is initialized from RoBERTa which is only pretrained on 512 tokens.

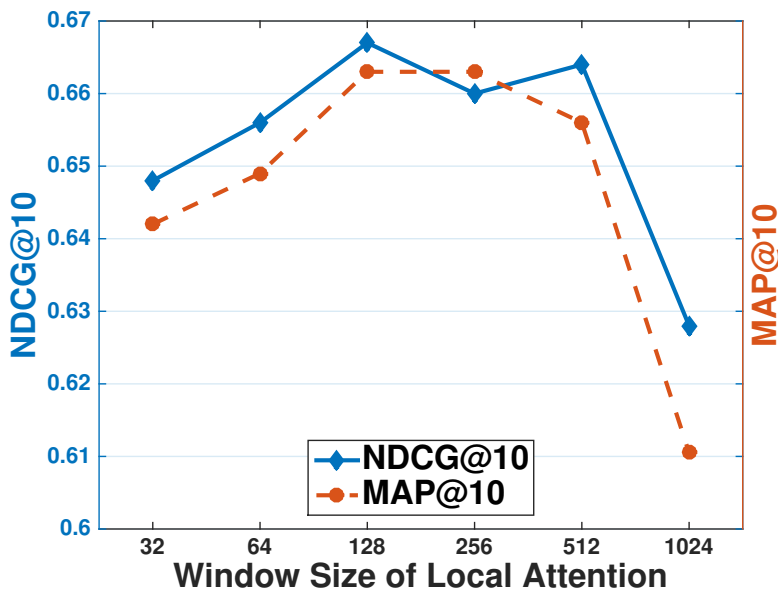


Figure 3.3: The performance of QDS-Transformer on TREC-19 DL track dataset with different local attention window sizes w .

Method	Length	Sparsity	ms per q-d	
			Train	Infer
RoBERTa	1024	100%	391	100
RoBERTa	2048	100%	799	205
RoBERTa (FirstP)	512	100%	138	17
RoBERTa (MaxP)	4*512	25%	305	55
Transformer-XH	4*512	25%	309	54
QDS-Transformer (128)	512	30.84%	218	45
QDS-Transformer (128)	1024	18.72%	249	52
QDS-Transformer (128)	2048	8.97%	321	92
Longformer-QA (128)	2048	4.70%	166	45
Sparse-Transformer (128)	2048	4.56%	154	40
QDS-Transformer (32)	2048	6.70%	201	50
QDS-Transformer (64)	2048	8.97%	309	86
QDS-Transformer (128)	2048	13.53%	321	92
QDS-Transformer (256)	2048	22.64%	475	127
QDS-Transformer (512)	2048	40.88%	512	160
QDS-Transformer (1024)	2048	77.34%	629	195
QDS-Transformer (Q)	2048	5.10%	316	108
QDS-Transformer (S)	2048	8.57%	322	105
Without TVM Implementation				
Sparse-Transformer (128)	2048	4.56%	251	62
QDS-Transformer (128)	2048	13.53%	390	103

Table 3.5: Efficiency Quantification. The local attention window size is shown in parentheses. Q and S indicate the usage of only query and sentence attention. Sparsity is compared with fully attention at same text length.

3.5.3 Model Efficiency

This experiment benchmarks the efficiency of different sparse attention patterns. Their training and inference time (ms per query-document pair, or MSpP) is shown in Table 3.5.

RoBERTa on 2048 tokens is prohibitive; we only measured its time with random parameters as we were not able to actually train it. Retrofitting was a natural choice to leverage pretrained models.

Sparsity helps. Sparse-Transformer (128) is much faster than MaxP. Interestingly, its attention matrix with only 4.56% non-zero entries leads to on par efficiency with retrofitted solutions and also only 5 times faster compared to full attention; this is due to the required

cost involved in feed-forward. This effect is also reflected in the efficiency of QDS-Transformer with different local window sizes.

Different sparsity patterns dramatically influence the optimization of TVM. Intuitively, patterns with more regular shape would be easier to optimize than more customized connections in TVM. For example, the skipping patterns along sentence boundary in QDS-Transformer (S) seems more forgiving than the query-oriented attentions (Q). Comparing efficiency with and without our TVM implementation, the diagonal sparse shape in Sparse-Transformer is much better optimized.

How to better utilize the advantage from sparsity and structural inductive biases is perhaps a necessary future research direction in an era where models with fewer than one billion parameters are no longer considered large [44]. Making progress in this direction may need more close collaborations between experts in application, modeling, and infrastructure.

3.5.4 Learned Attention Weights

This experiment analyzes the learned attention weights in QDS-Transformer, using the approach developed by Clark et al. [71].

Figure 3.4 illustrates the average maximum attention weights of the three attention patterns used in our model. Interestingly, the model tends to implicitly conduct hierarchical attention learning [356], where lower layers focus on learning structures and pay more attention to [SOS] tokens, while higher layers emphasize the relevance by attending to queries more. Attention on both types of tokens is consistently stronger than on the [CLS] token. The model is capturing the inductive biases emphasized by our sparse attention structures.

Figure 3.5 shows the average entropy of the attention weight distribution. Intuitively, lower layer attention tends to have high entropy and thus a very broad view over many words, to create contextualized representations. The entropy of query and [SOS] are in general lower, as they focus on capturing information needs and document structures. The entropy of all three types of tokens rises again in the last layer, implying that they may try

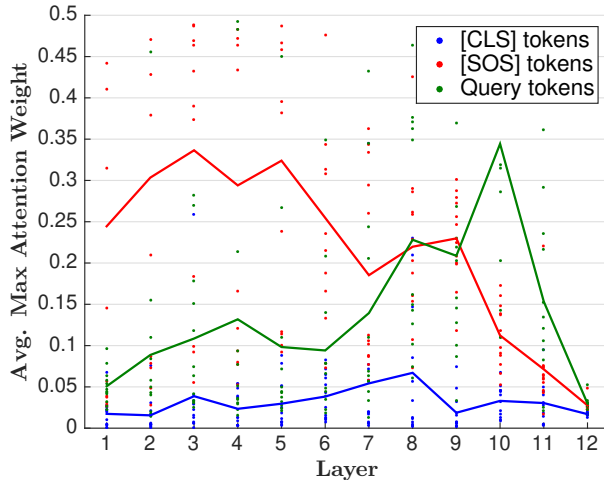


Figure 3.4: The average maximum attention scores to different types of tokens over Transformer layers.

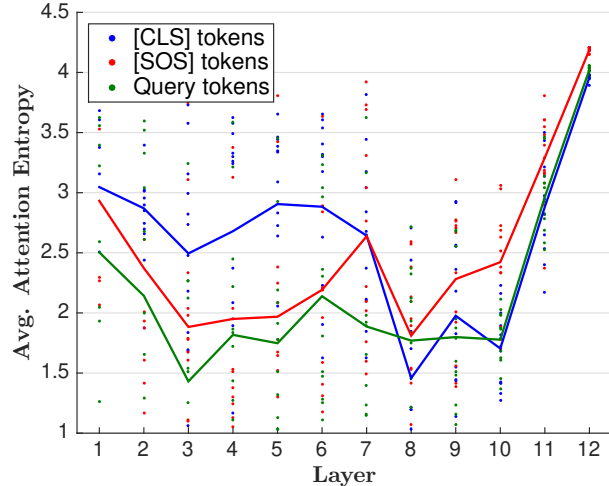


Figure 3.5: The average entropy scores of attention distributions for different token types over Transformer layers.

Q1: 1037798 (who is robert gray)	Q2: 1110199 (what is wifi vs bluetooth)
docid: D3533931	docid: D1325409
Heads 1,2,4,6,9,10,11,12:	Head 1: Bluetooth’s low power consumption make it useful where power is limited.
Robert Gray (title)	Head 2: Wi-Fi appliances are often plugged into wall outlets to operate.
Heads 3,5,7,8:	Head 7: The extremely low power requirements of the latest Bluetooth 4.0 standard allows wireless connectivity to be added to devices powered only by watch batteries.
Robert Gray, (born May 10, 1755, Tiverton, R.I. died summer 1806, at sea near eastern U.S. coast), captain of the first U.S. ship to circumnavigate the globe and explorer of the Columbia River.	Head 9: A Wi-Fi enabled network relies on a hub.
	Head 10: The advantages of using bluetooth from existing technology.
	Head 11: Wi-Fi is more suited to data-intensive activities such as streaming high-definition movies, while Bluetooth is better suited to tasks such as transferring keyboard strokes to a computer.
	Head 12: The greater power of Wi-Fi network also means it can move data more quickly than Bluetooth network.

Table 3.6: Case study of two queries on the sentences with the highest attention weights in the last transformer layer over different heads for the [CLS] token.

to aggregate representation for the whole input.

3.5.5 Case Study on Learned Attention Weights

Table 3.6 shows a case study of sentences with the highest attention weight from [CLS] in the last layer for two example queries. For factoid query Q1, all heads center on precise sentences that can directly answer the query. For Q2 that is on the exploratory side, different attention heads exhibit diverse patterns focusing on partial evidence that can provide a

Q3: 1112341 (what is the daily life of thai people)	
Query Token	Sentence with the highest attention weight in the document D1641978
life	Children are expected to show great respect for their parents, and they maintain close ties, even well into adulthood .
thai	Culture of Thailand (title)

Table 3.7: Case study of the query 1112341 on the sentences in the document D1641978 with the highest attention weights among all heads from two query tokens. Note that we use attention weights in the third transformer layer.

Sentence in the document D2944963 for Q4: 833860 (what is the most popular food in switzerland)	Top Query Token
Top 10 Swiss foods with recipes (title)	switzerland
You certainly won't go hungry in Switzerland.	food
You spear small cubes of bread onto long-stemmed forks and dip them into the hot cheese (taking care not to lose the bread in the fondue).	food
Jamie Oliver has this easy cheese fondue recipe, and this five-star recipe has good reviews.	popular

Table 3.8: Case study of the query 833860 with the query tokens with the highest attention weights in the 10-th transformer layer among all heads from the [SOS] tokens of sentences in the document D2944963.

broader understanding collectively.

Table 3.7 depicts the other case study on learned attention weights of sentences from query tokens. We adopt the third transformer layer, where sentences obtain more attention as shown in Figure 3.4, to emphasize significant sentences for query tokens. The results show query-directed attention can capture sentences with different topics matched to individual query tokens, thereby comprehending sophisticated document structure.

In addition to attention from the classification token [CLS] and query tokens as shown in Section 3.5.5, here we analyze the attention from sentences. Table 3.8 shows the query tokens with the highest attention weights in the 10-th transformer layer among all head from the [SOS] tokens of sentences. Note that the 10-th transformer layer indicates higher importance of query tokens as shown in Figure 3.4. The results show that QDS-Transformer is capable of directing sentences to the tokens with matched topics, thereby understanding sophisticated document structure with different topics.

These findings suggest that QDS-Transformer has an interesting potential to be applied to not only retrieval but also the question-answering task in NLP, providing a generic and

effective framework, while also being interpretable with its the sparse structural attention connectivity.

3.6 Conclusion

QDS-Transformer improves the efficiency and effectiveness of pretrained transformers in long document ranking using sparse attention structures. The sparsity is designed to capture the principal properties (IR-Axioms) that are crucial for relevance modeling: local contextualization, document structures, and query-focused matching. In four TREC document ranking tasks with variant settings, QDS-Transformer consistently outperforms competitive baselines that retrofit to BERT or use sparse attention not designed for document ranking.

Our experiments demonstrate the promising future of joint optimization of structural domain knowledge and efficiency from sparsity, while its current form is somewhat at the infancy stage. Our analyses also indicate the potential of better interpretability from sparse structures and more unified models for IR and QA.

CHAPTER 4

Sequence Modeling for Circular RNA Prediction

Similar to languages, many other types of human data are also in sequential forms. Although their semantics may not be as easy as texts to be directly interpreted, there still could have some structural information to implicitly indicate the semantics. In this chapter, we take genome sequence modeling as an example to leverage the exon structures to learn representations of human DNA reads for predicting the existence of circular RNAs. We also demonstrate how our proposed framework can provide practical insights for domain experts without computer science to discover unknown circular RNAs.

4.1 Introduction

The ENCODE project has revealed the vital role of different forms of nonprotein-coding RNAs. Among these types of RNAs, much attention has been placed on cataloging and studying the long non-coding RNAs (lncRNAs), due to their high relevancy to gene regulation and diseases [92, 267]. Long non-coding RNAs are typical of 200bp to > 100kbp in length [326]. As a particular type of lncRNA, endogenous circular RNA (circRNA) has recently received a tremendous amount of research highlights not only because of its circularity, but also its implications in a myriad of human diseases, such as cancer and Alzheimer’s disease [103, 270]. circRNA arises during the process of alternative splicing of protein-coding genes, where the 5’ end of an exon is covalently ligated to the 3’ end of the same exon or a downstream exon, forming a closed continuous loop structure. This mechanism is also known as “backsplicing.” The circular structure provides several beneficial properties over the linear RNAs. To be more specific, it can serve as templates for rolling circle amplification

of RNAs [41], rearrange the order of genetic information [208], resistant to exonuclease-mediated degradation [170], and create a constraint on RNA folding [208]. Although the consensus of biological functions, mechanisms, and biogenesis remains unclear for most circRNAs [30, 358], there are emerging evidence suggesting their roles in acting as sponges for microRNAs [144, 237], RNA-binding protein competition [15], and inducing host gene transcription [217]. Evidently, as a fundamental step to facilitate the exploration of circRNA, it is essential to have a high-throughput approach to identify the circRNAs.

Multiple factors can contribute to the formation of circRNAs. These factors include complementary sequences in flanking introns [168], the presence of inverted repeats [104], number of ALU and tandem repeats [170], and SNP density [317]. These factors, together with the evolutionary conservation and secondary structure of RNA molecules, have been considered as the discriminative features for circRNA identification. Several research efforts [60, 258, 329] have leveraged these features to train a conventional statistical learning model to distinguish circRNAs from other lncRNAs. These statistical learning algorithms include support vector machines (SVM), random forest (RF), and multi-kernel learning. However, methods along this line often require an extensive domain-specific feature engineering process. Moreover, the selected features may not provide sufficient coverage to characterize the backsplicing event.

Recently, the rising of deep learning architectures have been widely adopted as an alternative learning algorithm that can alleviate the inadequacy of conventional statistical learning methods. Specifically, these deep learning algorithms provide powerful functionality to process large-scale data and automatically extract useful features for object tasks [210]. In the domain of circRNA prediction, the convolution neural network (CNN) is the architecture that has been widely explored to automatically learn the critical features for prediction, either from the primary sequence [54, 330] or secondary structure [114]. Although CNN is capable of capturing relevant local patterns on gene sequences, positional information of the splice junctions and global context of each splice site cannot be recognized. One of these approaches [54] attempts to address this issue by applying recurrent neural networks (RNNs)

to learn sequential and contextual information; however, the essential knowledge, such as splice sites and junctions, are still ignored.

Understanding the properties of splice sites and their relationships is one of the keys to master RNA splicing and the formation of circular RNAs because the splicing event can be considered as interaction among those splice sites. To fathom the relations between splice sites, circDeep [54] explicitly analyzes the nucleotide sequences of two splice sites to predict the circular RNAs. DeepCirCode [330] utilizes CNNs to model the flanking regions around two splice sites to identify if there is a backsplice event. However, all of the existing methods fail in modeling deep interaction among splice sites for circular RNA prediction. For example, circDeep only measures shallow interaction among splice sites on the nucleotide level; DeepCirCode is limited to examine only a single pair of splice sites and lacks the capacity of modeling more complex relations among splice sites on multi-isoform genes. Hence, there is an immense gap to comprehensively understand the relationship between splice sites and their interaction regarding the formation of circular RNAs.

In this work, we propose the framework of Junction Encoder with Deep Interaction (JEDI) to address the limitations in circular RNA prediction. More precisely, we focus on predicting the existence of circular RNAs from either the reference gene/isoform sequences or assembled transcript sequences by modeling splice sites and their deep interaction with deep learning techniques. First, the attentive junction encoders are presented to derive continuous embedding vectors for acceptor and donor splice sites based on their flanking regions around junctions. Based on the acceptor and donor embeddings, we propose the novel cross-attention layer to model deep interaction between acceptor and donor sites, thereby inferring cross-attentive embedding vectors. Finally, the attention mechanism is applied to determine acceptors and donors that are more important than other ones to predict if there is a circRNA. It is also important to note that the interpretability of the attention mechanism and the cross-attention layer enables JEDI to automatically discover backsplicing without training on any annotated backspliced sites.

Our contributions are three-fold. First, to the best of our knowledge, this work is the

first study to model the deep interaction among splice sites for circular RNA prediction. The more profound understandings of the relationships among splice sites can intuitively benefit circular RNA prediction in implying backsplicing. Second, we propose a robust and effective end-to-end framework, JEDI, to deal with both isoform-level and gene-level circular RNA prediction based on the attention mechanism and the innovative cross-attention layer. More specifically, JEDI is capable of not only deriving appropriate representations from junction encoders but also routing the importance of forming circular RNAs on different levels. Third, JEDI creates a new opportunity of transferring the knowledge from circular RNA prediction to backsplicing discovery based on its extensive usage of attention mechanisms. Moreover, our approach can be utilized as a general and user-friendly detection tool to provide a robust estimated ranking for further validation. Extensive experiments on human circRNAs have demonstrated that JEDI significantly outperforms eight competitive baseline methods on both isoform-level and gene-level. The independent study on mouse circRNAs also indicates that JEDI is robust to transfer knowledge learned from human sequence to mouse for circRNA prediction. This phenomenon is supported by the observation of highly conserved circular RNA across species [30, 170, 303]. In addition, we conduct the experiments to demonstrate that JEDI can automatically discover backspliced site pairs without any further annotations. Finally, an in-depth analysis of model hyper-parameters and run-time presents the robustness and efficiency of JEDI.

4.2 Materials and Methods

In this section, we first formally define our objective, and then present our proposed deep learning framework, Junction Encoder with Deep Interaction (JEDI), to predict circRNAs.

4.2.1 Preliminary and Problem Statement

The vocabulary of four nucleotides is denoted as $\mathcal{V} = \{A, C, G, T\}$. For a gene sequence S , $s[i \dots j] \in \mathcal{V}^{j-i+1}$ indicates the subsequence from the i -th to the j -th nucleotide of a sequence

S . For a gene or an RNA isoform with the sequence S , $\mathcal{E}(S) = \{(a_i, d_i)\}$ represents the given exons in the gene or the isoform, where a_i and d_i are the indices of the acceptor and donor junctions of the i -th exon in S . Using only sequence information, the two goals of this work are listed as follows:

1. **Isoform-level Circular RNA Prediction:** Given a gene sequence S and the splicing information of an isoform $\mathcal{E}(s)$, the goal is to identify whether this RNA isoform is a circRNA.
2. **Gene-level Circular RNA Prediction:** Given a gene sequence S and all of its exon-intron boundaries $\mathcal{E}(S)$, this task aims at predicting if any of the junction pairs can backsplice to form a circRNA.

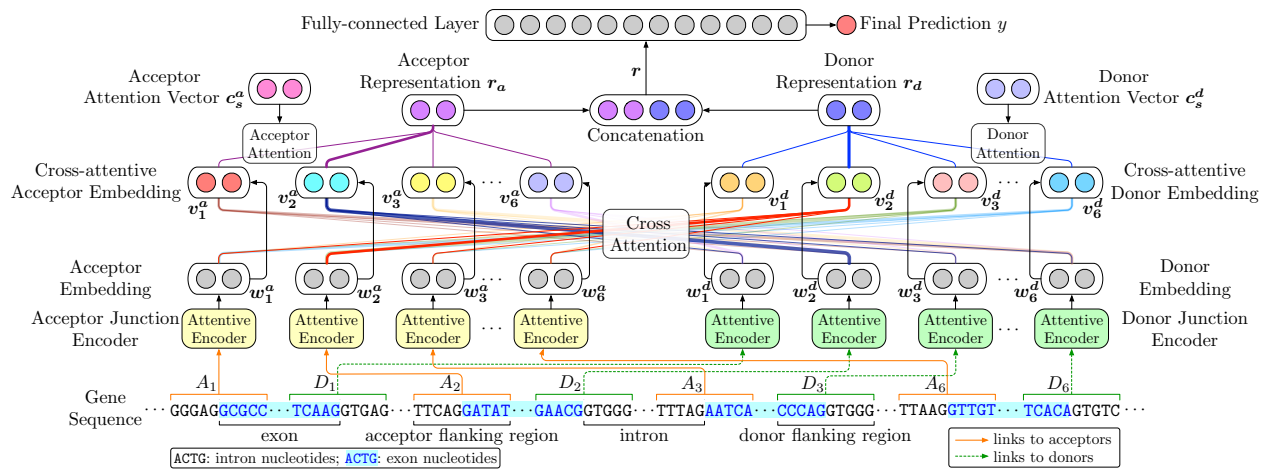


Figure 4.1: The schema of the proposed framework, *Junction Encoder with Deep Interaction* (JEDI), using the gene NM_001080433 with six exons as an example, where the second exon forms backsplicing. A_i and D_j represent the i -th and j -th potential acceptors and donors.

4.2.2 Framework Overview

Figure 4.1 illustrates the general schema of JEDI to predict circRNAs. Each acceptor a_i and donor d_j in the gene sequence are first represented by flanking regions A_i and D_j around exon-intron junctions. Two attentive junction encoders then derive embedding vectors of acceptors and donors, respectively. Based on the embedding vectors, we apply the cross-

attention mechanism to consider deep interactions between acceptors and donors, thereby obtaining donor-aware acceptor embeddings and acceptor-aware donor embeddings. Finally, the attention mechanism is applied again to learn the provided acceptor and donor representations so that the prediction can be inferred by a fully-connected layer based on the representations.

4.2.3 Attentive Junction Encoders

To represent the properties of acceptors and donors in the gene sequence S , we utilize the flanking regions around junctions to derive informative embedding vectors. Specifically, as shown in Figure 4.2, we propose attentive junction encoders using recurrent neural networks (RNNs) and the attention mechanism based on acceptor and donor flanking regions.

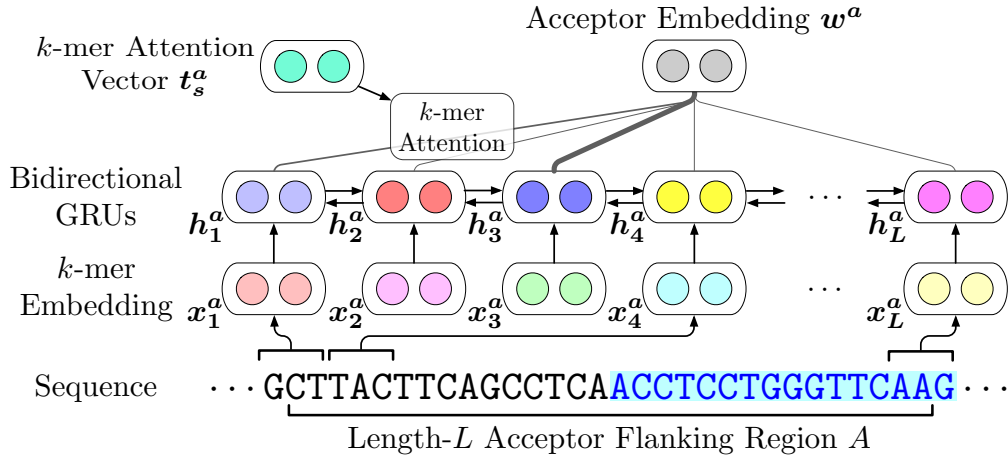


Figure 4.2: The illustration of the attentive encoder for acceptor junctions. Note that the donor junction encoder shares the same model structure with different model parameters.

Flanking Regions as Inputs. For each exon $(a_i, d_i) \in \mathcal{E}(S)$, length- L acceptor and donor flanking regions A_i and D_i can be computed as:

$$A_i = \left[a_i - \left\lfloor \frac{L-1}{2} \right\rfloor, \dots, a_i - 1, a_i, a_i + 1, \dots, a_i + \left\lfloor \frac{L}{2} \right\rfloor \right],$$

$$D_i = \left[d_i - \left\lfloor \frac{L-1}{2} \right\rfloor, \dots, d_i - 1, d_i, d_i + 1, \dots, d_i + \left\lfloor \frac{L}{2} \right\rfloor \right],$$

where $A_i[j]$ and $D_i[j]$ denote the j -th positions on S for the flanking regions of the acceptor a_i

and the donor d_i ; the region length L is a tunable hyper-parameter.

Suppose we are encoding an acceptor a and a donor d with the flanking regions A and D in the gene sequence S for the simplicity.

k -mer Embedding. To represent different positions in the sequence, we use k -mers as representations because k -mers are capable of preserving more complicated local contexts [185]. Each unique k -mer is then mapped to a continuous embedding vector as various deep learning approaches in bioinformatics [54, 242]. Formally, for each position $A[j]$ and $D[j]$, the corresponding k -mer embedding vectors \mathbf{x}_j^a and \mathbf{x}_j^d can be derived as follows:

$$\mathbf{x}_j^a = \mathcal{F} \left(S \left[A[j] - \left\lfloor \frac{K-1}{2} \right\rfloor \dots A[j] + \left\lfloor \frac{K}{2} \right\rfloor \right] \right),$$

$$\mathbf{x}_j^d = \mathcal{F} \left(S \left[D[j] - \left\lfloor \frac{K-1}{2} \right\rfloor \dots D[j] + \left\lfloor \frac{K}{2} \right\rfloor \right] \right),$$

where $\mathcal{F}(\cdot) : \mathcal{V}^K \mapsto \mathbb{R}^l$ is an embedding function mapping a length- K k -mer to a l -dimensional continuous representation; the embedding dimension l and the k -mer length K are two model hyper-parameters. Subsequently, A and D are represented by the corresponding k -mer embedding sequences, $\mathbf{x}^a = [\mathbf{x}_1^a, \dots, \mathbf{x}_L^a]$ and $\mathbf{x}^d = [\mathbf{x}_1^d, \dots, \mathbf{x}_L^d]$.

Bidirectional RNNs. Based on k -mer embedding vectors, we apply bidirectional RNNs (BiRNNs) to learn the sequential properties in genes. The k -mer embedding sequences are scanned twice in both directions as forward and backward passes. During the forward pass, BiRNNs compute forward hidden states $\overrightarrow{\mathbf{h}}^a$ and $\overrightarrow{\mathbf{h}}^d$ as:

$$\overrightarrow{\mathbf{h}}^a = [\overrightarrow{\mathbf{h}}_1^a, \dots, \overrightarrow{\mathbf{h}}_L^a] \text{ and } \overrightarrow{\mathbf{h}}^d = [\overrightarrow{\mathbf{h}}_1^d, \dots, \overrightarrow{\mathbf{h}}_L^d],$$

where $\overrightarrow{\mathbf{h}}_j^a = \overrightarrow{\text{GRU}}_a(\overrightarrow{\mathbf{h}}_{j-1}^a, \mathbf{x}_j^a)$; $\overrightarrow{\mathbf{h}}_j^d = \overrightarrow{\text{GRU}}_d(\overrightarrow{\mathbf{h}}_{j-1}^d, \mathbf{x}_j^d)$. $\overrightarrow{\text{GRU}}_a$ and $\overrightarrow{\text{GRU}}_d$ are gated recurrent units (GRUs) [69] with different parameters for acceptors and donors, respectively. Note that we adopt GRUs instead of other RNN cells like long-short term memory (LSTM) [152] because GRUs require fewer parameters [183]. Similarly, the backward pass reads the sequences

in the opposite order, thereby calculating backward hidden states $\overleftarrow{\mathbf{h}}^a$ and $\overleftarrow{\mathbf{h}}^d$ as:

$$\overleftarrow{\mathbf{h}}^a = [\overleftarrow{\mathbf{h}}_1^a, \dots, \overleftarrow{\mathbf{h}}_L^a] \text{ and } \overleftarrow{\mathbf{h}}^d = [\overleftarrow{\mathbf{h}}_1^d, \dots, \overleftarrow{\mathbf{h}}_L^d],$$

where $\overleftarrow{\mathbf{h}}_j^a = \overleftarrow{\text{GRU}}_a(\overleftarrow{\mathbf{h}}_{j+1}^a, \mathbf{x}_j^a)$; $\overleftarrow{\mathbf{h}}_j^d = \overleftarrow{\text{GRU}}_d(\overleftarrow{\mathbf{h}}_{j+1}^d, \mathbf{x}_j^d)$. To model k -mers with context information, we concatenate forward and backward hidden states as the hidden representations of k -mers in A and D as:

$$\mathbf{h}^a = [\mathbf{h}_1^a, \dots, \mathbf{h}_L^a] \text{ and } \mathbf{h}^d = [\mathbf{h}_1^d, \dots, \mathbf{h}_L^d],$$

where $\mathbf{h}_j^a = [\overrightarrow{\mathbf{h}}_j^a; \overleftarrow{\mathbf{h}}_j^a]$; $\mathbf{h}_j^d = [\overrightarrow{\mathbf{h}}_j^d; \overleftarrow{\mathbf{h}}_j^d]$.

k -mer Attention. Since different k -mers can have unequal importance for representing the properties of splice sites, we introduce the attention mechanism [23] to identify and aggregate the hidden representations of k -mers that are more important than others. The motivation of the attention mechanism is to learn a computational function for automatically estimating the importance score of each item so that the ultimate representation can focus on items that are more significant. More precisely, the importance scores of representations \mathbf{h}_j^a and \mathbf{h}_k^d can be estimated by the k -mer attention vectors \mathbf{t}_s^a and \mathbf{t}_s^d as:

$$\alpha_j^a = \frac{\exp(\mathbf{t}_j^a \top \mathbf{t}_s^a)}{\sum_{j'} \exp(\mathbf{t}_{j'}^a \top \mathbf{t}_s^a)} \text{ and } \alpha_j^d = \frac{\exp(\mathbf{t}_j^d \top \mathbf{t}_s^d)}{\sum_{j'} \exp(\mathbf{t}_{j'}^d \top \mathbf{t}_s^d)},$$

where $\mathbf{t}_j^a = \tanh(\mathcal{F}_t^a(\mathbf{h}_j^a))$; $\mathbf{t}_j^d = \tanh(\mathcal{F}_t^d(\mathbf{h}_j^d))$; $\mathcal{F}_t^a(\cdot)$ and $\mathcal{F}_t^d(\cdot)$ are fully-connected layers. $\tanh(\cdot)$ is the activation function for the convenience of similarity computation. The importance scores are first measured by the inner-products to the k -mer attention vectors and then normalized by a softmax function over the scores of all k -mers. Note that the k -mer attention vectors \mathbf{t}_s^a and \mathbf{t}_s^d are learnable and updated during optimization as model parameters. Finally, the acceptor embedding \mathbf{w}^a of A and the donor embedding \mathbf{w}^d of D can be derived by aggregating the hidden representations of k -mers weighted by their learned importance

scores as:

$$\mathbf{w}^a = \sum_j \alpha_j^a \cdot \mathbf{h}_j^a \text{ and } \mathbf{w}^d = \sum_j \alpha_j^d \cdot \mathbf{h}_j^d.$$

4.2.4 Cross-attention for Modeling Deep Interaction

Modeling interactions among splice sites is essential for circular RNA prediction because backsplices occur when the donors prefer the upstream acceptors over the downstream ones. Inspired by recent successes in natural language processing [146] and computer vision [212], we propose the cross-attention layer to learn deep interaction between acceptors and donors.

Cross-attention Layer. For acceptors, the cross-attention layer aims at deriving cross-attentive acceptor embeddings that not only represent the acceptor sites and their flanking regions but also preserve the knowledge of relevant donors from donor embeddings. Similarly, the cross-attentive donor embeddings are simultaneously obtained for donors. To directly model relations between embeddings, we adopt the dot-product attention mechanism [321] for the cross-attention layer. For each acceptor embedding \mathbf{w}_i^a , the relevance of a donor embedding \mathbf{w}_j^d can be computed by a dot-product $\mathbf{w}_i^{a\top} \mathbf{w}_j^d$ so that the attention weights $\beta_{i,j}^a$ can be calculated with a softmax function over all donors. Likewise, the attention weights $\beta_{j,i}^d$ for each donor embedding \mathbf{w}_j^d can also be measured by dot-products to the acceptor embeddings. Stated formally, we have:

$$\beta_{i,j}^a = \frac{\exp(\mathbf{w}_i^{a\top} \mathbf{w}_j^d)}{\sum_{j'} \exp(\mathbf{w}_i^{a\top} \mathbf{w}_{j'}^d)} \text{ and } \beta_{j,i}^d = \frac{\exp(\mathbf{w}_j^{d\top} \mathbf{w}_i^a)}{\sum_{i'} \exp(\mathbf{w}_j^{d\top} \mathbf{w}_{i'}^a)}.$$

Therefore, the cross-attentive embeddings of acceptors and donors can then be derived by aggregations based on the attention weights as:

$$\mathbf{v}_i^a = \sum_j \beta_{i,j}^a \cdot \mathbf{w}_j^d \text{ and } \mathbf{v}_j^d = \sum_i \beta_{j,i}^d \cdot \mathbf{w}_i^a.$$

Note that we do not utilize the multi-head attention mechanism [321] because it requires much more massive training data to learn multiple projection matrices. As shown in Sec-

tion 4.3, the vanilla dot-product attention is sufficient to obtain satisfactory predictions with significant improvements over baselines.

4.2.5 Circular RNA Prediction

To predict circRNAs, we apply the attention mechanism [23] again to aggregate cross-attentive acceptor and donor embeddings into an acceptor representation and a donor representation as ultimate features to predict circRNAs.

Acceptor and Donor Attention. Although the cross-attention layer provides information cross-attentive embeddings for all acceptors and donors, most of the splice sites can be irrelevant to backsplicing. To tackle this issue, we present the acceptor and donor attention to identify splice sites that are more important than other ones. Similar to k -mer attention, the importance scores of cross-attentive embeddings for acceptors and donors can be computed as:

$$\gamma_i^a = \frac{\exp(\mathbf{c}_i^a \top \mathbf{c}_s^a)}{\sum_{i'} \exp(\mathbf{c}_{i'}^a \top \mathbf{c}_s^a)} \text{ and } \gamma_j^d = \frac{\exp(\mathbf{c}_j^d \top \mathbf{c}_s^d)}{\sum_{j'} \exp(\mathbf{c}_{j'}^d \top \mathbf{c}_s^d)},$$

where $\mathbf{c}_i^a = \tanh(\mathcal{F}_c^a(\mathbf{v}_i^a))$; $\mathbf{c}_j^d = \tanh(\mathcal{F}_c^d(\mathbf{v}_j^d))$; $\mathcal{F}_c^a(\cdot)$ and $\mathcal{F}_c^d(\cdot)$ are fully-connected layers. Subsequently, the acceptor and donor representations \mathbf{r}_a and \mathbf{r}_d can be derived based on the attention weights of cross-attentive embeddings as:

$$\mathbf{r}_a = \sum_i \gamma_i^a \cdot \mathbf{v}_i^a \text{ and } \mathbf{r}_d = \sum_i \gamma_i^d \cdot \mathbf{v}_i^d.$$

Prediction as Binary Classification. Here we treat circular RNA prediction as a binary classification task. More specifically, we estimate a probabilistic score \hat{y} to approximate the probability of existing circRNA. The ultimate features \mathbf{r} for machine learning are provided by concatenating the acceptor and donor representations as $\mathbf{r} = [\mathbf{r}_a; \mathbf{r}_d]$. Finally, the probabilistic score \hat{y} can be computed by a sigmoid function with a fully-connected layer as follows:

$$\hat{y} = \sigma(\mathcal{F}_p(\text{ReLU}(\mathcal{F}_r(\mathbf{r})))),$$

where $\mathcal{F}_p(\cdot)$ and $\mathcal{F}_r(\cdot)$ are fully-connected layers; $\text{ReLU}(\cdot)$ is the activation function for the hidden layer [128]; $\sigma(\cdot)$ is the logistic sigmoid function [142]. The binary prediction can be further generated by a binary indicator function as $\mathbb{1}(\hat{y} > 0.5)$.

4.2.6 Learning and Optimization

To solve circular RNA prediction as a binary classification problem, JEDI is optimized with a binary cross-entropy [151]. Formally, the loss function for optimization can be written as follows:

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \|\theta\|_2,$$

where N is the number of training gene sequences; y_i is a binary indicator demonstrating whether the i -th training sequence exists a circRNA; \hat{y}_i is the approximated probabilistic score for the i -th training gene sequence; λ is the L2-regularization weight for the set of model parameters θ .

4.2.7 Remarks on the Interpretability of JEDI

The usage of attention mechanisms is one of the most essential keys in JEDI, including the donor and acceptor attention, the cross-attention layer, and the k -mer attention in junction encoders. In addition to choosing important information to optimize the objective, one of the most significant benefits of using attention mechanisms is the interpretability.

Application: Zero-shot Backsplicing Discovery. For circRNAs, the attention weights can become interpretable hints for discovering backsplicing without training on the annotated backspliced sites. For example, when the model is optimized for accurately predicting circRNAs, the weights of donor attention are reformed to denote the important and relevant donors, which are preferred for the upstream acceptors to backsplice. In other words, the probabilistic attention weight γ_j^d for each donor d_j can be interpreted as the probability of being a backsplice donor site as:

$$P(d_j) = \gamma_j^d,$$

where the softmax function guarantees $\sum_j P(d_j) = 1$. Similarly, the attention weight $\beta_{j,i}^d$ of each acceptor a_i for deriving the cross-attentive embedding of the donor d_j can be explained as the conditional probability of being selected as the backsplice acceptor site from the donor d_j as:

$$P(a_i | d_j) = \beta_{j,i}^d,$$

where we also have the probabilistic property $\forall j : \sum_i \beta_{j,i}^d = 1$ from the softmax function. Based on the above interpretations, for any pair of a donor d_j and an acceptor a_i , the probability of forming a backsplice can be approximated by decomposing the joint probability $P(d_j, a_i)$ as:

$$P(d_j, a_i) = P(d_j)P(a_i | d_j) = \gamma_j^d \beta_{j,i}^d.$$

Therefore, without any training backsplice site annotation as zero-shot learning [296], we can transfer the knowledge in the training data for circular RNA prediction to discover potential backsplice sites by ranking the pairs of acceptors and donors according to $P(d_j, a_i)$. Particularly, the interpretations can be also aligned with the process of RNA splicing, bringing more biological insights into JEDI. In Section 4.3.6, we further conduct experiments to demonstrate that JEDI is capable of addressing the task of zero-shot backsplicing discovery.

4.3 Experiments

In this section, we conduct extensive experiments on benchmark datasets for two tasks and in-depth analysis to verify the performance and robustness of the proposed framework, JEDI.

4.3.1 Datasets

Human circRNA. We use the benchmark dataset generated by Chaabane et al. [54]. The positive data generation follows a similar setting as described in Pan and Xiong [258] to derive 31,939 isoforms of human circRNAs covering a diverse range of tissues and cell types from the circRNADb database [64]. The negative set is composed of other lncRNAs, such as processed

transcripts, anti-sense, sense intronic and sense overlapping. It is constructed based on the annotation provided by GENCODE v19 [118] with strong evidence. Specifically, only the experimentally validated or manually annotated transcripts are considered, resulting in 19,683 negative isoforms. To avoid information leaks through training and evaluating on paralogous genes, we group isoforms into the same cluster if they come from the same gene or duplicated genes. The duplicated gene information is retrieved from the Duplicated Genes Database [256]. Combining both the positive and negative cases, these 51,622 isoforms are grouped into 23,674 clusters. The clusters are divided into five parts to conduct 5-fold cross validation. The sequences of all positive and negative cases are based on hg19.

Mouse circRNA on isoform level. The mouse circRNAs are obtained through *circbase* [127] which contains public circRNA datasets for several species reported in literature. There are 1,903 mouse circRNAs. Using the annotation provided by GENCODE vM1, we randomly select other lincRNAs, generating 1,522 negative cases. The sequences of all positive and negative cases are based on mm9.

4.3.2 Experimental Settings

Baseline Methods. To evaluate the performance of JEDI, we compare with eight competitive baseline methods, including circDeep [54], PredcircRNA [258], DeepCirCode [330], nRC [114], Support Vector Machines (SVM), Random Forest (RF), attentive-CNN (Att-CNN), and attentive-RNN (Att-RNN). Specifically, circDeep and PredcircRNA are the state-of-the-art circular RNA prediction methods. DeepCirCode originally takes individual splice site pairs for backsplicing prediction, which is another research problem, and leads to an enormous number of false alarms in our problem settings. To conduct fair comparisons, we modify DeepCirCode by extending the inputs to all sites and aggregating CNN representations for acceptors and donors with two max-pooling layers before applying its model structure. nRC represents lincRNA classification methods that are compatible to solve circular RNA prediction as a sequence classification problem. SVM and RF apply conventional statistical learning frameworks with the compositional k -mer features proposed by Wang

and Wang [329] for backsplicing prediction. Attentive CNN and RNN as popular deep learning approaches utilize CNNs and RNNs with the attention mechanism [23] for sequence modeling, thereby predicting circRNAs based on a fully-connected hidden layer with the ReLU activation function [128]. Note that we do not compare with CIRCexplorer2 [370] and CIRC [123] because they aim at aligning the sequencing reads to known circRNAs, and performing de-novo assembly of novo circRNAs, which is a completely different approach than our proposed method.

Evaluation Metrics and Protocol. Six conventional binary classification metrics are selected as the evaluation metrics for both tasks, including the overall accuracy (Acc), precision (Prec), sensitivity (Sens), specificity (Spec), F1-score, as well as Matthew correlation coefficient (MCC) and the Area under the ROC curve (AUC) on positive cases. For all metrics, the higher metric scores indicate more satisfactory performance. We conduct a 5-fold cross-validation for evaluation on both isoform-level and gene-level circular RNA prediction. Specifically, for each task, the data are randomly shuffled and evenly partitioned into five non-overlapping subsets. In the five folds of experiments, each subset has a chance to be considered as the testing data for assessing the model trained by the remaining four subsets, thereby ensuring an unbiased and fair evaluation. Finally, we evaluate the methods by aggregating the scores over the 5-fold experiments for each metric.

Implementation Details. Our approach, JEDI, is implemented in Tensorflow [1] and released in GitHub as shown in Abstract. The AMSGrad optimizer [275] is adopted to optimize the model parameters with a learning rate $\eta = 10^{-3}$, exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a batch size 64, and an L2-regularization weight $\lambda = 10^{-3}$. As the hyper-parameters of JEDI, the k -mer size K and the number of dimensions l for k -mer embeddings are set to 3 and 128. We set the length of flanking regions L to 4. The hidden state size of GRUs for both directions in junction encoders is 128. The size of all attention vectors is set to 16. The number of units in the fully-connected hidden layer $\mathcal{F}_r(\cdot)$ for circular RNA prediction is 128. The model parameters are trained until the convergence for each fold in cross-validation. For the baseline methods, the experiments for circDeep,

Table 4.1: Evaluation of isoform-level circular RNA prediction based on the 5-fold cross-validation. We report the mean and standard deviation for each metric.

Method	Accuracy	Precision	Sensitivity	Specificity	F1-score	MCC	AUC
SVM	0.728 ± 0.069	0.748 ± 0.097	0.893 ± 0.108	0.453 ± 0.341	0.804 ± 0.026	0.403 ± 0.178	0.673 ± 0.120
RF	0.761 ± 0.008	0.776 ± 0.012	0.861 ± 0.008	0.598 ± 0.008	0.817 ± 0.010	0.480 ± 0.012	0.730 ± 0.005
Att-CNN	0.752 ± 0.007	0.774 ± 0.026	0.853 ± 0.039	0.587 ± 0.053	0.811 ± 0.008	0.461 ± 0.017	0.720 ± 0.010
Att-RNN	0.764 ± 0.008	0.777 ± 0.016	0.858 ± 0.035	0.617 ± 0.051	0.815 ± 0.009	0.496 ± 0.013	0.738 ± 0.011
nRC	0.756 ± 0.012	0.784 ± 0.039	0.841 ± 0.060	0.619 ± 0.100	0.809 ± 0.012	0.478 ± 0.028	0.828 ± 0.009
PredcircRNA	0.655 ± 0.008	0.698 ± 0.014	0.595 ± 0.007	0.720 ± 0.012	0.642 ± 0.009	0.317 ± 0.016	0.588 ± 0.010
circDeep	0.875 ± 0.010	0.939 ± 0.013	0.816 ± 0.022	0.941 ± 0.014	0.873 ± 0.011	0.758 ± 0.019	0.740 ± 0.013
DeepCirCode	0.900 ± 0.004	0.935 ± 0.023	0.902 ± 0.025	0.897 ± 0.038	0.918 ± 0.004	0.791 ± 0.007	0.899 ± 0.008
JEDI	0.988 ± 0.001	0.991 ± 0.003	0.991 ± 0.003	0.984 ± 0.004	0.990 ± 0.001	0.974 ± 0.001	0.987 ± 0.001

PredcircRNA, and nRC are carried out according to the publicly available implementations released by the authors of original papers. SVM and RF are implemented in Python with the scikit-learn library [262]. As for deep learning approaches, DeepCirCode, Attentive-CNN, and Attentive-RNN are implemented in Tensorflow, which is the same as our proposed JEDI. For all methods, we conduct parameter fine-tuning for fair comparisons. All of the experiments are also equitably conducted on a computational server with one NVIDIA Tesla V100 GPU and one 20-core Intel Xeon CPU E5-2698 v4 @ 2.20GHz.

4.3.3 Isoform-level Circular RNA Prediction

Table 4.1 shows the performance of all methods for isoform-level circular RNA prediction. Among the baseline methods, circDeep as the state-of-the-art approach and DeepCirCode considering junctions perform the best. It is because circDeep explicitly accounts for the reverse complimentary sequence matches in flanking regions of the junctions, and DeepCirCode models the flanking regions with deep learning. Consistent with the previous study [54], PredcircRNA performs worse than circDeep. With compositional k -mer based features designed for backsplicing prediction, SVM and RF surprisingly outperform PredircrRNA by 11.13% and 16.14% in accuracy. It not only shows that the k -mers are universally beneficial across different tasks but also emphasizes the rationality of using k -mers for junction encoders in JEDI. As an lncRNC classification method, nRC also shows its potential for circRNA prediction with a 15.37% improvement over PredcircRNA in accuracy. Although Att-CNN and Att-RNN utilize the attention mechanism, they can only model the whole

Table 4.2: Evaluation of gene-level circular RNA prediction based on the 5-fold cross-validation. We report the mean and standard deviation for each metric.

Method	Accuracy	Precision	Sensitivity	Specificity	F1-score	MCC	AUC
SVM	0.712 ± 0.056	0.835 ± 0.060	0.584 ± 0.196	0.853 ± 0.107	0.665 ± 0.121	0.466 ± 0.072	0.719 ± 0.050
RF	0.732 ± 0.006	0.704 ± 0.010	0.850 ± 0.004	0.602 ± 0.013	0.770 ± 0.005	0.469 ± 0.011	0.726 ± 0.006
Att-CNN	0.725 ± 0.005	0.756 ± 0.028	0.830 ± 0.045	0.552 ± 0.062	0.790 ± 0.009	0.401 ± 0.007	0.691 ± 0.009
Att-RNN	0.730 ± 0.008	0.757 ± 0.025	0.834 ± 0.041	0.564 ± 0.054	0.792 ± 0.009	0.416 ± 0.015	0.699 ± 0.010
nRC	0.729 ± 0.009	0.738 ± 0.033	0.759 ± 0.059	0.696 ± 0.066	0.746 ± 0.015	0.459 ± 0.019	0.801 ± 0.008
PredcircRNA	0.619 ± 0.003	0.659 ± 0.009	0.573 ± 0.009	0.670 ± 0.012	0.613 ± 0.005	0.243 ± 0.008	0.609 ± 0.031
circDeep	0.839 ± 0.007	0.878 ± 0.014	0.806 ± 0.009	0.875 ± 0.013	0.840 ± 0.007	0.681 ± 0.014	0.752 ± 0.011
DeepCirCode	0.863 ± 0.022	0.894 ± 0.027	0.842 ± 0.061	0.886 ± 0.037	0.866 ± 0.027	0.730 ± 0.038	0.864 ± 0.020
JEDI	0.966 ± 0.006	0.967 ± 0.008	0.969 ± 0.018	0.963 ± 0.010	0.968 ± 0.006	0.932 ± 0.012	0.966 ± 0.005

sequences and present limited performance without any knowledge of junctions. As our proposed approach, JEDI significantly outperforms all of the baseline methods across all evaluation metrics. Particularly, JEDI achieves 9.80% and 7.90% improvements over DeepCirCode in accuracy and F1-score, respectively. The experimental results have demonstrated the effectiveness of junction encoders and the cross-attention layer that models deep interaction among splice sites.

4.3.4 Gene-level Circular RNA Prediction

We further evaluate all methods on gene-level circular RNA prediction. Note that this task is more difficult than the isoform-level prediction because each junction can be a backsplice site. Since a full gene sequence can encode for multiple isoforms, there can be multiple site pairs forming backsplices for different isoforms. Consequently, models cannot learn from absolute positions for circRNA prediction. As shown in Table 4.2, all methods deliver worse performance than the results in isoform-level circRNA prediction. Notably, the evaluation metrics have demonstrated a similar trend as shown in Table 4.1. DeepCirCode and circDeep are still the best baseline methods, showing the robustness of exploiting the knowledge about splice junctions. SVM, RF, and nRC still outperform PredircRNA by at least 15.08% in accuracy. Att-CNN and Att-RNN using the attention mechanism still fail to obtain extraordinary performance because they are unaware of junction information, which is essential for backsplicing events. In this more difficult task, JEDI consistently surpasses all of the baseline methods across all evaluation metrics. For instance, JEDI beats DeepCirCode

Table 4.3: Independent study of isoform-level circular RNA prediction for mouse circRNAs based on the models trained on human circRNAs.

Method	Acc	Prec	Sens	Spec	F1	MCC	AUC
SVM	0.7328	0.7742	0.8108	0.6011	0.7921	0.4196	0.7059
RF	0.7186	0.7393	0.8523	0.4929	0.7918	0.3733	0.6726
Att-CNN	0.7264	0.7452	0.7957	0.6330	0.7696	0.4352	0.7143
Att-RNN	0.7030	0.7189	0.7930	0.5816	0.7541	0.3844	0.6873
PredcircRNA	0.5696	0.6218	0.5056	0.6437	0.5577	0.1501	0.6067
nRC	0.7410	0.7662	0.8455	0.5647	0.8039	0.4298	0.8097
circDeep	0.6140	0.7495	0.6982	0.7509	0.7229	0.4491	0.7669
DeepCirCode	0.8129	0.9271	0.7620	0.8989	0.8365	0.6392	0.8304
JEDI	0.8654	0.9074	0.8749	0.8493	0.8909	0.7162	0.8621

by 11.94% and 11.75% in accuracy and F1-score, respectively. The experimental results further reveal that our proposed JEDI is capable of tackling different scenarios of circular RNA prediction with consistently satisfactory predictions.

4.3.5 Independent Study on Mouse circRNAs

To demonstrate the robustness of JEDI, we conduct an independent study on the dataset of mouse circRNAs. Previous studies have shown that circRNAs are evolutionarily conserved [30, 170, 303], and thus we evaluate the potential of predicting the circRNAs across different species. More precisely, we train each method using the human dataset on isoform-level, thereby predicting the circRNAs on the mouse dataset. Note that some of the required features for PredcircRNA are missing on the mouse datasets. In addition to this, PredictcRNA perform the worst in other experiments. For these reasons, we exclude PredcircRNA from this study. Table 4.3 presents the experimental results of the independent study. Compared to the experiments conducted on the same species as shown in Table 4.1, most of the deep learning methods have slightly lower performance because they are specifically optimized for human data; SVM and RF have similar performance in the independent study probably because k -mer features are simpler and more general to different species. Interestingly, the accuracy of circDeep significantly drops in the study. It is likely due to the fact that circDeep heavily pre-trains the sequence modeling on human data with the serious

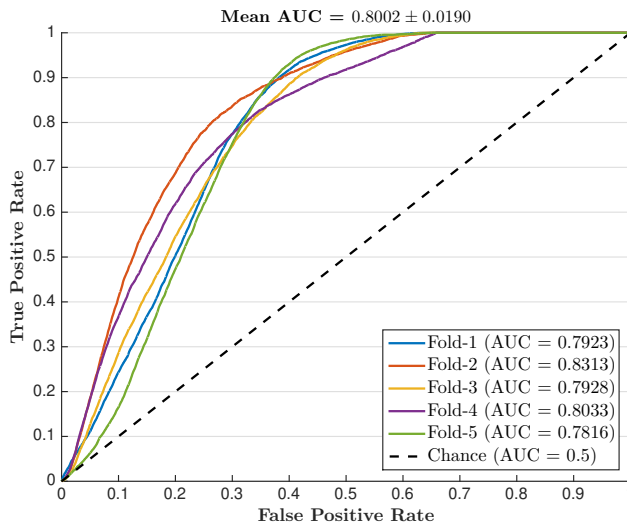


Figure 4.3: The ROC curves for zero-shot backsplicing discovery based on the 5-fold cross-validation and JEDI trained for gene-level circular RNA prediction.

over-fitting phenomenon. As a result, our proposed JEDI still outperforms all of the baseline methods. It demonstrates that JEDI is robust across the datasets of different species.

4.3.6 Zero-shot Backsplicing Discovery

As mentioned in Section 4.2.7, the interpretability of the attention mechanisms and the cross-attention layer enables JEDI to achieve zero-shot backsplicing discovery. To evaluate the performance of zero-shot backsplicing, we compute the probabilistic score $P(d_j, a_i)$ using the attention weights γ_j^d and $\beta_{j,i}^d$, thereby indicating the likelihood of forming a backsplice for each pair of a candidate donor d_j and a candidate acceptor a_i . Hence, we can simply evaluate the probabilistic scores with the receiver operating characteristic (ROC) curve and the area under the ROC curve (AUC). Note that here we still apply 5-fold cross-validation for experiments based on the gene-level human circRNA dataset. Since none of the existing methods can address the task of zero-shot backsplicing prediction, we compare with random guessing, which is equivalent to the chance line in ROCs with an AUC score of 0.5. Figure 4.3 depicts the ROC curves with AUC scores over five folds of experiments. The results show that the backspliced site pairs discovered by JEDI are effective with an average AUC score of 0.8002. In addition, JEDI is also robust in this task with a small standard deviation of

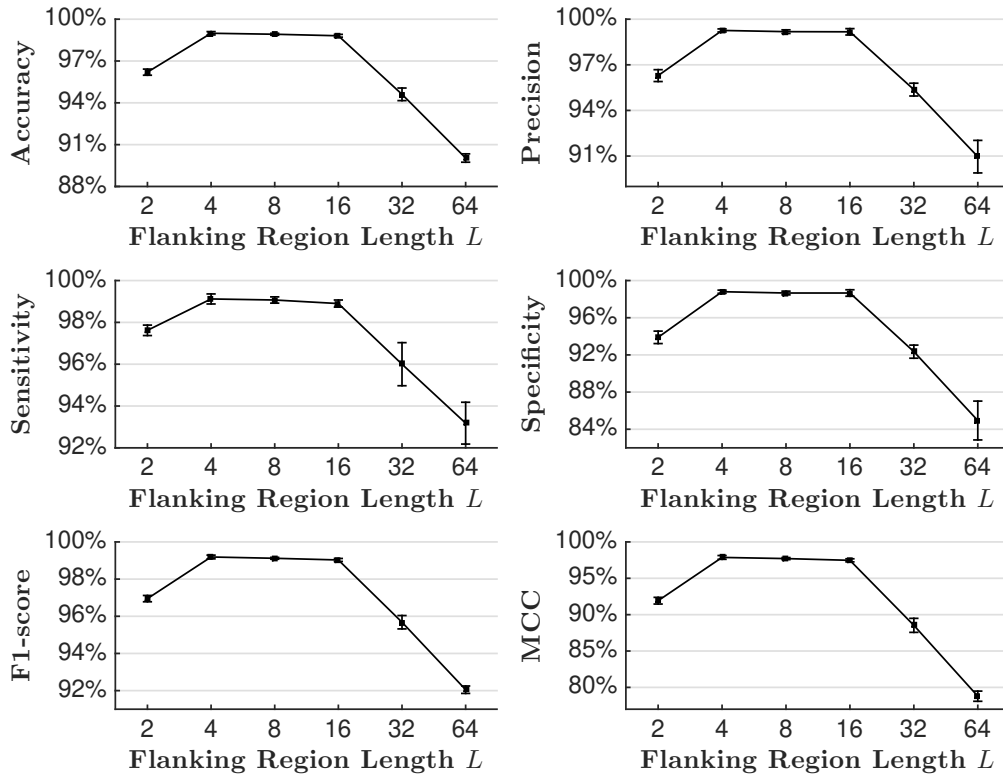


Figure 4.4: The isoform-level circular RNA prediction performance of JEDI with different flanking region lengths L based on the 5-fold cross validation. We report the mean for each metric and apply error bars to indicate standard deviations.

AUC scores. Since the cross-attention layer is a major contribution in JEDI, we conduct another study to analyze how donor and acceptor embeddings interact with each other in Section S1 in the supplementary materials.

4.3.7 Analysis and Discussions

In this section, we first discuss the impacts of hyper-parameters for JEDI and then conduct the run-time analysis for all methods to verify the model efficiency of JEDI. Note that, for hyper-parameter analysis, we adjust the target hyper-parameter while other hyper-parameters are fixed as the values utilized in the experiments as mentioned in Section 4.3.2.

Length of Flanking Regions L . The flanking region length L for junction encoders plays an important role in JEDI to represent splice sites. Figure 4.4 illustrates the circular RNA prediction performance of JEDI over different flanking region lengths. For all evaluation

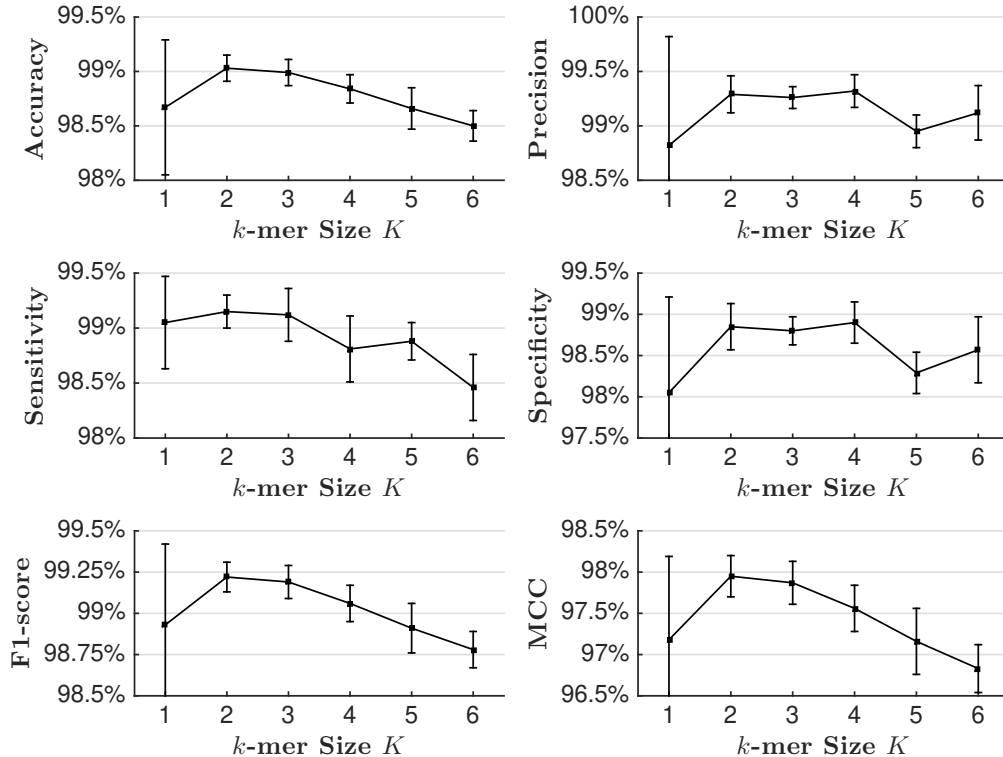


Figure 4.5: The isoform-level circular RNA prediction performance of JEDI with different k -mer sizes K based on the 5-fold cross validation. We report the mean for each metric and apply error bars to indicate standard deviations.

metrics, the performance slightly improves when L increases to 4. However, the performance significantly drops when $L \geq 32$. It shows that nucleotides nearer to junctions are more important than other ones for predicting backsplicing. This result is also consistent with previous studies on RNA splicing [253]. Moreover, circRNAs tend to contain fewer nucleotides than other transcripts from the same gene [169], so excessive and redundant information could only lead to noises and lower the prediction performance.

Size of k -mers K . The derivation of k -mers is crucial for JEDI because JEDI treats k -mers as the fundamental inputs over gene sequences. Figure 4.5 shows how the size of k -mers affects the prediction performance. JEDI performs the best with 2-mers and 3-mers when the performance gets worse with longer or shorter k -mers. It could be because a small k -mer size makes k -mers less significant for representations. In addition, the embedding space of long k -mers could be too enormous for JEDI to learn with limited training data. It is

also worthwhile to mention that 1-mers lead to much higher standard deviations because of their low significance induces high instability and sensitive embeddings during the learning process. This finding is also consistent with previous studies [282].

In addition to the flanking region length L and the k -mer size K , we also conduct the analysis to study how the embedding dimension l affects the performance in Section S2 in the supplementary materials.

Table 4.4: Run-time analysis on isoform-level circular RNA prediction in seconds (**s**), minutes (**m**), and hours (**h**), based on the 5-fold cross-validation. We report the mean of the training time (over five folds).

Method	Time	Method	Time	Method	Time
SVM	28.76s	Att-CNN	13.35m	circDeep	>24h
RF	21.03s	Att-RNN	51.53m	DeepCirCode	3.80m
nRC	4.07m	PredcircRNA	43.66s	JEDI	2.75m

Run-time Analysis. To verify the efficiency of JEDI, we conduct the run-time analysis for all methods in our experiments based on the task of isoform-level circular RNA prediction. For fair comparisons, all methods can access the same computational resources. Note that we only consider the time in training and testing. The run-time of feature extraction and disk I/O are ignored because the features can be pre-processed. Disk I/O can be affected by many factors that are irrelevant to methods, such as I/O scheduling in operating systems. As shown in Table 4.4, JEDI is efficient and averagely needs only less than three minutes because it only focuses on junctions and flanking regions. Similarly, DeepCirCode, which is also a junction based deep learning method, has comparable execution time to JEDI. In contrast, Att-CNN and Att-RNN are relatively inefficient because they scan the whole sequences in every training batch, where Att-RNN with non-parallelizable recurrent units is slower. Although nRC reads the whole sequences, it runs faster than some attention-based methods because of its simpler model structure. SVM, RF, and PredcircRNA are the most efficient because they apply straightforward statistical machine learning frameworks for training. As a side note, the feature extraction of PredcircRNA is extremely expensive in execution time and averagely costs more than 28 hours to extract multi-facet features in

our experiments. circDeep is the most inefficient in our experiments because it consists of many time-consuming components, such as embedding and LSTM pre-training.

4.4 Conclusions

We propose a novel end-to-end deep learning approach for circular RNA prediction by learning to appropriately model splice sites with flanking regions around junctions and studying the deep relationships among these sites. The attentive junction encoders are first introduced to represent each splice site, and the innovative cross-attention layer is proposed to learn the deep interaction among splice sites. Moreover, JEDI is capable of discovering backspliced site pairs without training on annotated site pairs. The experimental results demonstrate that JEDI is effective and robust in circular RNA prediction on different data levels and across different species. Most importantly, the backspliced site pairs discovered by JEDI are promising as they designate the hotspots for circular RNAs formation. The reasons and insights for these observations and discoveries can be concluded as follows: (1) JEDI only models valuable and essential flanking regions around the junctions of splice sites, thereby discarding irrelevant and redundant information for circular RNA prediction; (2) the properties of splice sites and essential information for forming circular RNAs can be well-preserved by junction encoders; (3) the attention mechanisms and the cross-attention layer provide intuitive and interpretable hints to implicitly model the backsplicing events as demonstrated in the experiments. Due to data limitation, we are only able to examine the effectiveness of transferring the learned knowledge between humans and mice. As a future direction, we plan to experiment with more species when more data is available. Additionally, we also plan on exploring the potential to extend JEDI to support circRNA prediction from sequencing reads.

CHAPTER 5

Heterogeneous Network Representation Learning with Meta-context Aware Random Walk

Graphs are one of the most universal data types to describe the complex relations among heterogeneous entities in the real world. Therefore, it is very important to develop effective and robust representation learning methods for heterogeneous graphs. In this chapter, we propose a novel unsupervised learning framework to derive machine-readable representations for nodes in heterogeneous networks based on the types of neighbor nodes as meta-context information. We also demonstrate the wide impacts of our approach in diverse machine learning tasks and various real-world applications.

5.1 Introduction

Network analysis has already been a prevalent research topic because of its enormous potential in many downstream applications, such as node classification [315], node clustering [254], and link prediction [219]. More specifically, most of the important tasks in network analysis involve predictions over nodes and edges. However, the sparsity of networks usually results in significant difficulty of generalization for machine learning models. To resolve this issue, one of the most popular approaches is to map nodes to continuous low-dimensional representations as embeddings that preserve the structural information and semantics of nodes [47].

To efficiently learn node representations, random walks have been widely exploited to preserve the proximity between node pairs [135, 264]. More precisely, the embedded repre-

sentations of nodes are optimized to infer the nearby nodes on random walks [264] with a skip-gram model [241] inspired by word embedding in the field of natural language processing [241]. Moreover, the complicated proximity structures of networks can be also gained by sampling biased random walks [135]. Practically, each of the generated random walks can be treated as a word sequence so that the task of network embedding is equivalent to the setting of word embedding [98, 135, 164, 264]. More specifically, a sliding window is applied to capture the nearby nodes as the context for each node over random walks. To ensure the coverage of the nodes for learning representations, most of the existing approaches simply sample a few random walks starting from each of the nodes. However, there are a few shortcomings for the existing sampling approaches. First, one-directional random walks that evenly start from all of the nodes would favor nodes with higher degree and betweenness scores when nodes in the network should be equally important for the downstream applications. Second, tail nodes tend to be visited at the very beginning of random walks, especially for the random walks starting from them. As a result, the number of context nodes in the sliding window will be much underestimated for the tail nodes. In addition, the tail nodes will have fewer chances to be observed as the context of other nodes during optimization.

Compared to homogeneous networks with a singular type of node, heterogeneous networks with various types of nodes are more common in real-world applications. Although the homogeneous network embedding methods can still learn the representations for heterogeneous networks, the information of node types can be significantly neglected. As a result, the semantics of the heterogeneous knowledge in networks is totally lost in the embeddings. To leverage the heterogeneous knowledge in networks for representation learning, existing methods usually rely on meta-paths [306], which are predefined sequences of node types. In other words, different meta-paths indicate distinct human-explainable semantics. For example, the meta-paths **APA** and **APVPA** are used to indicate that two authors had co-authorship and published papers in the same venue respectively, where **A**, **P**, and **V** are the node types referring to author, paper, and venue in a heterogeneous bibliographic network. To exploit the meta-paths, most of the existing heterogeneous network embedding methods guide the

generated random walks through a predefined set of meta-paths so that the prior knowledge can be incorporated into the produced node sequences [63, 98, 122, 291]. For instance, each meta-path can be solely applied to measure the relationship between two nodes with a short random walk [122, 291]; different meta-paths may also overlap to approximate longer random walks as a mixture of prior knowledge [63, 98]. However, the choices of random walk significantly affect the quality of network representations [164]. Accordingly, the requirement of high-quality meta-paths that are hand-picked by domain experts leads to reduced robustness for general tasks. In addition, the usage of meta-paths can limit and distort the understanding of the network structures. More precisely, given a limited set of meta-paths, a new path in a network is less likely to be induced. Even though some works [164] have proposed to employ specific strategies to guide random walks instead of using meta-paths, adjusted random walks can still be biased and overlook some vital network structures.

To learn network representations with random walks, one of the most popular optimized approaches is the skip-gram model inspired by word embedding in the field of natural language processing [241]. Each of the generated random walks can be treated as a word sequence so that the task of network embedding is equivalent to the setting of word embedding [98, 135, 164, 264]. More specifically, a sliding window is applied to capture the nearby nodes as the context for each node over random walks. To ensure the coverage of the nodes for learning representations, most of the existing approaches simply sample a few random walks starting from each of the nodes. However, there are a few shortcomings for the existing sampling approaches. First, one-directional random walks that evenly start from all of the nodes would favor nodes with higher degree and betweenness scores when nodes in the network should be equally important for the downstream applications. Second, tail nodes tend to be visited at the very beginning of random walks, especially for the random walks starting from them. As a result, the number of context nodes in the sliding window will be much underestimated for the tail nodes. In addition, the tail nodes will have fewer chances to be observed as the context of other nodes during optimization.

In this chapter, Meta-context Aware Random Walk (MARU) is proposed to address the

limitations of the existing heterogeneous network embedding approaches. More specifically, we focus on deriving robust embeddings that are more comprehensive and fair to represent the heterogeneous networks. The algorithm of bidirectional extended random walks is first introduced to alleviate the bias caused by classical random walks. Instead of manipulating random walks [98, 135, 164], we employ general random walks for a more comprehensive understanding of network structures and encode the types of surrounding nodes as meta-contexts to incorporate heterogeneous knowledge. Given a node and its meta-contexts in the random walk, we extend the skip-gram model to infer not only the nearby nodes but also their corresponding meta-contexts. In other words, the learned representations can reflect various situations in terms of different meta-contexts, thereby describing the nature of heterogeneous networks more precisely. Here, we summarize our contributions in the following.

- To the best of our knowledge, this is the first work to address the bias of classical random walks for network representation learning. For the tail nodes with lower degree and betweenness scores, the proposed bidirectional extended random walks can capture the context and optimize the representations more fairly and comprehensively.
- We propose the framework MARU, generating network representations that simultaneously capture general network structures and local heterogeneous knowledge. More specifically, leveraging the types of surrounding nodes as meta-contexts enable the model to represent different semantics according to local contexts in random walks. Hence, the learned network representations are more robust to preserve the properties of heterogeneous networks.
- Extensive experiments conducted on three large-scale real-world datasets indicate that MARU significantly outperforms existing heterogeneous network embedding methods. A study of parameter sensitivity then demonstrates the robustness of the proposed framework across different situations. In addition, we will release our implementations to facilitate future research.

5.2 Problem Statement

In this section, we first introduce the notations of heterogeneous networks and then formally define the objective of learning heterogeneous network representations.

5.2.1 Heterogeneous Network

We first formally define the notations to represent heterogeneous networks. Note that the definition is consistent with previous studies [98, 305, 307].

Definition 5.1 (Heterogeneous Network). A heterogeneous network is defined as a graph $G = (V, E, T)$, where V is the set of nodes; $E \subseteq V \times V$ is the set of edges connecting nodes; T represents the set of node types. For each node $v \in V$, a mapping function $\psi(v) \in T$ indicates the corresponding type of the node.

To simplify the representation and implementation, for each node v , we denote the neighbors in the graph as

$$N(v) = \{v_i \mid \forall (v, v_i) \in E\},$$

which can be treated as an adjacency list [77] generated by the edge set E .

5.2.2 Problem Definition

We formalize the problem of learning heterogeneous network representations based on the aforementioned notations.

Problem 5.1 (Representation Learning for Heterogeneous Networks). Given a heterogeneous network $G = (V, E, T)$, for each node $v \in V$, the task aims to learn a d -dimensional embedding vector $\Phi(v) : V \rightarrow \mathbb{R}^d$, where $d \ll |V|$, so that $\Phi(v)$ can capture the structural information and semantic knowledge of the node.

More specifically, the network representations project nodes onto a d -dimensional continuous latent feature space. Note that although nodes can belong to different types, all of

the nodes are projected on the identical feature space for the convenience of representing relationships among different nodes. As a result, the learned node representations can further benefit various data mining tasks for heterogeneous networks, such as node classification, node clustering, and link prediction. Moreover, heterogeneous network representation learning is an unsupervised machine learning task. In other words, the representations can be acquired with only the network and then directly applied to various downstream applications for heterogeneous network data mining. Therefore, the problem of heterogeneous network representation learning is important and beneficial.

5.3 MARU for Heterogeneous Network Embedding

In this section, we present the proposed framework, Meta-context Aware Random Walks (MARU), for learning heterogeneous network representations.

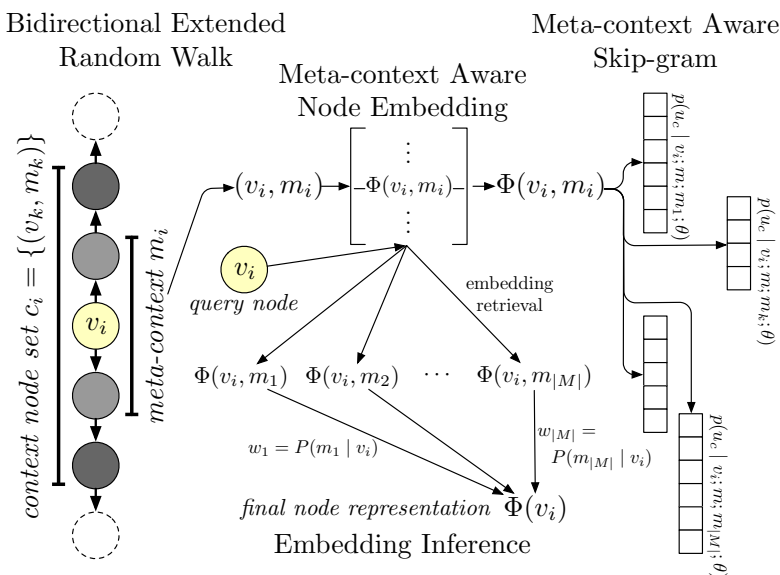


Figure 5.1: The schema of the proposed framework Meta-context Aware Random Walks (MARU).

Table 5.1: Summary of notations and their descriptions.

Notation	Descriptions
G	the heterogeneous network for learning representations
V	the set of nodes
E	the set of edges connecting nodes
T	the set of node types
$\psi(v)$	the function mapping a node v to the corresponding type
$N(v)$	the set of neighbors of the node v in the graph
d	the embedding dimension
l	the walk length for bidirectional extended random walk
k	the neighborhood size in the skip-gram model
w	the number of generated random walks per node
t	the meta-context size
r	the number of negative samples per neighbor
M	the set of available meta-contexts
$C(v, m, m_c)$	the context nodes with $m_c \in M$ for the node v with $m \in M$
$\Phi(v, m)$	the embedding of the node v with the meta-context m
$\Phi(v)$	the ultimate embedding of the node v

5.3.1 Framework Overview

Figure 5.1 depicts the general schema of MARU. More specifically, the model mainly consists of four stages, including bidirectional extended random walks, meta-context aware node embedding, meta-context aware skip-gram, and embedding inference. To efficiently and adequately capture the structural information, bidirectional extended random walks guarantee the generality of sampled structures and the fairness of context information for each node in random walks. To properly encode the heterogeneous knowledge, the stage of meta-context aware node embedding represents a node with different embedding vectors for distinct meta-contexts, which are the types of surrounding nodes on random walks. Based on the meta-context aware embeddings, the meta-context aware skip-gram model optimizes the representations by inferring not only the context nodes but also their meta-contexts. Finally, the ultimate representation of a node can be computed as an aggregation of meta-context aware embeddings over the estimated distribution of meta-contexts for the node in the stage of embedding inference. In sum, Table 5.1 summarizes the major notations and the corresponding descriptions.

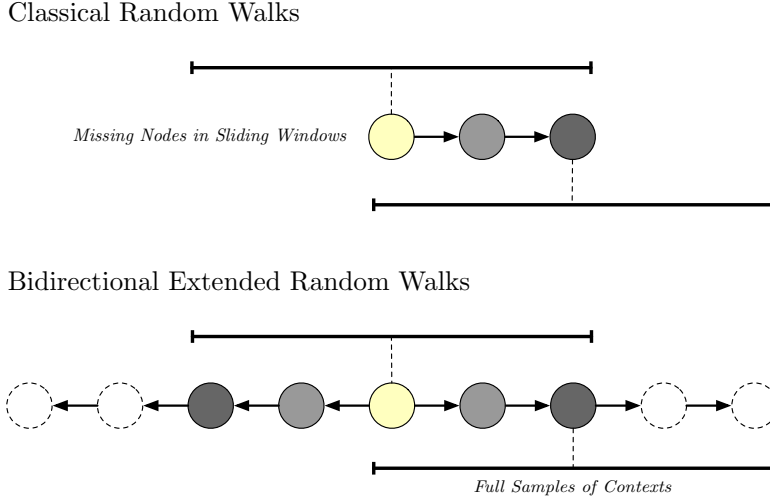


Figure 5.2: The illustrations of classical random walks and our proposed bidirectional extended random walks for learning network representations. The yellow nodes are the starting nodes of random walks while the white nodes with dotted strokes are the extended nodes. The lines are the sliding windows for the corresponding nodes for optimization.

5.3.2 Bidirectional Extended Random Walks

One of the most efficient approaches of capturing the network structures is to sample a few random walks that cover the network and then optimize the proximity between nodes within a sliding window on the random walks. However, classical random walks result in significant biases. More precisely, simple random walks would favor the nodes with high degree and betweenness scores, especially for the walks with longer lengths [89]. In addition, conventional random walks also lead to the bias of underestimating the contextual information of tail nodes while learning network representations. Figure 5.2 shows how classical random walks are applied to network representation learning. For the endpoints of random walks, there can be at most half of nodes that are missing in the sliding windows for deriving the contexts. Moreover, the most typical approach to optimize tail nodes is to start a number of random walks from them. In other words, the contexts for the tail nodes can be highly underestimated, and thus reveal incorrect structural information.

To address this problem, we propose the algorithm of bidirectional extended random walks as presented in Algorithm 5.1. Instead of walking through only a single direction, the starting node is treated as the center of the walk that grows from both sides simultaneously.

Algorithm 5.1: BidirectionalExtendedRandomWalk(G, u, l, k)

Input: the graph G , the starting node u , the walk length l , the neighborhood size k
Output: the bidirectional extended random walk L

- 1 $W = [u]$
- 2 $v_f = v_b = u$
- 3 **for** $iter = 1$ to $\lceil \frac{l-1}{2} \rceil + k$ **do**
- 4 $v_f = \text{RandomlySample}(N(v_f))$ // forward step.
- 5 $v_b = \text{RandomlySample}(N(v_b))$ // backward step.
- 6 $W = [v_b] + W + [v_f]$
- 7 **return** W

Furthermore, to secure the fairness of the observed contexts, the number of actual walking steps is extended according to the size of sliding windows in optimization. As shown in Figure 5.2, all of the nodes in the random walks can fairly have full samples of contexts for optimization. Moreover, bidirectional random walks can theoretically retrieve more tail nodes than one-directional random walks as shown in Corollary 5.1.

Corollary 5.1. Assume head nodes are never transitioning to tail nodes in random walks, and the probability of transitioning between tail nodes is $0 < p < 1$. Given a tail starting node u and the walking length $2n + 1$, the expected number of tail nodes in a bidirectional random walk is greater than the expected number in a one-directional random walk.

Note that we show the proof of Corollary 5.1 in Appendix B.1. As a result, the algorithm of bidirectional extended random walks is able to efficiently and fairly capture the structural information and provide enough knowledge for the optimization of network representation learning.

5.3.3 Meta-context Aware Skip-gram Model

To exploit the node types as heterogeneous information, most of the existing approaches guide the random walk through predefined meta-paths [98, 122] or specific strategies [164] before optimizing the proximity between nodes on random walks. However, these manipulations of random walks can distort the understanding of network structures. More specifically,

a portion of network structures can be ignored or inadequately covered by manipulated random walks. Hence, we do not guide random walks with any external knowledge. Instead, *meta-contexts* are taken into account to exploit heterogeneous knowledge.

Meta-contexts on Random Walks. Meta-contexts are defined as the node types within a sliding window. The motivation is that a node should have different contexts of nodes for different local meta-contexts. For example, if the meta-contexts for the node of an author in a bibliographic network are APAPA, the corresponding contexts should be the authored papers and the co-authors instead of the published venues. To some degree, meta-contexts can be treated as the conditions of the particular segments in random walks. The idea is beneficial for the model to learn the dynamic structures in the networks. Formally, given a random walk as $L = [v_1, v_2, \dots, v_{|L|}]$, the meta-contexts of the node v_i can be defined as:

$$m_i = (\psi(v_{i-t}), \dots, \psi(v_i), \dots, \psi(v_{i+t})),$$

where t is the window size for meta-contexts. For simplicity, we denote M as the set of all possible meta-contexts that can be found in the sampled random walks.

Meta-context Aware Node Embedding. To incorporate the knowledge of meta-contexts into the model, we propose the meta-context aware node embedding, which considers a node with different meta-contexts separately. More precisely, instead of learning a stationary representation $\Phi(v)$ for a node v , the node can have distinct representations $\Phi(v, m)$ for different meta-contexts $m \in M$. Note that although meta-contexts can be encoded independently with conditional bits [130] or individual embeddings [321], both of the methods perform unsatisfactorily in our experiments. This observation is mainly due to the sophisticated network structures of our framework. Independently learning representations with different meta-contexts for a node can better model heterogeneous networks.

Meta-context Aware Skip-gram. Similar to the previous studies [98, 135, 164, 264], we extend the skip-gram model originally proposed in the field of natural language processing [241] to learn network representations with the concept of meta-contexts. In addition

to the nearby nodes in random walks, we also optimize the likelihood of the corresponding meta-contexts for the context nodes. Given a heterogeneous network $G = (V, E, T)$, the objective of meta-context aware skip-gram model is to maximize the proximity between nodes in terms of local structures and meta-contexts as:

$$\arg \max_{\theta} \sum_{v \in V} \sum_{m \in M} \sum_{u_c \in C(v, m, m_c)} \log p(u_c | v; m; m_c; \theta),$$

where θ is the set of model parameters; $u_c \in C(v, m, m_c)$ denotes the context nodes u_c with specific meta-contexts m_c for the node v with the meta-contexts m . Different from conventional skip-gram models that output a single multinomial distribution of all available nodes, the meta-context aware skip-gram model learns multiple multinomial distributions for different meta-contexts. More specifically, as illustrated in Figure 5.1, the likelihood $p(u_c | v; m; m_c; \theta)$ can be estimated by the learned meta-context aware node embeddings and the softmax function [131] as:

$$p(u_c | v; m; m_c; \theta) = \frac{\Phi(v, m) \cdot \Phi(u_c, m_c)}{\sum_{\forall u_i \in V_{m_c}} \Phi(v, m) \cdot \Phi(u_i, m_c)},$$

where V_{m_c} is the set of nodes that have been associated with the meta-context m_c . During the training process, positive samples are generated by retrieving neighbors in the random walks with a length- k sliding window while a negative sample u_n can be randomly drawn from the distribution $P(u_n | m_c)$ for each neighbor. Therefore, the model can be optimized by using the stochastic gradient descent algorithm [274].

Embedding Inference. To generate the representations of individual nodes, the ultimate node embeddings can be further computed by aggregating the meta-context aware node embeddings as:

$$\Phi(v) = \sum_m P(m | v) \cdot \Phi(v, m), \text{ and } P(m | v) = \frac{\#(v, m)}{\sum_{m'} \#(v, m')},$$

where $\#(v, m)$ denotes the number of occurrences for the association of the node v and the

Algorithm 5.2: MetaContextAwareSkipGram(G, w, l, k, t, r, M)

Input: the graph $G = (V, E, T)$, the number of walks per node w , the walk length l , the neighborhood size k , the meta-context size t , the number of negative samples per neighbor r , the set of available meta-contexts M .

Output: the node representations $\Phi(v) : V \rightarrow \mathbb{R}^d$

```
1  $\Phi^{\text{meta}} = \Phi^{\text{node}} = \emptyset$ 
2 for  $iter\_w = 1$  to  $w$  do
3   for  $u \in V$  do
4      $W = \text{BidirectionalExtendedRandomWalk}(G, u, l, k)$ 
5     for  $i = k + 1$  to  $k + l$  do
6       for  $j = i - k$  to  $i + k$   $\mathcal{E} \ i \neq j$  do
7          $\Phi^{\text{meta}} = \text{SGD}(\Phi^{\text{meta}}, P(W_j | W_i; m_i; m_j; \theta) = 1)$ 
8         for  $iter\_n = 1$  to  $r$  do
9           Draw a negative sample  $u_n \sim P(u_n | m_j)$ 
10           $\Phi^{\text{meta}} =$ 
11           $\text{SGD}(\Phi^{\text{meta}}, P(u_n | W_i; m_i; m_j; \theta) = 0)$ 
12 for  $v \in V$  do
13    $\Phi^{\text{node}}(v) = \mathbf{0}$ 
14   for  $m \in M$  do
15      $\Phi^{\text{node}}(v) = \Phi^{\text{node}}(v) + P(m | v) \cdot \Phi^{\text{meta}}(v, m)$ 
16 return  $\Phi^{\text{node}}$ 
```

meta-context m in the training random walks. Finally, Algorithm 5.2 gives the pseudocode of the whole meta-context aware skip-gram model.

5.3.4 Complexity Analysis

Here we analyze the complexity of MARU.

For the time complexity, the bidirectional extended random walk algorithm spends $O(l + k)$ time to generate each random walk so that the overall time complexity for random walk generation is $O(w|V|(l + k))$. For each random walk, it costs $O(lkd \log(|V||M|))$ time to update the skip-gram model with negative sampling for learning meta-context aware node embeddings. Finally, the embedding inference takes $O(|V||M|)$ to derive the ultimate node

Table 5.2: The statistics of three experimental datasets of heterogeneous networks.

Dataset	Node Types and Number of Nodes			
DBIS [306] (264,323 edges)	Author (A) 60,694	Paper (P) 72,902	Venue (V) 464	
MovieLens [147] (1,097,495 edges)	Movie (M) 10,197	Actor (A) 95,321	Director (D) 4,060	User (U) 2,113
Yelp [56] (411,263 edges)	User (U) 16,239	Business (B) 14,284	Category (C) 511	Location (L) 47

embeddings. Therefore, the overall time complexity of MARU is:

$$O(wlkd|V|(\log |V| + \log |M|) + |V||M|).$$

For the space complexity, random walk generation requires $O(l + k)$ space as a buffer for the generated random walks. The meta-context aware node embeddings and ultimate node embeddings occupy $O(d|V||M|)$ and $O(d|V|)$ memory space while the skip-gram model has $O(d|M||V|)$ additional parameters. Hence, the overall space complexity of MARU is $O(l + k + d|M||V|)$.

5.4 Experiments

In this section, we conduct extensive experiments and in-depth analysis to verify the quality of learned heterogeneous network representations and the robustness of MARU in three general machine learning tasks.

5.4.1 Datasets and General Experimental Settings

Dataset. In the experiments, we adopt three large-scale publicly available heterogeneous network datasets, including DBIS [306], MovieLens [147], and Yelp [56]. Table 5.2 further shows the statistics of three datasets with more details as follows.

- **DBIS [306]** is a bibliographic network dataset in the field of database and information system. The network consists of papers (P), authors (A), and venues (V) as nodes while

the relationships of authorship (P-A) and published venues (P-V) are edges.

- **MovieLens** [147] is a network dataset of a movie recommendation system. The nodes of the network include movies (M), actors (A), directors (D), and users (U) while the edges comprise of actorship (M-A), directorship (M-D), and user ratings (M-U).
- **Yelp** [56] is a dataset extracted from the social media released in the competition of *Yelp Dataset Challenge* [56]. The nodes in the network involve users (U), businesses (B), categories (C), and locations (L) while the edges represent the relationships of friendships (U-U), user reviews (B-U), business locations (B-L), and business categories (B-C).

Baseline Methods. To evaluate the performance of MARU and the quality of learned representations, we compare MARU with five state-of-the-art homogeneous and heterogeneous network embedding methods as follows.

- **DeepWalk (DW)** [264] and **node2vec (N2V)** [135] represent random walk based homogeneous network embedding methods. DeepWalk generates a number of fixed-length plain random walks starting from each node while node2vec employs alias-sampling to mimic the process of breadth-first search and manipulate random walks. Both of the methods are based on the vanilla skip-gram model [241].
- **LINE** [313] represents an edge-sampling based homogeneous network embedding method. Based on the edge-sampling algorithm, LINE is able to efficiently capture both the first-order and second-order proximity in the networks.
- **HIN2Vec (H2V)** [122] learns node embeddings by predicting the existence of particular meta-paths between nodes with a meta-path conditioned binary classifier.
- **metapath2vec (M2V)** [98] stands for meta-path based heterogeneous network embedding methods. With a predefined set of meta-paths, metapath2vec guides the random walks through meta-paths so that the prior heterogeneous knowledge can be leveraged to the learned embeddings.
- **JUST** [164] is a heterogeneous network embedding method that manipulates random walks by specific strategies. JUST introduces a tactic for random walks to either jump to other nodes of particular types or to stay on the current paths.

- **HeGAN** [158] enhances HIN by adversarial learning that provides effective negative examples for more robust representations.

Note that we do not compare with GCN-based approaches because most of those methods cannot tackle unsupervised representation learning. Although some methods like GraphSAGE [139] and GAE [199] are applicable, they heavily rely on node features are not in major comparisons as shown in previous studies [109] For instance, the macro-F1 scores of both GraphSAGE and GAE are less than 23% on the Yelp dataset when all of the other baseline methods can reach over 30% with an arbitrary amount of training data.

Implementation Details. MARU is implemented by C and C++. The size of sliding windows for meta-contexts t is set as 6. The walk length l in the algorithms is 40 while the length of each generated random walks is 81. For all of the methods, the dimension of node embeddings is set to 128; the neighborhood size k is set as 7; the initial learning rate of stochastic gradient descent is set as 0.025; the number of negative samples for each neighbor r is 5.

5.4.2 Task 1: Multi-label Node Classification

Experimental Setup. In the task of multi-label node classification, every node is associated with one or more labels from a finite label set \mathbb{L} . We adopt the author domains, movie genres, and user compliments respectively for the DBIS, MovieLens, and Yelp datasets. The statistics of these datasets are shown in Table 5.3. Moreover, the labels are encoded in the networks so that the task is challenging because the node embeddings need to reflect the semantics that is not explicitly presented in the networks. To evaluate the performance, we randomly sample 10% of the nodes as testing data while the remaining nodes are treated as labeled data for training. In addition, we also adjust the percentage of labeled data used in the training process to demonstrate the robustness of methods. The node representations of each method are treated as the input of a one-vs-rest logistic regression model with L2 regularization. Macro-F1 and Micro-F1 scores [269] are adopted as the evaluation metrics for multi-label classification, thereby indicating the quality of different representations.

Table 5.3: The statistics of three datasets for the task of multi-label node classification.

Dataset	DBIS	MovieLens	Yelp
Node Type	Author (A)	Movie (M)	User (U)
Semantics	Domains	Genres	Compliments
$ \mathcal{L} $	8	19	11
Avg. #(labels)	1.00	2.04	5.33

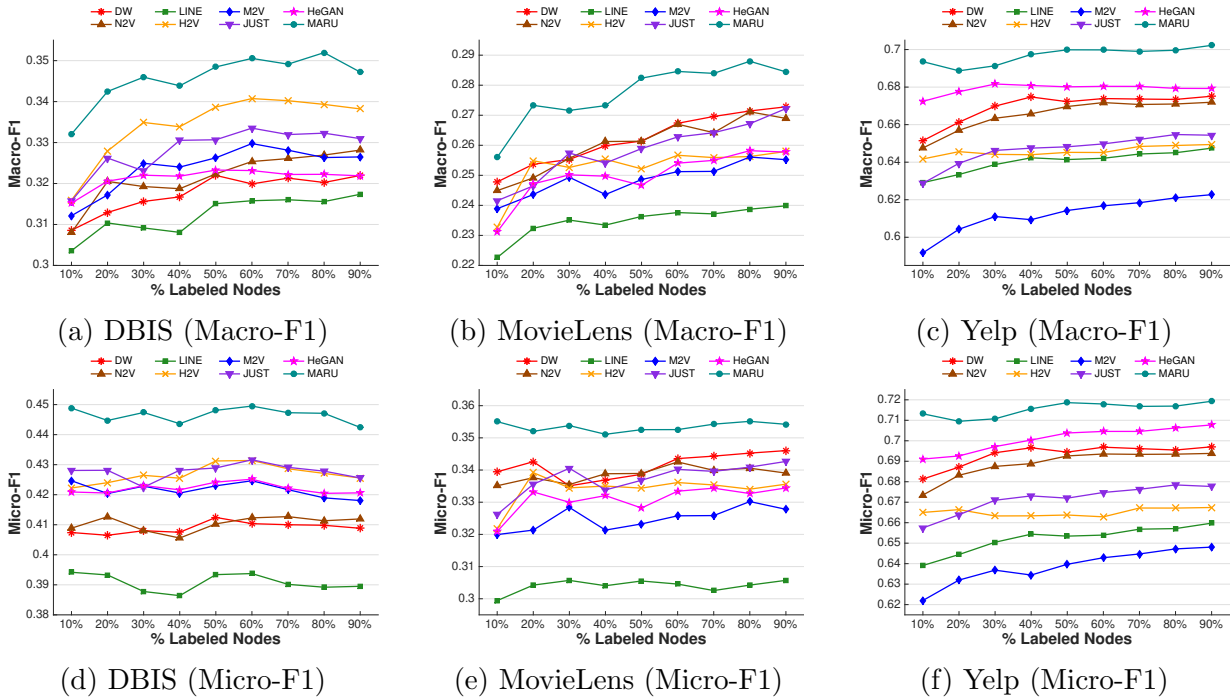


Figure 5.3: Performance of different methods for the multi-label node classification task in three datasets. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test. Note that the Micro-F1 scores do not increase with more labeled nodes in some cases because of the imbalance of class distribution.

Experimental Results. Figure 5.3 demonstrates the performance of six methods on the task of multi-label node classification with three datasets. Among all of the baseline methods, most of the heterogeneous network embedding methods, including H2V, M2V, and JUST, outperform the other baselines in DBIS but perform worse than others in Yelp. It can be because the structural information is more important than the heterogeneous knowledge in Yelp. To be more precise, existing heterogeneous network embedding methods sacrifice the comprehensive understanding of network structures to encode the heterogeneous knowledge

and obtain unsatisfactory performance when the structural information is imperative. Although HeGAN applies adversarial learning to obtain better robustness in Yelp, it performs worse in both DBIS and MovieLens due to more parameters and overfitting. Our approach MARU significantly outperforms all of the baselines across different percentages of training labeled nodes in three datasets. MARU does not distort the generated random walks while incorporating heterogeneous knowledge. At the same time, meta-contexts are also beneficial for MARU as it picks up the tiny differences in local heterogeneous contexts.

5.4.3 Taks 2: Node Clustering

Experimental Setup. The problem of node clustering is an unsupervised machine learning task. We aim to cluster the nodes so that the generated groups are as close to the true clusters as possible. In each dataset, we modify the classes in multi-label classification to construct the ground truth. For the DBIS dataset, the authors can be categorized into different research domains. Each research domain represents one type of cluster. For the MovieLens dataset, five genres, including **Adventure**, **Action**, **Crime**, **Horror**, and **Sci-Fi**, represent five clusters. For the Yelp dataset, we separate users into two groups. One group represents those users who have received at least one compliment. The rest of the users are labeled otherwise. For simplicity, the nodes in multiple clusters are removed. In total, DBIS, MovieLens, and Yelp datasets have 8, 5, and 2 clusters, respectively. For evaluation, the node representations of each method are treated as the input of the K-Means++ algorithm [14] to derive clusters. Finally, normalized mutual information (NMI) and Adjusted Mutual Information (AMI) [324] are the evaluation metrics that reveal the quality of node representations.

Experimental Results. Figure 5.4 illustrates the performance of different methods for the task of node clustering in three datasets. Similar to the results in the multi-label classification task, H2V and M2V perform the best among all of the baselines in DBIS but obtain worse performance than others in Yelp. Differently, JUST and HeGAN perform reasonably well on all datasets. On the other hand, the homogeneous network embedding methods

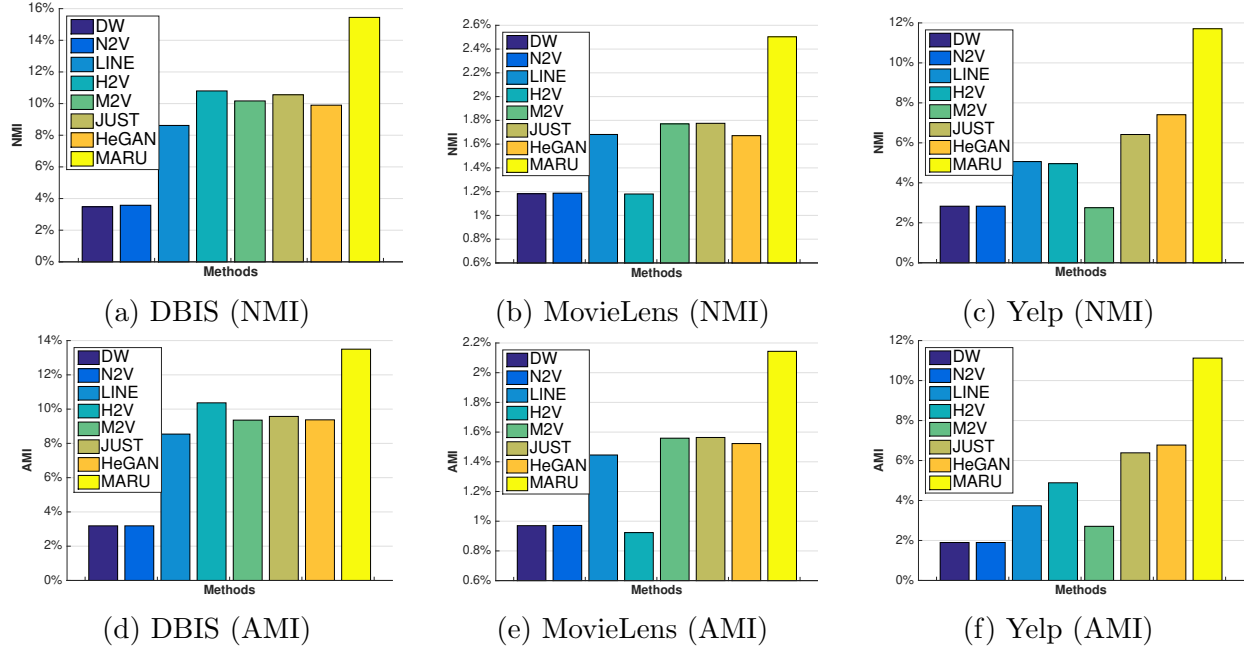


Figure 5.4: Performance of different methods for the node clustering task in three datasets. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.

perform poorly in all of the datasets. One explanation is that the heterogeneous knowledge is important for the task of clustering. Interestingly, even though M2V exploits the heterogeneous knowledge by using the meta-paths, the clustering performance significantly drops in Yelp compared to other datasets. A possible reason could be the lack of meaningful meta-paths for clustering in the Yelp network. On the other hand, JUST does not need meta-paths and still performs well. Compared to all of the baseline methods, our proposed MARU consistently presents significant improvements against all baseline methods across all datasets. As a result, it demonstrates that meta-contexts and the algorithm of bidirectional extended random walks are valuable for the node clustering task.

5.4.4 Task 3: Link Prediction

Experimental Setup. In the task of link prediction, we predict the missing edges in the given network datasets. Here we randomly remove 50% of edges from the networks for obtaining positive examples while generating an equal number of node pairs as negative

Table 5.4: The *AUC* scores of different methods with four operators for link prediction in three datasets.

Method	Operator	DBIS	MOVIE	YELP
DeepWalk [264]	Hadamard	0.6367	0.9110	0.7330
	Weighted-L2	0.6094	0.7904	0.6872
node2vec [135]	Hadamard	0.6362	0.9060	0.6622
	Weighted-L2	0.6292	0.7968	0.6848
LINE [313]	Hadamard	0.5001	0.8631	0.5689
	Weighted-L2	0.5751	0.7611	0.6229
HIN2Vec [122]	Hadamard	0.8028	0.9651	0.8117
	Weighted-L2	0.7240	0.7885	0.7137
metapath2vec [98]	Hadamard	0.6778	0.9151	0.7372
	Weighted-L2	0.7363	0.6996	0.8240
JUST [164]	Hadamard	0.6463	0.9119	0.7453
	Weighted-L2	0.6260	0.7845	0.6009
HeGAN [158]	Hadamard	0.9597	0.9207	0.6361
	Weighted-L2	0.6714	0.7970	0.7289
MARU	Hadamard	0.9979	0.9963	0.7241
	Weighted-L2	0.7468	0.7979	0.8315

examples. To generate the edge features, we follow the previous study [135] to exploit two binary operators to represent edges by aggregating two node representations over all dimensions, including the Hadamard product and weighted L2-distance. The features of example edges are treated as the input of a logistic regression model to learn their existence. Finally, the scores of Area Under Curve (AUC) can be applied to evaluate the performance of link prediction and the quality of representations.

Experimental Results. Table 5.4 shows the performance of different methods for the task of link prediction in three datasets. In the task of link prediction, our proposed approach MARU significantly surpasses all of the baseline methods. Among the baseline methods, HIN2Vec and metapath2vec perform the best as heterogeneous network embedding methods. Interestingly, although LINE does not have outstanding performances in the tasks of multi-label node classification and node clustering, it has a satisfactory performance for link prediction. It can be because LINE is an edge-sampling based method so that it has more advantage in link prediction to model the edge distributions. Interestingly, Grover and Leskovec [135] report that the Hadamard operator always performs the best in their study

while only the datasets with homogeneous networks are evaluated. This is partially inconsistent with the experimental results of heterogeneous networks. The reason can be that the embeddings become too sophisticated to estimate the relationship between nodes by a simple dot-product when the types of nodes are heterogeneous. The results also show the difference between homogeneous and heterogeneous network and emphasize the importance of designing satisfactory algorithms to derive heterogeneous network representations.

5.4.5 Analysis and Discussions

In this section, we first analyze the effectiveness of the proposed algorithm of bidirectional extended random walks and then discuss the sensitivity of the window size for observing meta-contexts.

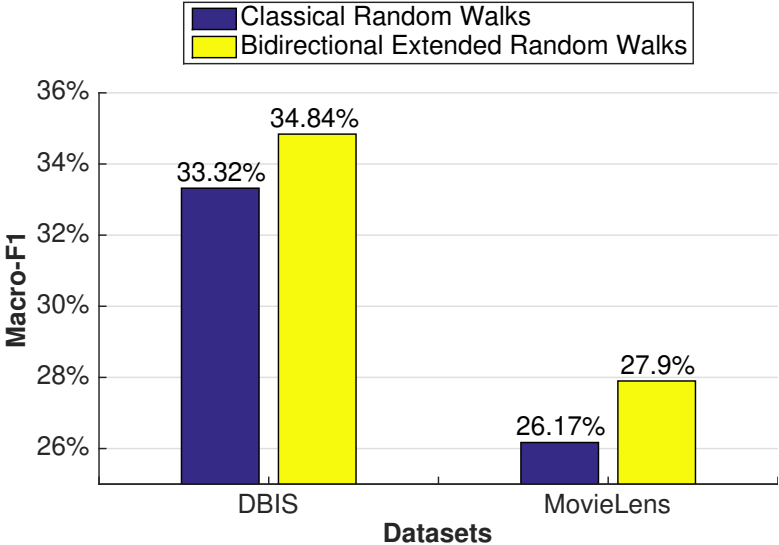


Figure 5.5: The macro-F1 scores of MARU with classical random walks and our proposed bidirectional extended random walks with 50% of training labeled nodes in the task of multi-label node classification in DBIS and MovieLens.

Effectiveness of Bidirectional Extended Random Walks. To verify the contribution of our proposed bidirectional extended random walks, we first investigate the effectiveness of the algorithm. Figure 5.5 shows the macro-F1 scores of MARU with classical random walks and the proposed bidirectional extended random walks with 50% of training labeled nodes in the task of multi-label node classification in DBIS and MovieLens. After replacing the classical

random walks with the bidirectional extended random walks, the classification performances are significantly improved by 2.04% and 4.07% in DBIS and MovieLens, respectively. It shows that the proposed algorithm to generate bidirectional extended random walks is actually beneficial to alleviate the insensitivity of classical random walks to the tail nodes, thereby improving the performance of downstream applications.

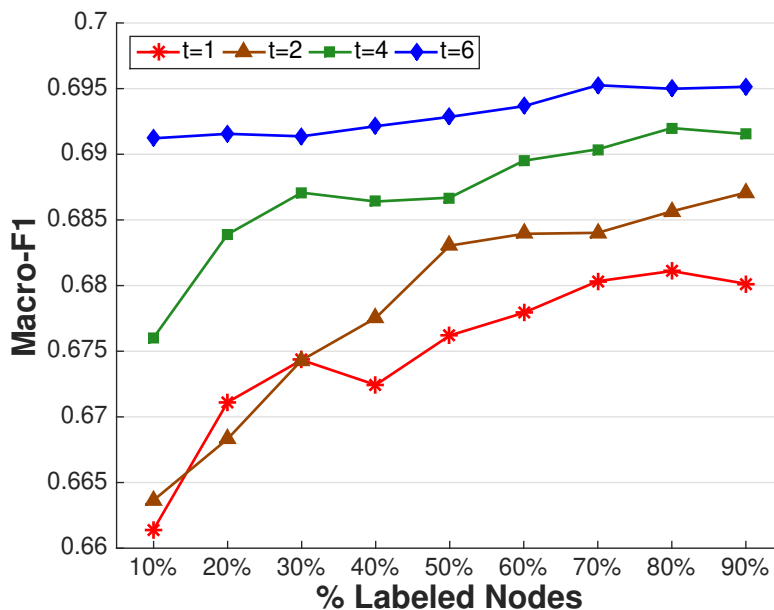


Figure 5.6: The Macro-F1 scores of MARU as a function of percentage of labeled training data and sliding window size for meta-contexts in Yelp.

Window Size of Meta-contexts. Here we study how the size of the sliding windows for meta-contexts affects the performance. Figure 5.6 shows the macro-F1 scores of MARU over different percentages of labeled training data with different window sizes for meta-contexts in the Yelp dataset. It is obvious that greater window sizes lead to a better classification performance because the observed contexts are more flexible and informative. However, larger window sizes also lead to larger body of meta-contexts M . For example, in the Yelp dataset with $t = 6$, the size of M is greater than 10,000, which can significantly increase the memory or disk space consumption. On the other hand, the size of M is less than 1,000 with $t = 4$, rendering memory footprints more manageable. Therefore, we set the window size t as 4 in the parameter settings.

Walk Length l of Bidirectional Extended Random Walks. Here we study how the

length of random walks affects the performance. Table 5.5 presents the classification performance of MARU over different walk lengths l of bidirectional extended random walks (See Algorithm 5.1 and 5.2) with 50% training labeled nodes in the Yelp dataset. While the length of random walks increases, both micro-F1 and macro-F1 scores improve because of more prevalent information. However, the performance peaks at $l = 40$ and then drops with longer random walks. This can be because longer random walks cover more nodes with high scores of degree and betweenness so that the contexts with tail nodes are less observed in the generated random walks. The results also demonstrate that it is important to design a good algorithm, such as the proposed bidirectional extended random walk, to alleviate the bias of conventional random walk algorithms.

Table 5.5: The classification performance of MARU over different walk lengths l of bidirectional extended random walks with 50% of training labeled nodes in Yelp. Note that the length of generated random walks is $2 \times l + 1$ because MARU conducts random walks bidirectionally.

Metric	$l = 10$	$l = 20$	$l = 40$	$l = 80$	$l = 100$
Macro-F1	0.6842	0.6961	0.6999	0.6958	0.6969
Micro-F1	0.6998	0.7139	0.7186	0.7150	0.7151

Size of Embedding Dimensions. We also discuss how the size of embedding dimensions affects the performance. Table 5.6 shows the classification performance of MARU over different sizes of embedding dimensions d with 50% of training labeled nodes in the Yelp dataset. When the dimension increases, the performance improves and peaks at 128. With a larger size of embedding dimensions, the classification model becomes overfitted. As a result, we apply $d = 128$ as the experimental setting across all experiments.

5.5 Conclusion

In this chapter, we propose MARU, a novel approach for heterogeneous network embedding by exploiting meta-contexts in random walks. To address the bias caused by conventional random walks, the algorithm of bidirectional extended random walks is proposed to efficiently and fairly capture the comprehensive structural information in the networks. The

Table 5.6: The classification performance of MARU over different sizes of embedding dimensions d with 50% of training labeled nodes in Yelp.

Metric	$d = 16$	$d = 32$	$d = 64$	$d = 128$	$d = 256$
Macro-F1	0.6931	0.6933	0.6982	0.6999	0.6974
Micro-F1	0.7112	0.7114	0.7154	0.7186	0.7156

meta-context aware node embeddings are then designed and optimized to represent properties of the nodes for different local heterogeneous contexts, thereby inferring the node representations based on aggregations over the meta-context distributions. Extensive experiments demonstrate that our proposed approach significantly outperforms state-of-the-art heterogeneous network embedding methods across three general network mining tasks, including multi-label node classification, node clustering, and link prediction. The reasons and insights can be concluded as follows: (1) the algorithm of bidirectional extended random walks effectively alleviates the bias for tail nodes with a theoretical guarantee; (2) the effectiveness of meta-contexts and meta-context aware node embeddings implies that a node can have distinct properties with different local heterogeneous contexts, which benefit the network representation learning; (3) the nature of heterogeneous networks can be much different from the traits of homogeneous networks, so it is crucial to tackle the problems of heterogeneous networks with specific and appropriate technologies.

CHAPTER 6

Learning User Coresets to Accelerate Large-scale Top-K Recommender Systems

Although many machine learning models can be trained within a short time based on some tactics like negative sampling, they usually suffer from the stage of inferring predictions because of the need of examining all candidates for high accuracy. To overcome this efficiency bottleneck, it is essential to develop a robust approximation method without losing precision for the inference stage of machine learning models. In this chapter, we focus on accelerating the inference stage of latent vector models for top- K recommender systems. By leveraging the structure of clustered user affinity groups, we propose to discover a coreset of users to construct a preferred item set, thereby significantly reducing the number of ranking candidates and speeding up the inference.

6.1 Introduction

Building large-scale personalized recommender systems has already become a core problem in many online applications since the explosive growth of internet users in the recent decade. For example, user-item recommender systems achieve many successes in e-commerce markets [220] while link prediction in social networks can be treated as a variant of recommender systems [21, 314]. To establish recommender systems, latent factor models for collaborative filtering have become popular because of their effectiveness and simplicity. More precisely, each user or item can be represented as a low-dimensional vector in a latent space so that the inner products between user and item vectors are capable of indicating the user-item prefer-

ences. Furthermore, these latent vectors can then be learned by optimizing a loss function with sufficient training data. For instance, matrix factorization [203] has been empirically shown to outperform conventional nearest-neighbor based approaches in a wide range of application domains [100].

After obtaining user and item latent vectors, to make item recommendations for each user, recommender systems need to calculate the inner products for all user-item pairs. Although learning user and item latent vectors is efficient and scalable for most existing models, recommender systems can take an enormous amount of time in evaluating all user-item pairs. More specifically, the time complexity of learning latent vectors is only proportional to the number of user-item pairs in the training data which is a small subset of all possible user-item pairs, but finding the top recommendations entails examining all $O(mn)$ inner products between all m users and n items. As a result, the quadratic complexity becomes a hurdle for large-scale recommender systems. For example, it can take more than a day to compute and rank all preference scores, and consequently the systems cannot be updated on a daily basis [102]. In order to make large-scale recommender systems practical, it is critical to accelerate the process of computing and ranking the inner products of user and item latent vectors, in order to efficiently obtain the top- K recommendations for all users.

To accelerate the computation of inner products, the maximum inner product search (MIPS) [249, 295, 359] is one of the feasible approaches. Locality sensitive hashing (LSH) [167] and PCA tree [299] may be applied to solve MIPS after reducing the problem to nearest-neighbor search. To reduce the computation for making recommendations for a given user, one may find a small group of candidate items whose latent vectors have large inner products with the user’s latent vector using clustering algorithms [61], or sort entries of each dimension in the latent vectors separately by some greedy algorithms [102, 359]. In essence, most of the existing MIPS algorithms adopt a two-stage strategy, decomposing the computation into a preparation process and a prediction process. In the preparation stage, these methods will construct suitable data structures [359] or reduce the number of ranking candidates [61], and these prepared data structures are used to conduct efficient maximum inner product search

for query vectors in the inference stage. However, most of these existing MIPS algorithms have the following two weaknesses, making them often impractical for real applications: (1) they only focus on optimizing the inference speed for a given user at the cost of considerable preparation time, but for recommender systems, the overall execution time (including both preparation and inference time) matters more because the system needs to be re-trained frequently as new data arrive. (2) All the MIPS approaches aim to quickly identify the top item set for *any* query vector. However, in recommender systems queries are not arbitrary vectors. They are user latent factors and usually have very strong *clustering structure*, which is ignored in most of the MIPS algorithms.

In order to speed up the overall execution time, our main idea is to exploit the relationships between users. More precisely, users with similar latent factors are more likely to share similar item preferences which may be reflected by their high inner products. However, existing methods for accelerating recommender systems do not consider user relationships and the distribution of user latent vectors. For instance, existing greedy strategies [102, 359] only consider the values of item latent vectors. Some studies based on proximity graphs [226, 366] and clustering algorithms [61] also solely reduce the search space of items. In the inference stage, these approaches treat the recommendation to each user as an independent query to the data structures and algorithms. As a consequence, it can be extremely time-consuming, especially with myriad users and enormous spaces of candidate items.

We propose a novel model for clustering and navigating for top- K recommenders (CANTOR) that leverages the knowledge of user relationships to accelerate the process of generating recommendations for all users with a given latent factor model. CANTOR consists of two stages: preparation and prediction. In the preparation stage, we aim to cluster users sharing similar interests into affinity groups and compute a small set of preferred items for each affinity group. More specifically, the user vectors (generated from a given latent factor model) are used in clustering affinity groups. To further accelerate the preparation time, a user coreset of few representative vectors are derived for each affinity group, and are used to obtain a small set of preferred items for users in this group by an efficient approximate nearest neigh-

bor search algorithm. Finally, in the prediction stage, the top- K recommendations for each user can be retrieved by ranking these preferred items of the corresponding affinity group, which can be done much more efficiently than evaluating and ranking all items.

Our contributions are three-fold: (1) To the best of our knowledge, this is the first work to focus on the preparation time and user relationships for accelerating the prediction process of large-scale top- K recommender systems. (2) Clustering users into affinity groups based on the distribution of user latent vectors provides significant speedup of the prediction process, compared to conventional approaches that independently deal with each user. The representative vectors of the affinity groups offer a theoretically guaranteed precision for users with similar preferences. Approximate nearest neighbor search is applied to efficiently retrieve the satisfactory recommendations for each user from a small set of candidate items. (3) Experiments conducted on six publicly available datasets demonstrate that CANTOR can significantly accelerate large-scale top- K recommender systems for both item recommendation and personalized link prediction. An in-depth analysis then indicates the robustness and effectiveness of the proposed framework.

6.2 Problem Statement

In this section, we first introduce the notations and then formally define the objective of this work. Suppose that we have an incomplete $m \times n$ one-class matrix $\mathbb{R} = \{R_{ij}\} \in \{0, 1\}^{m \times n}$, where m and n are the numbers of users and items in the system. $R_{ij} = 1$ if user i prefers item j in the training data; otherwise, $R_{ij} = 0$. Based on \mathbb{R} , a matrix factorization based algorithm learns d -dimensional user and item latent vectors, denoted by $\mathbb{P} \in \mathbb{R}^{m \times d}$ and $\mathbb{Q} \in \mathbb{R}^{n \times d}$ respectively, where $\hat{\mathbb{R}} = \mathbb{P}\mathbb{Q}^T \in \mathbb{R}^{m \times n}$ reflects the underlying preferences. To compute top- K recommendations for each user, we need to find items with the K highest scores among $\hat{\mathbf{R}}(i) = \{\hat{R}_{ij'} \mid j' \in 1 \dots m\}$. Note that $m = n$ for personalized link prediction in social networks, where the goal is to suggest other users as recommended items.

Although matrix factorization models can be learned expeditiously when \mathbb{R} is sparse,

inferring the top- K recommendations requires computing and sorting the scores \hat{R}_{ij} of all items j for each user i . As a result, the inference process can be time-consuming with an $O(nmd)$ time complexity which becomes intractable when n and m are large. To address this problem, our goal is to speed up the inference time of top- K recommenders with a high precision. More specifically, given the trained matrices \mathbb{P} and \mathbb{Q} , we aim to propose an efficient approach that approximates the top- K recommended items for each user.

6.3 Constructing User Coresets for Top-K Recommender Systems

In this section, we present CANTOR for accelerating top- K recommender systems, starting with several key preliminary ideas.

6.3.1 Preliminary

In order to leverage the relationship between users, we first formally define the *affinity groups* of users in recommender systems as follows:

Definition 6.1. (Affinity Group) An *affinity group* \mathbb{A}_t is a set of users sharing similar interests in items. Even though any similarity metrics may be used, we adopt cosine similarity as the metric to define the affinity groups.

By this definition, the sets of satisfactory recommendations should be similar for users in the same affinity group. This suggests that the top recommendations for all users in an affinity group are confined to a small subset of the items and such item subset can be learned by examining only a few carefully selected users in the group, leading to the following definition of the *preferred item set*.

Definition 6.2. (Preferred Item Set) A *preferred item set* c for an affinity group is a set of (potentially) satisfactory items for the users in the group, and the size of the preferred item set is usually much smaller than the total number of items, i.e., $|c| \ll n$.

Therefore, we only need to examine the preferred item set to generate top recommendations,

leading to significant time saving overs the alternative of examining all items.

In order to robustly generate the preferred item set for each affinity group, we generate a few representatives from the group to compute the preferred item set. This is statistically more robust than using only the “centroid” user in the latent space, and is more computationally efficient than using all users in the group.

Definition 6.3. (User Coreset of an Affinity Group) A δ -user coreset s_t of an affinity group \mathbb{A}_t is a (small) set of latent representative vectors to preserve the item preference of the users in \mathbb{A}_t such that $\forall q \in \mathbb{Q}, i \in \mathbb{A}_t$:

$$|p_i q^T - \mathcal{N}_{s_t}(p_i) q^T| \leq \delta,$$

where $\mathcal{N}_{s_t}(p_i) \in s_t$ is the nearest coreset representative for p_i ; $\delta > 0$ is a small enough constant.

The user interests in the affinity group can be well captured by the representative vectors in the user coreset. Note that the representative vectors do not have to be identical to actual user latent vectors in the group.

6.3.2 Framework Overview

Figure 6.1 shows the general framework of CANTOR. The framework consists of two stages: preparation and prediction. In the preparation stage as shown in Algorithm 6.1, the m user latent vectors \mathbb{P} are first sub-sampled as $\hat{\mathbb{P}}$ and clustered into r affinity groups \mathbb{A}_t with a centroid vector v_t computed from the corresponding user vectors \mathbb{P}_t , where $t = 1 \dots r$. For each affinity group \mathbb{A}_t , we aim at deriving a small user coreset s_t . To do so, we propose an adaptive representative selection method (Algorithm 6.2) to greedily construct a set cover of user latent vectors for each affinity group after mathematically proving that the set covers can be the coresets of affinity groups. Finally, a small preferred item set c_t can be obtained by approximate nearest neighbor search using its coreset s_t for each affinity group. In the prediction stage (Algorithm 6.4), CANTOR first locates the corresponding affinity group \mathbb{A}_t

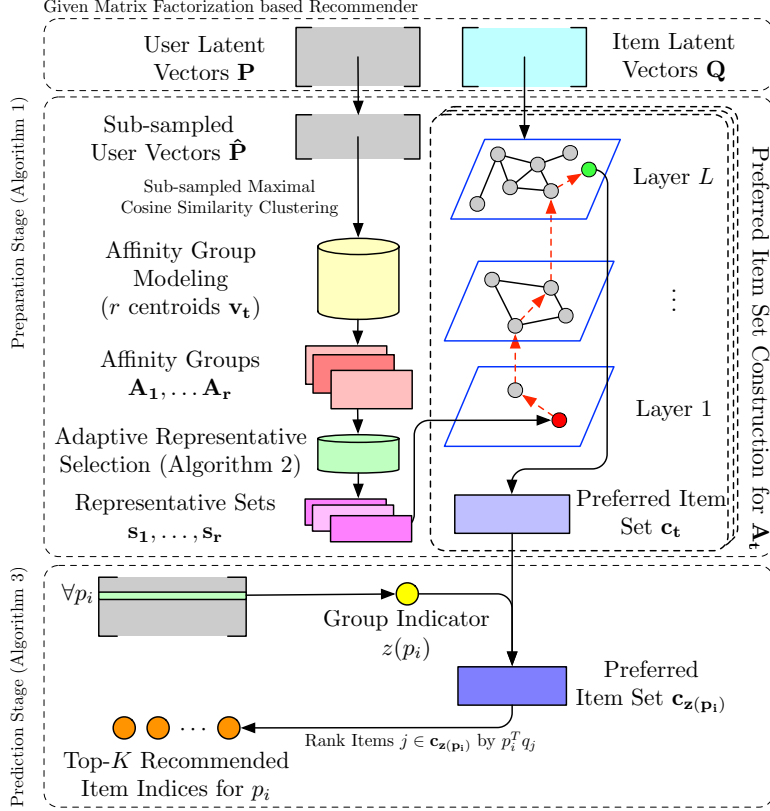


Figure 6.1: The general framework of the proposed clustering and navigating for top-K recommenders (CANTOR).

for each user and then ranks the small number of items in the preferred item set \mathbf{c}_t , thereby efficiently providing satisfactory recommendations.

6.3.3 Preparation Stage

To overcome the hurdle of extremely long preparation time experienced by conventional methods, we propose to exploit similarities between user vectors in the latent space for acceleration as shown in Algorithm 6.1.

Affinity Group Modeling by User Clustering. Most of the conventional algorithms only rely on similarities of item latent vectors [293] and proximity graphs [134, 366] to accelerate the recommendation, and have not used the relationships between users and the distributions of user latent vectors in this endeavor. To exploit the knowledge of user relationships, we propose a clustering based framework to model user affinity groups.

Algorithm 6.1: Preparation Process for CANTOR

Input: User latent vectors \mathbb{P} ; item latent vectors \mathbb{Q} ; degree of each user $deg_{i=1}^m$; the number of desired recommendation K

Output: Centroid vectors v_t and preferred item sets c_t for each affinity cluster \mathbb{A}_t for $t = 1 \dots r$.

- 1 **Hyperparameter:** Number of affinity groups r ; small world graph search size efs ; number of sub-sampled users u ;
- 2 $\hat{P} = \text{MultinomialSampling}(\mathbb{P}, deg_{i=1}^m, u)$; $\hat{P} \in \mathbb{R}^{u \times d}$;
- 3 $v_1, \dots, v_r = 0$; $I = 0, I \in \mathbb{R}^{u \times 1}$;
- 4 **repeat**
- 5 **for** $i = 1, \dots, r$ **do**
- 6 $v_i = \sum_{j \in \{j | I[j]=i\}} \hat{P}[j]$;
- 7 $v_i = v_i / \|v_i\|_2$;
- 8 $I = \arg \max_t v_t^T \hat{P}$;
- 9 **until** *Convergence*;
- 10 $G = \text{CreateProximityGraph}(\mathbb{Q}, efs)$;
- 11 $c_1, \dots, c_r = \emptyset, \dots, \emptyset$;
- 12 $I = \arg \max_t v_t^T \hat{P}$;
- 13 **for** $i = 1, \dots, r$ **do**
- 14 $\hat{\mathbb{P}}_i = \{p_j \mid p_j \in \hat{\mathbb{P}}, I[j] = i\}$;
- 15 $s_i = \text{AdaptiveClustering}(\hat{\mathbb{P}}_i, \epsilon, w)$;
- 16 **for** $q \in s_i$ **do**
- 17 $\hat{I}_i = \text{QueryProximityGraph}(G, s, K)$;
- 18 $c_i = c_t \cup \hat{I}_i$;
- 19 **return** c_t, v_t for all $t = 1, \dots, r$.

Let r be the number of affinity groups for all m users, where r is a hyperparameter in CANTOR. We partition all m users into r disjoint clusters as the affinity groups $\mathbb{A} = \{\mathbb{A}_t \mid t = 1 \dots r\}$ based on the user latent vectors $\mathbb{P} = \{p_i \mid i = 1 \dots m\}$. In addition, each affinity group \mathbb{A}_t has a centroid vector $v_t \in \mathbb{R}^d$ in the latent space. Each user i with the latent vector p_i belongs to $\mathbb{A}_{z(p_i)}$, where $z(p_i)$ is the affinity group indicator represented as:

$$z(p_i) = \arg \max_r \sum_r^T p_i. \quad (6.1)$$

Let $C(p_i, K)$ be the top- K preferred items for user i which is defined as:

$$\{j \mid p_i^T q_j \geq p_i^T q_{j'}, \forall j' \notin C(p_i, K) \text{ and } |C(p_i, K)| = K\},$$

where $q_j \in \mathbb{Q}$ is the latent vector of item j . Intuitively, if users i and k are in the same affinity group, their preferred sets $C(p_i, K)$ and $C(p_k, K)$ may have substantial overlap because of their similar interests. This motivates us to compute a preferred item set c_t for users in the same affinity group \mathbb{A}_t so that each c_t contains only a small subset of all n items, i.e., $|c_t| \ll n$. Instead of computing the inner products between p_k and all item latent factors $q \in \mathbb{Q}$, we can narrow down the candidate set to be c_t , and only evaluate the items in c_t to find the top- K predictions for user k .

Since our task is to accelerate the maximum inner product search, the centroid vector \mathbf{v}_t for each affinity group \mathbb{A}_t can then be updated by the maximum cosine similarity criteria as:

$$\mathbf{v}_t = \frac{\sum_{i=1}^{|\mathbb{P}_t|} \mathbb{P}_{ti}}{\|\sum_{i=1}^{|\mathbb{P}_t|} \mathbb{P}_{ti}\|_2}, \quad (6.2)$$

where $\mathbb{P}_t = \{p_i \mid z(p_i) = t\}$ contains the latent vectors of users that belong to the affinity group \mathbb{A}_t . Therefore, each affinity group \mathbb{A}_t can obtain a centroid vector \mathbf{v}_t by iteratively running Equations (6.1) and (6.2). However, iteratively performing Equations (6.1) and (6.2) can still cost a long computational time when the number of users m is large. To address this issue, we propose to sub-sample a portion of the m user latent vectors to learn the centroid vectors. Moreover, we sample the latent vectors based on the degree distribution in the one-class matrix \mathbb{R} . For example, Figure 6.2a shows that degree distribution of users usually follows a power-law distribution. Hence, instead of using a uniform sampling, we sample user i with a probability proportional to a log function of its degree as:

$$P(X = i) \propto \log \sum_{j=1}^n R_{ij}, \quad (6.3)$$

where X denotes the random variable of the target sampling process. We will later show

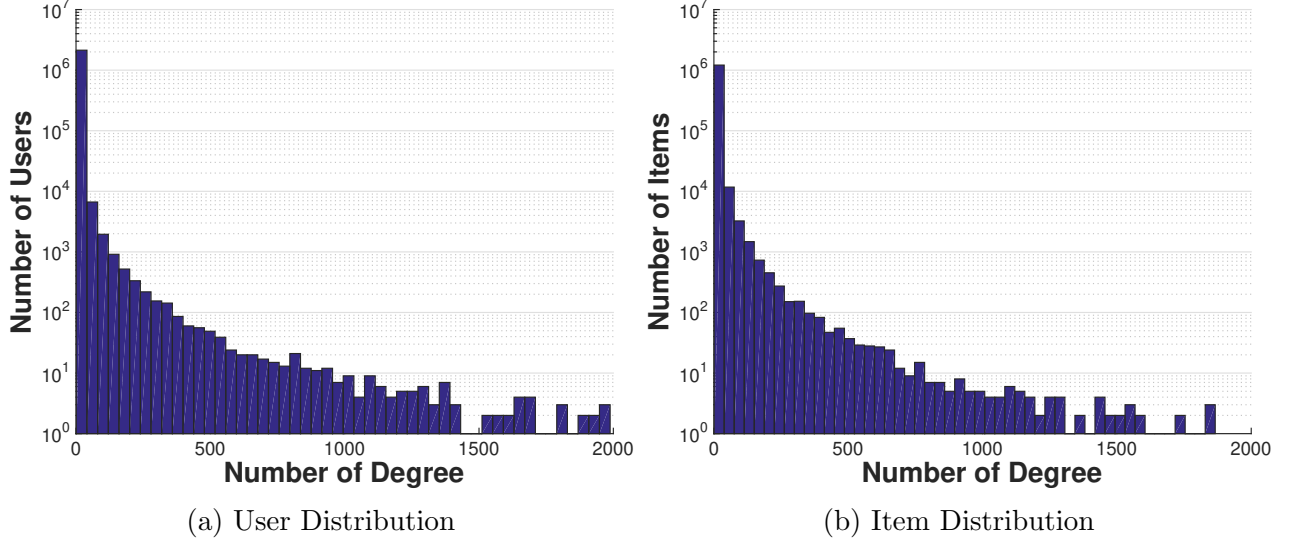


Figure 6.2: The distributions of users and items over different degrees in the Amazon dataset.

in Theorem 6.2 that error of approximation based on sub-sampling will be asymptotically bounded.

After learning the centroids $v_1, \dots, v_r \in \mathbb{R}^d$ and the corresponding user latent vectors $\mathbb{P}_1, \dots, \mathbb{P}_r$ for r affinity groups $\mathbb{A}_1, \dots, \mathbb{A}_r$, the preferred item set c_t for each group \mathbb{A}_t can be constructed so that user vectors \mathbb{P}_t only need to search over this set of preferred items for top recommendations. However, the naïve approach to generate c_t would require $O(nd)$ operations to examine all n items in order to derive the top candidates for each user in \mathbb{A}_t . Each affinity group \mathbb{A}_t would need $O(|\mathbb{P}_t|nd)$ operations for considering all $|\mathbb{P}_t|$ users in the group to construct the preferred item set c_t .

Coreset Construction as Finding a Set Cover. To accelerate the process of constructing the preferred item set c_t for an affinity group A_t , we want to find a δ -user coreset of A_t , and use it only instead of whole \mathbb{A}_t to construct c_t . We achieve this by first defining the idea of ϵ -set cover, and then show that each ϵ -set cover corresponds to a δ -coreset.

Definition 6.4. (ϵ -Set Cover) s_t is an ϵ -cover of \mathbb{P}_t if $\exists \mathcal{N}_{s_t}(p) \in s_t$ so that $\mathcal{N}_{s_t}(p)p^T \geq \epsilon$ for all $p \in \mathbb{P}_t$, where ϵ is a real number, and $\mathcal{N}_{s_t}(p_i) \in s_t$ denotes the nearest vector in s_t of p_i .

Theorem 6.1. Given an ϵ -cover s_t of \mathbb{A}_t , there exists a δ such that ϵ -cover s_t is a δ -user coreset of the affinity group \mathbb{A}_t .

The proof is shown in Appendix C.1. Therefore, we could construct a user coreset with an arbitrarily small δ by finding a cover with a greater ϵ .

Another nice property is that we could find an ϵ -set cover on sampled subset of \mathbb{P} and generalize asymptotically with bounded error. Denote $\mathbb{P}_{\mathbb{A}_t}$ to be same sampling process of \mathbb{P} generating user vectors p_i belonging to \mathbb{A}_t . We will have following result:

Theorem 6.2. For an affinity group \mathbb{A}_t , given any query q , an ϵ -cover of k samples $\{p_i\}$ drawn from $\mathbb{P}_{\mathbb{A}_t}$ would satisfy following inequality with probability at least $1 - \gamma$:

$$\min_i (|\mathcal{N}_{s_t}(p_i) q^T - p_i q^T|) \leq \delta + \sqrt{\frac{2 \log(1/\gamma)}{k}}.$$

Note that we demonstrate the proof in Appendix C.2.

Theorem 6.2 indicates that we could construct an ϵ -cover of sub-sampled vectors to have an asymptotically guaranteed difference of inner-product values to true distributions within the same affinity group. Consequently, our task becomes finding an ϵ -cover of all \mathbb{A}_t s and constructing the preferred item set c_t of it. Hence, we propose a fast adaptive representative selection method to efficiently construct an ϵ -cover with sub-sampled user latent vectors for each affinity group as summarized in Algorithm 6.2. For each affinity group \mathbb{A}_t , the adaptive representative selection method is applied to obtain a few representative ϵ -cover s_t . If there exists at least one user whose latent vector has cosine similarity lower than ϵ to all representative vectors, the algorithm iteratively generates more representatives until every user has high cosine similarity to at least one representative vector. As a result, the number of ϵ -cover $|s_t|$ must be less than or equal to the number of user vectors in the cluster $|\mathbb{P}_t|$, and in practice, $|s_t| \ll |\mathbb{P}_t|$ in most cases. Note that the adaptive representative selection method is applied on each affinity group \mathbb{A}_t independently. Next, the ϵ -cover s_t will be utilized to construct the preferred item set to reduce complexity from $O(|\mathbb{P}_t|nd)$ to $O(|s_t|nd)$.

Proximity Graph Navigation for Preferred Item Set Construction. To avoid ex-

Algorithm 6.2: Adaptive Representative Selection

Input: User latent vectors for an affinity group \mathbb{P} , the number of iterations T , the threshold ϵ , the number of new representatives w ;

Output: Representative vectors s .

```
1 Initialize  $s = \emptyset$  ;
2  $I = \arg \max_t s^T \mathbb{P}$  ;
3 repeat
4   for  $i = 1 \dots |s|$  do
5      $s_i = \sum_{j \in \{j | I[j]=i\}} \mathbb{P}[j]$  ;
6      $s_i = s_i / \|s_i\|_2$  ;
7    $I = \arg \max_t s^T \mathbb{P}$  ;
8    $Outliers = \{j | s_{I[j]}^T \mathbb{P}_j < \epsilon\}$  ;
9   for  $j \in Outliers$  do
10    Draw  $i$  from  $1 \dots w$  ;
11     $I[j] = |s| + i$  ;
12   if  $Outliers \neq \emptyset$  then
13    Append  $w$  vectors to  $s$  ;
14 until  $Outliers = \emptyset$  ;
15  $Outliers = \{j | s_{I[j]}^T \mathbb{P}_j < \epsilon\}$  ;
16 Append  $\mathbb{P}_{Outliers}$  to  $s$  ;
17 return  $s$ .
```

aming all n items ($O(nd)$ complexity) in preferred item set construction, we apply an approximate nearest neighbor search (ANNS) method to accelerate the computation. We adopt a model based on proximity graphs [226, 366] which has shown the state-of-the-art performance in ANNS. Specifically, a proximity graph is generated in which item vectors are nodes and nodes of similar item vectors are connected by edges. Since the item degree in recommender systems tends to follow a power-law distribution as illustrated in Figure 6.2b, this proximity graph has the small world properties [40] with sparse edges that offer high reachability between nodes. Hence, we apply the model of hierarchical navigable small world graphs [225, 226] to obtain the preferred item set c_t for each affinity group \mathbb{A}_t .

To construct the proximity graph of item vectors \mathbb{Q} as a hierarchical small world graph G , we iteratively insert the item vectors into the graph, where each node q has a list $\mathbb{E}(q)$ of at most efs approximate nearest neighbors that could be dynamically updated when inserting other item vectors, where efs is a hyperparameter. In addition, the edges in the graph are

Algorithm 6.3: QueryProximityGraph

Input: Hierarchical small world graph G ; the query vector q ; the number of the output approximate nearest neighbors K

Output: K nearest vectors in G

- 1 $p =$ Randomly select an entry node in G ;
 - 2 **for** $l = 1$ to L **do**
 - 3 $p = \arg \max_{\mathbf{r} \in \{p' | p' \in \mathbb{E}(p,l)\}} q^T \mathbf{r}$;
 - 4 **return** K Nearest Nodes in $\mathbb{E}(p, L)$ to q ;
-

organized as a hierarchy so that edges connecting items that have a high inner product value of their corresponding item vectors are at the bottom layers and edges connecting items whose vectors have low inner product values are at the top layers, thereby shrinking the search spaces for nearest neighbors. Let $L(e)$ denote the corresponding layer of edge e . Given two edges e_i and e_j , if $L(e_i) > L(e_j)$, then the nodes connected by edge e_i has a smaller inner product score than that of edge e_j . For simplicity, let $\mathbb{E}(q, l)$ denote the list of nodes connected to node q by edges in the l -th layer. Finally, the hierarchical small world graph G of item vectors \mathbb{Q} can be constructed in $O(dn \log n)$ [226, 366], where n is the total number of items; efs is treated a constant hyperparameter. Note that efs controls the trade-off between efficiency and accuracy for searching nearest neighbors because it decides the size of search space and the potential coverage of real nearest neighbors.

The hierarchical small world graph G provides the capability of efficiently querying K nearest neighbors of a vector q with a hierarchical greedy search algorithm. More specifically, we can greedily traverse the graph G by navigating the query vector from the bottom layer to the top layer to derive K approximate nearest neighbors to q as shown in Algorithm 6.3 with a $O(d \log n)$ time complexity for each query. For each affinity group \mathbb{A}_t , we perform a small world graph query to approximate $C(s_{t,i}, K)$ for each representative vector $s_{t,i} \in s_t$. The preferred item set c_t can then be constructed by taking the union operation to individual top- k sets as

$$c_t = \bigcup_{i=1}^{|s_t|} C(s_{t,i}, K). \quad (6.4)$$

Algorithm 6.4: Prediction Process for CANTOR

Input: User latent vectors p_i ; item latent vectors \mathbb{Q} ; Number of top recommendations K

Output: The indices of estimated top- K recommendations for the user i .

- 1 $z(p_i) = \arg \max_t v_t^T p_i$;
 - 2 $\text{logits} = p_i^T \mathbb{Q} [c_{z(p_i)}]$;
 - 3 $\text{topIndices} = \text{argsort}(\text{logits}, K)$;
 - 4 **return** topIndices .
-

6.3.4 Prediction Stage

To predict top recommendations for a user with the latent vector p_i , CANTOR relies on the clustering model parameterized by the centroid vector $v_t \in \mathbb{R}^d$ and the preferred item set c_t for each affinity group \mathbb{A}_t . More precisely, we first compute the affinity group indicator $z(p)$ as:

$$z(p_i) = \arg \max_r v_r^T p_i, \quad (6.5)$$

and evaluate full vector matrix product $p^T \mathbb{Q}_I$ over the corresponding item vectors of the preferred item set $\mathbb{Q}_I, I = \{j | j \in c_{z(p_i)}\}$. The computed results are then sorted to provide the final top- K recommendations for the user. Algorithm 6.4 shows the procedure of the prediction process.

6.4 Experiments

In this section, we conduct extensive experiments and in-depth analysis to demonstrate the performance of CANTOR.

6.4.1 Experimental Settings

Experimental Datasets. We evaluate the performance in two common tasks: item recommendation and personalized link prediction, using six publicly available real-world large-

Table 6.1: The statistics of six experimental datasets. Note that the personalized link prediction problem can be mapped to an item recommendation problem by treating each user as an item and recommending other users to a user in a similar way to that of recommending items to a user, and in this case the numbers of users and items are equal.

Task	Item Recommendation		
Dataset	MovieLens	Last.fm	Amazon
#(Users)	138,493	359,293	2,146,057
#(Items)	26,744	160,153	1,230,915
Task	Personalized Link Prediction		
Dataset	YouTube	Flickr	Wikipedia
#(Users)	1,503,841	1,580,291	1,682,759
#(Items)	1,503,841	1,580,291	1,682,759

scale datasets as shown in Table 6.1. For the task of item recommendation, the MovieLens 20M dataset (MovieLens) [147] consists of 20-million ratings between users and movies; the Last.fm 360K dataset (Last.fm) [53] contains the preferred artists of about 360K users; the dataset of Amazon ratings (Amazon) includes ratings between millions of users and items [204]. For the task of personalized link prediction, we follow the previous study [102] to construct three social networks among users: YouTube, Flickr, and Wikipedia [204]. Note that four of the six experimental datasets, Amazon, YouTube, Flickr, and Wikipedia, are available in the Koblenz Network Collection [204].

Evaluation Metrics. To measure the quality of an approximate algorithm for top- K recommendation we evaluate the top- K approximated recommendations with Precision@ K ($P@K$), which is defined by

$$\frac{1}{m} \sum_i \frac{|Y_K^i \cap S_K^i|}{K},$$

where Y_K^i and S_K^i are the top- K items computed by the approximate algorithm and full inner-product computations for user i ; m is the number of users. To measure the speed of each algorithm, we report the speedup defined by the ratio of wall clock time consumed by the full set of $O(mn)$ inner products to find the top- K recommendations divided by the wall clock time of the approximate algorithm.

Baseline Methods. To evaluate our proposed CANTOR, we consider the following five algorithms as the baseline methods for comparison.

- ϵ -approximate link prediction (ϵ -Approx) [102] sorts entries of the latent factor for each dimension to construct a guaranteed approximation of full inner products.
- Greedy-MIPS (GMIPS) [359] is a greedy algorithm for solving the MIPS problem with a trade-off controlled by varying a computational budget parameter in the algorithm.
- SVD-softmax (SVDS) [293] is a low-rank approximation approach for fast softmax computation. We vary the rank of SVD to control the trade-off between prediction speed and accuracy.
- Fast Graph Decoder (FGD) [366] directly applies small world graph on all items \mathbb{Q} and navigates to derive recommended items with user latent vectors as queries on the proximity graph. It also serves a direct baseline of only using proximity graph navigation.
- Learning to Screen (L2S) [61] is the first clustering-based method on fast prediction in NLP tasks with the state-of-the-art results on inference time but suffers from long preparation time. CANTOR is inspired by the clustering step in L2S, thus L2S serves as a direct baseline. In our implementation, random sub-sampling is applied to choose a subset of users for training L2S.

Note that [359] has shown that Greedy-MIPS outperforms other MIPS algorithms including LSH-MIPS [249, 295], Sampling MIPS [27] and PCA-MIPS [20], so we omit those other MIPS algorithms in our comparisons. Although bandit-based methods [126, 214, 215] have elegant mathematical properties and theoretical bounds, we did not include them originally because they generally perform worse than other methods in practical cases. For example, SCLUB [215], which is one of the state-of-the-art bandit-based approaches, only achieves 0.81x and 0.62x speedups on the Amazon and Wikipedia datasets with the official implementations. This is because bandit-based methods independently manipulate each dimension and cannot benefit from low-level optimization for linear algebra operations.

Implementation Details. For each dataset, the LIBMF library [66] is used to train a non-negative MF (NMF) model. More specifically, the number of dimensions for latent

vectors is 10 while the models are trained with all data for 100 iterations. Note that we adopt NMF models because of the restrictions of ϵ -Approx, but CANTOR does not have any limitation on matrix types. We implement CANTOR in Python with NumPy optimized by BLAS [33]. For the baseline methods, the implementations of GMIPS, SVDS, FGD, and L2S are provided by the original authors and highly-optimized while we utilize an efficient C++ implementation of ϵ -Approx. All experiments were run on a 64-bit Linux Ubuntu 16.04 server with 512 GB memory and single thread regime on an Intel[®] Xeon[®] CPU E5-2698 v4 2.2 GHz.

Hyperparameters in CANTOR. For the general settings of hyperparameters in CANTOR, we fix the number of sub-sampled user latent vectors u as 50,000 and the number of clusters r as 8. For adaptive representative selection, we set the number of iterations in the adaptive selection T as 10 and the similarity threshold ϵ as 0.99. The number of new representatives w in adaptive representative selection algorithm is set as 8. We also tune the size of dynamic nearest neighbor lists efs in the construction of hierarchical small world graphs for each dataset to achieve acceptable accuracy scores. As a result, the selections of efs are 200, 200, 1,500, 500, 1,500, and 100 for the datasets MovieLens, Last.fm, Amazon, YouTube, Flickr, and Wikipedia, respectively.

6.4.2 Performance Comparison

To fairly compare the performance, for each dataset, we tune the parameters such that each method can roughly achieve 0.99 P@1 accuracy. Table 6.2 shows the efficiency and the precision scores of CANTOR and all baseline methods on six datasets. Note that since the open-sourced library of GMIPS does not provide the breakdown of execution time into preparation and prediction time, the reported time includes both preparation and prediction processes. Among the baseline methods, FGD performs the best because it exploits the state-of-the-art algorithm for approximate nearest neighbor search to retrieve recommendations for each user. Although L2S is the most efficient baseline in the inference process, its preparation process is slow so that the overall speedup is further degraded. SVDS can

Table 6.2: Comparisons of top- K recommendation results on six datasets in two tasks. Note that P@ K measures the precision of approximating the top- K recommendations of full inner-product computations. SU indicates the ratio of speedup based on the original full inner product time of inferring top- K recommendations. For example, 9.4x means the computation time of the method is 9.4 times faster than the full inner product computation time. PT means the preparation time and IT represents the inference time in prediction process. The time units of *seconds*, *minutes*, and *hours* are represented as s, m, and h, respectively. Computation time of the full inner product method for each dataset is 71s (MovieLens), 1,017s (Last.fm), 92,828s (Amazon), 56,824s (Youtube), 71,653s (Flickr), and 72,723s (Wikipedia).

Task	Item Recommendation														
Dataset	MovieLens					Last.fm					Amazon				
Method	SU	PT	IT	P@1	P@5	SU	PT	IT	P@1	P@5	SU	PT	IT	P@1	P@5
ϵ -Approx	0.7x	0.19s	99.00s	0.753	0.671	0.5x	1.40s	36.78m	0.378	0.467	0.2x	23.42s	107.34h	0.529	0.559
GMIPS	3.9x	N/A	18.41s	1.000	0.972	2.3x	N/A	7.55m	0.997	0.966	1.8x	N/A	14.57h	0.993	0.952
SVDS	1.0x	0.10s	69.00s	1.000	1.000	0.9x	0.10s	19.25m	0.984	0.984	1.3x	5.32s	19.46h	0.952	0.953
FGD	2.8x	4.94s	20.10s	1.000	0.999	10.9x	0.49m	1.07m	0.997	0.988	19.7x	42.76m	35.83m	0.986	0.977
L2S	3.0x	22.15s	1.72s	1.000	1.000	9.0x	1.77m	0.12m	0.993	0.980	21.2x	71.02m	1.86m	0.988	0.979
CANTOR	9.4x	6.17s	1.36s	1.000	0.999	37.1x	0.37m	0.09m	0.999	0.998	29.0x	52.13m	1.26m	0.994	0.991
Task	Personalized Link Prediction														
Dataset	YouTube					Flickr					Wikipedia				
Method	SU	PT	IT	P@1	P@5	SU	PT	IT	P@1	P@5	SU	PT	IT	P@1	P@5
ϵ -Approx	0.1x	0.3m	129.2h	0.364	0.432	0.4x	0.29m	53.44h	0.545	0.581	0.2x	0.39m	130.61h	0.374	0.480
GMIPS	1.4x	N/A	11.12h	0.987	0.965	2.0x	N/A	10.10h	0.987	0.962	3.6x	N/A	5.64h	0.991	0.974
SVDS	1.0x	0.03m	15.30h	0.965	0.963	1.4x	0.03m	14.00h	0.952	0.946	1.4x	0.03m	14.83h	0.949	0.944
FGD	44.8x	10.28m	10.85m	0.989	0.981	37.5x	17.61m	14.25m	0.985	0.980	93.7x	4.18m	8.76m	0.990	0.985
L2S	6.9x	135.93m	0.79m	0.984	0.968	8.3x	142.84m	0.58m	0.989	0.980	22.4x	53.38m	0.84m	0.988	0.968
CANTOR	112.7x	7.75m	0.65m	0.993	0.985	54.7x	21.31m	0.53m	0.994	0.990	355.1x	2.45m	0.97m	0.995	0.991

efficiently decompose the preference matrix as its preparation process, but it still requires to examine all items many times to achieve sufficient accuracy so that the acceleration is unsatisfactory. In addition, it is worth noting that, although ϵ -Approx theoretically needs fewer multiplications than the full evaluation, it actually does not provide any acceleration in practice. Similar to bandit-based methods, this is because each dimension is independently processed so that the model cannot benefit from any low-level optimization for linear algebra operations.

Our approach CANTOR significantly outperforms all of the baseline methods in accelerating the overall execution time to provide top- K recommendations in all datasets. More specifically, CANTOR has similar inference time for the prediction process to that of L2S (that also reduces the candidate item sets for less computation) but the preparation process of CANTOR is much faster. This is because similarities between user latent vectors are well leveraged to avoid unnecessary and redundant computation.

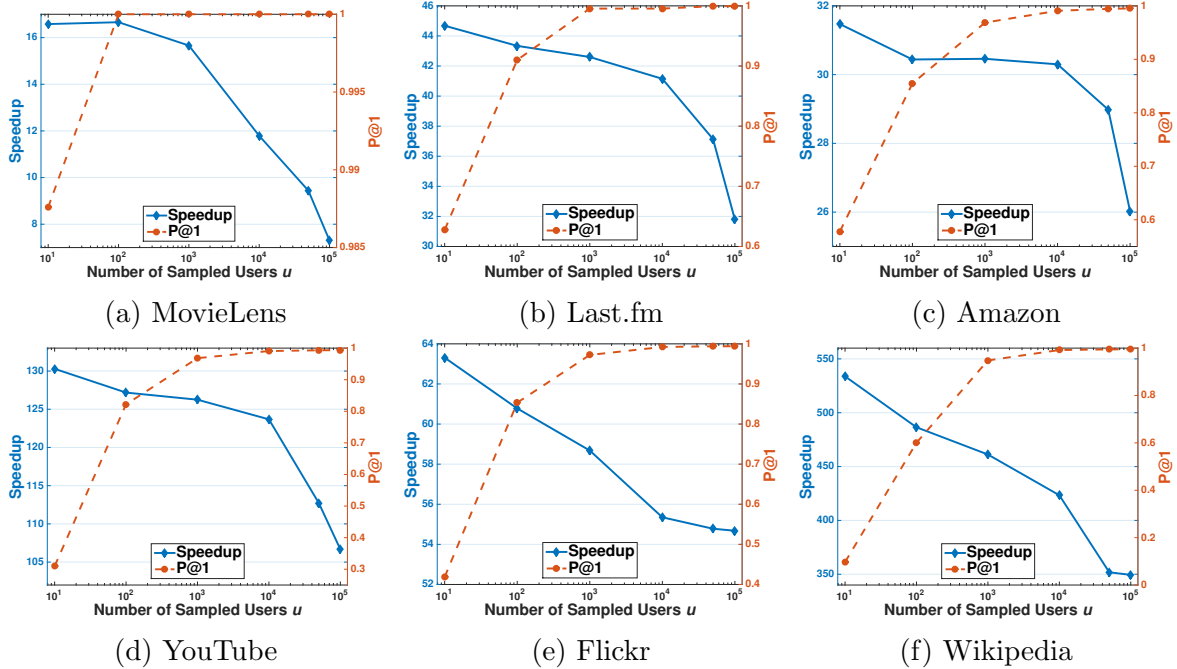


Figure 6.3: The ratios of speedup and the P@1 scores of CANTOR over different numbers of sampled users u in six datasets.

6.4.3 Number of Sub-Sampled User Latent Vectors

Since only a small subset of user latent vectors will be sub-sampled for user clustering, we verify how the number of sub-sampled users affects the performance in both efficiency and accuracy. Figure 6.3 illustrates the P@1 scores and the ratios of speedup of CANTOR for different numbers of sampled users in six datasets. It is obvious that smaller number of sampled user latent vectors is accompanied with greater speedup and lower P@1 score. However, CANTOR can generally achieve 99% P@1 scores after sampling more than around 10^4 users in all datasets. In other words, the distribution of the whole dataset can be preserved by sampling a small portion of users. For example, the Wikipedia dataset needs to sample only 5% of all users for high accuracy of recommendations.

6.4.4 Trade-off in Proximity Graph Construction

Proximity graph plays an important rule in CANTOR while the hyperparameter efs controls a trade-off between the efficiency and accuracy for generating the preferred item sets.

Figure 6.4 depicts the P@1 scores and the speedup ratios of CANTOR for different efs in six datasets. Obviously, a too-small efs leads to unsatisfactory approximation and low accuracy scores. More precisely, the $P@1$ scores considerably drops when efs is below 10^2 . On the other hand, CANTOR works well when efs is greater than 10^3 in all datasets. Hence, we suggest to tune efs in the range between 10^2 and 10^3 to reach a balance between efficiency and accuracy.

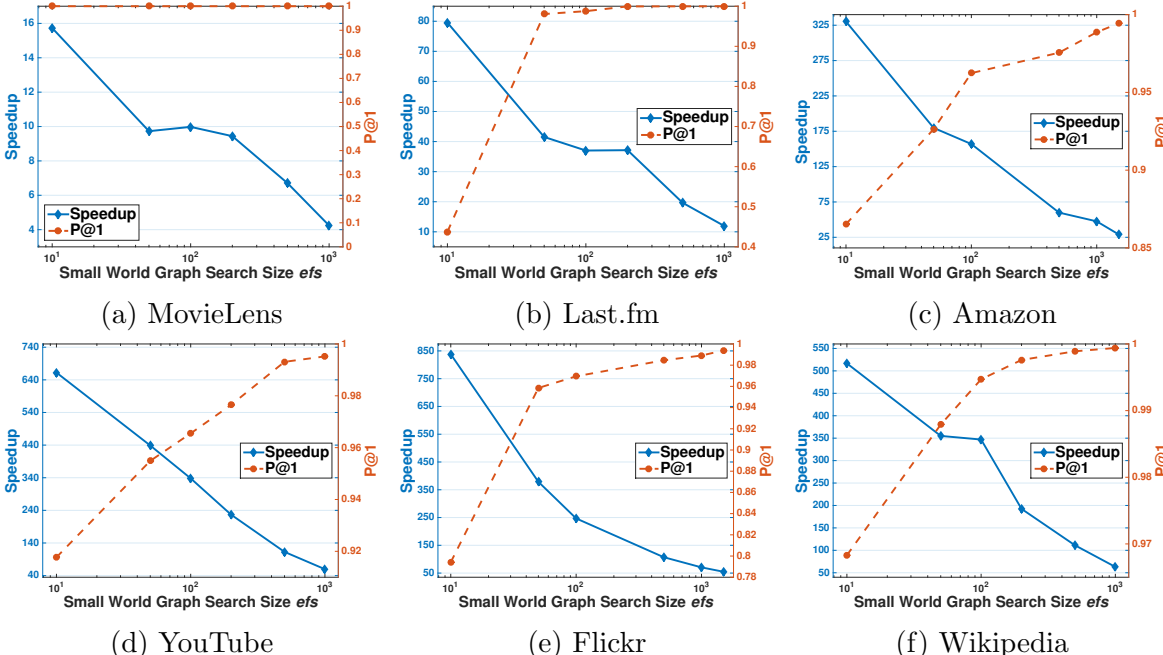


Figure 6.4: The ratios of speedup and the P@1 scores of CANTOR over different sizes of the hyperparameter efs in six datasets.

6.4.5 Number of Affinity Groups

Figure 6.5 shows the performance of CANTOR with different numbers of user affinity groups r in six datasets. When there are more affinity groups, the sizes of preferred item sets shrink because of fewer representative vectors for each cluster. As a consequence, CANTOR with larger group numbers considers fewer items in each affinity group, thereby achieving greater speedup. It is also noted that the great speedup comes with slight drop in accuracy. For example, there is only a 0.1% drop from $r = 2$ to $r = 16$. From these results, we suggest to

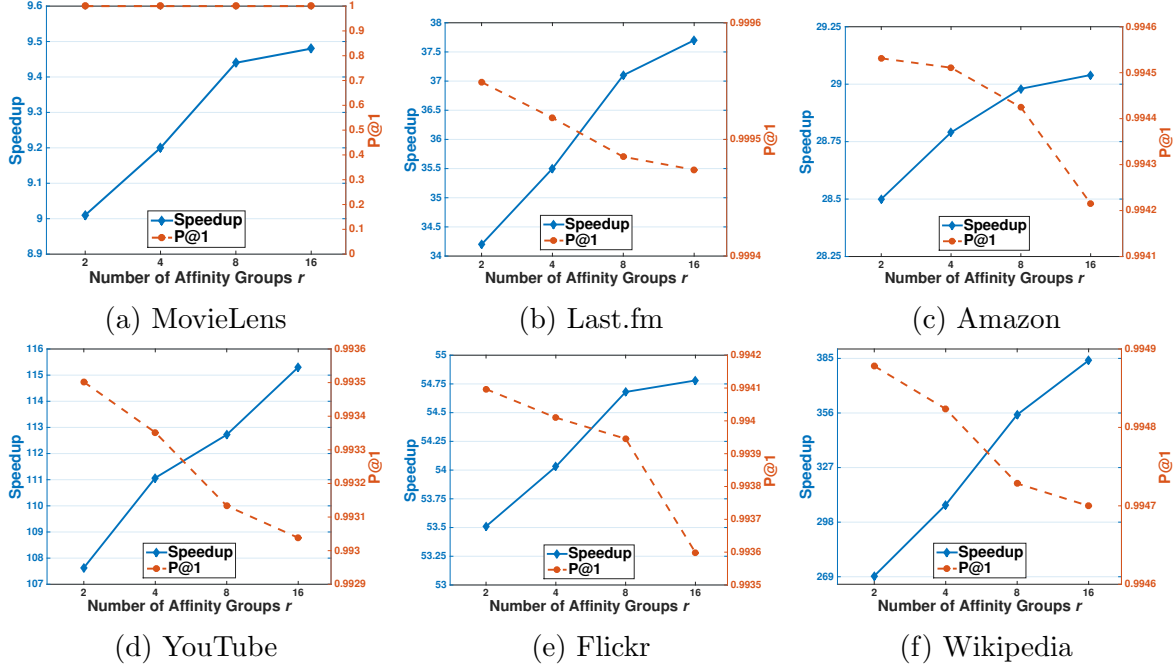


Figure 6.5: The ratios of speedup and the P@1 scores of CANTOR over different numbers of affinity groups r in six datasets.

set the number of user clusters r as a reasonable large number.

6.4.6 Effectiveness of Adaptive Representative Selection

The adaptive representative selection (ARS) method as shown in Algorithm 6.2 is important for CANTOR to accelerate the preparation process, so we also evaluate its effectiveness and robustness. Figure 6.6 illustrates the preparation time of CANTOR in six datasets with and without applying ARS. As a result, CANTOR using adaptive representative selection significantly outperforms the one without using ARS across all datasets when achieving similar accuracy. This further demonstrates the effectiveness and robustness of the adaptive representative selection method.

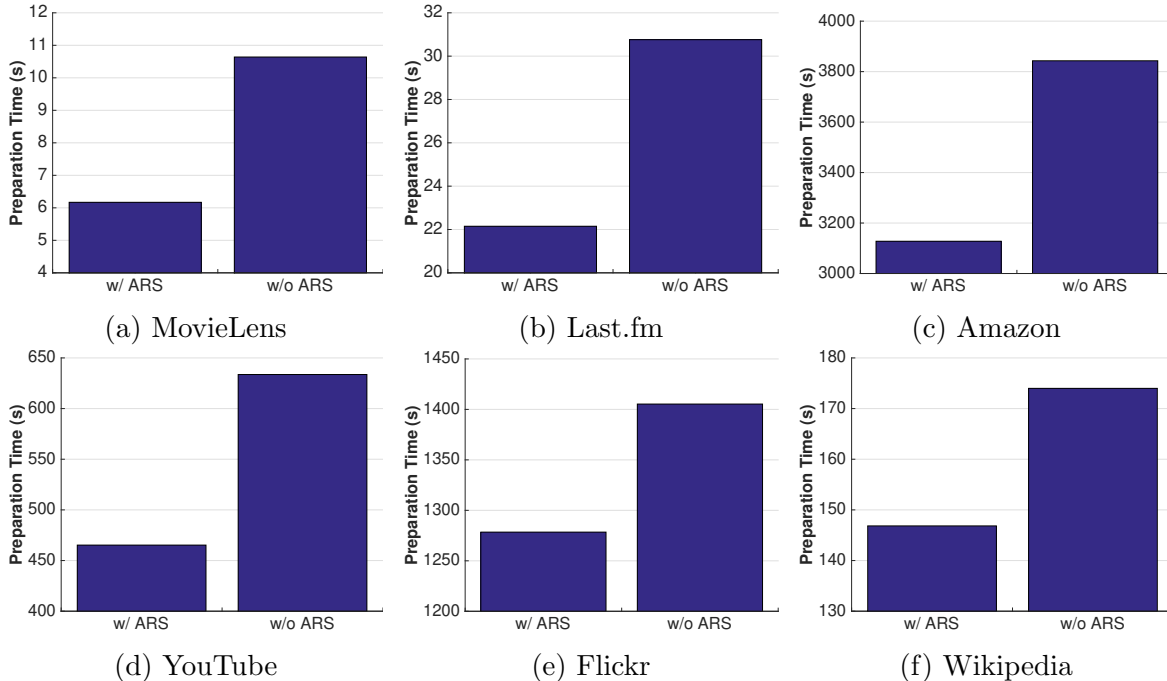


Figure 6.6: The preparation time of CANTOR with (w/) and without (w/o) the adaptive representative selection method (ARS) in six datasets.

6.5 Conclusions

In this chapter, we propose a novel framework for accelerating large-scale top- K recommender systems by exploiting user relationships and the redundancy of user vectors in the latent space. Our model, CANTOR, first clusters users into affinity groups, thereby determining as a user coreset of representative vectors for each group so that only a limited number of preferred items need to be examined for the users in the affinity group. Moreover, we mathematically prove that user coresets can be efficiently constructed by set covers of sub-sampled user latent vectors with an asymptotically guaranteed bound. Experimental results demonstrate that CANTOR significantly outperforms existing MIPS and approximate MF algorithms for accelerating top- K recommender systems. In particular, CANTOR achieves 355x and 29x speedup on the largest Wikipedia and Amazon datasets in two tasks while the accuracy scores still remain to be 99% for both P@1 and P@5. Moreover, the results of analysis also show the effectiveness and robustness of CANTOR.

CHAPTER 7

Efficient Signature Profiling in Genomic Data

With massive datasets in the era of big data, efficient data management also becomes an inevitable part of modeling human behaviors with those data. In this chapter, we use signature profiling in genomic data as an example to demonstrate how to efficiently manage sequence data and retrieve significant patterns in any incoming queries.

7.1 Introduction

K -mer profiling is a widely used technique to decipher relevant knowledge from sequencing data. These k -mers are short consecutive substring of a genomic sequence, and represent certain signatures to characterize different genomes or different regions in one genome. Instead of alignment, existing lightweight approaches pre-compute a searchable database of k -mers representing the “signatures,” and count the occurrences of these signatures in sequencing data. RNA and metagenomic sequencing are the predominant fields that use k -mer approaches. To name a few methods, Sailfish [261], RNA-Skim [371], and Kallisto [43] are the prevalent methods for RNA-Seq transcript quantification; LMAT [10] and Kraken [344] present efficient strategies to assign taxonomic labels for each metagenomic read. Other k -mer applications include studying the CpG island evolution in mammalian genomes using k -mer and k -flank patterns [55], comparing k -mer profiles of family trios to detect disease-causing variants [285], and mining differentially occurred k -mers between cases and controls for association mapping [273].

Most of the existing applications employ a set of fixed-size k -mers; however, selecting the appropriate k is challenging. If a k -mer is too long, it can fail to map a read with

sequencing errors. On the other hand, if a k -mer is too short, it can appear everywhere in the read data. In addition, the best k to characterize different genomic regions can vary. Several genome assemblers, such as SPAdes [28], Velvet [362], and SOAPdenovo [349], recognize the impact of k -mer sizes and consider building the de Bruijn graph with different sizes of k -mers. Chae *et al.* [55] have shown that it is necessary to consider patterns of 3- to 10-mers to construct the phylogenetic tree. Rahman *et al.* [273] have proposed to merge the differentially occurred k -mers to form longer sequences, resulting in variable-length sequences, for downstream analysis. Ju *et al.* [184] have also demonstrated the advantage of using variable-length k -mers for transcript abundance quantification. To the best of our knowledge, there are two existing approaches capable of generating variable-length k -mers as a set of signatures of interests. One exploits the suffix-tree structure to discover the shortest uncommon substrings [184], which represent the signatures of different transcript sequences. The other employs a pattern-growth approach to generate frequent k -mers of variable sizes, vl -mers [364], for both DNA and protein sequences. Despite the efforts in discovering variable-length k -mers in DNA sequences, current k -mer counters are optimized to process k -mers of a fixed length. Inevitably, it is critical to have a feasible data structure to store k -mers of different sizes, and to analyze sequencing data efficiently and accurately.

Given a set of k -mers with the same size, a straightforward counting method is to index the given k -mers with a hash table, and scan through read sequences with a fixed-size window. If a set contains k -mers of different sizes, the read sequences need to be scanned multiple times with different k 's. This repetition limits the analysis to evaluate only a small range of k 's. An alternative approach is to use existing efficient k -mer counting algorithms. Read sequences are first indexed and stored as k -mers, and the number of occurrences of each k -mer can be computed from the index. One of these counters, **Jellyfish** [228], has been widely used as the underlying structure for Sailfish, Kraken, and DIAMUND [285]. **Squeakr** [259] is a newly developed algorithm that also employs the thread-safe approach to efficiently query the counts of a specific k -mer. In addition, probabilistic hashing is commonly used in k -mer counting. Its implementations include **BFCOUNTER** [236] and **khmer** [367].

Disk-based hashing is another popular technique, and its related algorithms are **DSK** [279], **MSPkmerCounter** [216] and **KMC** [90, 91, 202]. Other data structures include burst tries used in **KCMBT** [227], and suffix-array structures employed in **Tallymer** [205] and **MSBWT** [156]. Several of these implementations, such as khmer and KCMBT, restrict the choice of k to fall in a threshold to mitigate the memory consumption and running time. Suffix-array based approach is the only one that presents the potential to process k -mers of variable lengths. All other methods are designed to process sequences with a fixed k . Thus, repeating the counting for different k 's is unavoidable.

Computing the frequencies of a set of k -mers with different sizes can be reduced to a multiple pattern matching problem [248] in computer science. A linear solution to scan the reads once is the Aho-Corasick algorithm [5], which constructs a tree automaton upon the trie of keywords. In this trie, there are additional links between internal nodes to facilitate the k -mer matching without backtracking, i.e., jumping back and forth of the query sequence. A drawback of maintaining this automaton is the memory requirement for storing long or large number of k -mers. As we increase the number or the length of k -mers, the tree grows wider and deeper respectively, which generates more nodes and links to facilitate the traversal. In addition, larger k -mers are usually more diverse and have shorter common prefixes, requiring more space for k -mer representation. Fortunately, the concise representation of DNA molecules allows further reduction in memory requirement of this automaton. Since these k -mers are composed of only four different characters: A, C, G, and T, they can be succinctly represented in a binary format. Traditionally, each nucleotide is encoded into two bits for its binary representation. Here, we propose to use an even more concise representation with one bit. We partition these four characters into two groups, and use one bit, i.e., 0 or 1, to represent them. This binarized representation allows us to significantly shrink the structure of the trie, and to substantially reduce the memory. The degenerated representation can cause collisions where different k -mers are encoded with identical binarized representation. To avoid this collision, each node on the tree contains a hash table to facilitate recovering the original k -mers.

It is important to note that the goal of our work is not to compute the frequencies of all possible k -mers with different sizes, but to profile a pre-defined set of variable-length k -mers as signatures in sequencing reads. The focus of our work is also different from assembling reads with variable sizes of k -mers. An example of a pre-defined set can contain the genetic markers of different microorganism in metagenomic studies. Since the term signatures here refer to a set of representative k -mers, we use these two terminologies interchangeably throughout this chapter. Our contributions are four-fold. First, we emphasize the needs of having a viable data structure to store and to profile k -mers of different sizes in DNA sequences. Second, to the best of our knowledge, this is the first work to profile a vast amount of pre-defined set of variable-length k -mers simultaneously in genomic data. We propose to apply the Aho-Corasick algorithm with a memory efficient automaton. Third, we leverage the properties of DNA sequence to construct an efficient in-memory structure, and employ the rolling hash technique to accelerate the match. Fourth, we adapt existing k -mer counters to perform the same task, and conduct a comprehensive analysis over 13 different methods. Results show that our method, TahcoRoll is more efficient in profiling signatures with a wide range of sizes than conventional k -mer counters. It is also resistant to the change of read length and quantity. The parallelization of TahcoRoll has demonstrated a promising improvement over different number of threads, where the parallelizations of KMCs and MSBWT are constrained by the disk I/O. Most importantly, TahcoRoll is able to analyze reads from Illumina, PacBio, and Oxford Nanopore on a commodity desktop computer while KMC3 and MSBWT fail on long reads.

7.2 Materials and Methods

We propose the thinned Aho-Corasick automaton accelerated by rolling hash (TahcoRoll) to profile variable-length k -mers in genomic data. In this section, we formally define the problem of signature profiling, and provide the details of our algorithm. We also describe how we adapt different k -mer counters to account for a set of variable-length k -mers. Lastly, we report the source of different datasets.









Signature p	Sequencing Reads T		Number of Occurrences c_p
	AATTGAGAT	ATTGACATCG	
ATT			2
GA			3
TTG			2
AGAT			1
TC			1

Figure 7.1: An example with five signatures and two sequencing reads in signature profiling. Each segment represents an occurrence of the corresponding signature in the read.

7.2.1 Problem Statement

We focus on counting the occurrences of a set of representative k -mers instead of all possible variable-length k -mers because of the following two reasons. First, the number of all variable-length k -mers in a DNA sequence is huge. More specifically, the lower-bound of time complexity to examine all occurrences of possible k -mers with different k is at least $O(L^2)$ for a read of length L . Second, there are existing works [184, 364] that focus on the discovery of relevant variable-length k -mers for different applications. Given a list of these signatures, it is not necessary to profile all possible variable-length k -mers.

Suppose that \mathbf{P} is the set of representative k -mers as signatures, where the length k is different across the set. Given a set of sequencing reads \mathbf{T} , our goal is to profile the occurrences of each signature $p \in \mathbf{P}$. Note that the lengths of occurred patterns are shorter than the read length. More formally, for each signature $p \in \mathbf{P}$, we aim to develop an efficient algorithm to compute the number of overall occurrences c_p is: $c_p = \sum_{t \in \mathbf{T}} |\{i \mid t[i \dots i + |p| - 1] = p, 1 \leq i \leq |t| - |p| + 1\}|$, where $|p|$ and $|t|$ indicate the lengths of the signature p and the sequencing read t , and $t[i \dots j]$ denotes the substring of t from the i -th to the j -th character. Figure 7.1 shows an example with five signatures $\mathbf{P} = \{\text{ATT}, \text{CA}, \text{TTC}, \text{ACAT}, \text{TG}\}$ and two sequencing reads $\mathbf{T} = \{\text{AATTGAGAT}, \text{ATTGACATCG}\}$. Each segment indicates an occurrence in the read for the corresponding signature. Occurrences can overlap with each other. A signature could have multiple occurrences in a single read (e.g., CA) or across the set of reads (e.g., ATT). For each signature $p \in \mathbf{P}$, occurrences

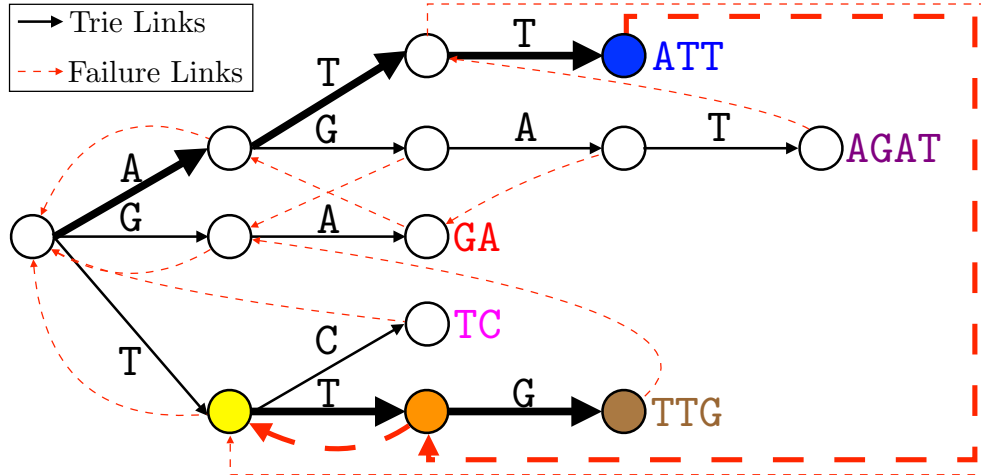


Figure 7.2: The automaton of AC with five signatures. Black solid links are trie links, and red dashed links are failure links. Colored nodes and thicker links are traversed while profiling the read ATTTG.

in the read set \mathbf{T} are counted as a number c_p , which is the objective of signature profiling.

7.2.2 Aho-Corasick Automaton

To solve signature profiling, an intuitive way is to reduce the task into multiple pattern matching [248] by mapping signatures onto patterns and each set of reads onto the input text. Multiple pattern matching algorithms find all occurrences in a read for each signature, so the signature profiling results can be obtained by aggregating these occurrences. We propose to apply the Aho-Corasick algorithm (AC) [5], one of state-of-the-art approaches for multiple pattern matching, to profile signatures.

AC conducts the matching process along a trie that corresponds to patterns. Each node in AC has a failure link that allows fast transitions from one node to the other representing its longest possible suffix without backtracking. Informally, AC constructs a finite state machine (or an automaton) that resembles a trie and failure links. The pattern matching process can be treated as transitions between nodes in the automaton, and failure links provide efficient transitions between failed matches. Figure 7.2 shows an example of AC with five signatures. For example, the node of signature ACAT has a failure link to the node of AT. Given a sequencing read ATTTG to be profiled, AC will first match the signature ATT

in the *blue* node. Then, it fails to match the third T and transits to the *orange* node that still has no child of T. After traveling along the failure link again to the *yellow* node, both the last two characters TC can proceed towards the *orange* and *brown* nodes that indicate a match of signature TTC.

The construction of the automaton in AC with signatures $p \in \mathbf{P}$ only requires a simple breadth-first search (BFS) with a $O\left(\sum_{p \in \mathbf{P}} |p|\right)$ linear time complexity. To profile signatures in reads $t \in \mathbf{T}$, AC only needs to simulate transitions on the automaton, which also has a linear time complexity $O\left(\sum_{t \in \mathbf{T}} |t| + \sum_{p \in \mathbf{P}} c_p\right)$. The space complexity of AC is also linear, $O\left(\sum_{p \in \mathbf{P}} |p|\right)$, to maintain a node and a constant number of links for each character. In theory, AC is a perfect fit for signature profiling.

7.2.3 Thinned Automaton with Binarized Pattern Matching

Even though we have shown the theoretical capability of AC for signature profiling, there are still some hurdles for AC in practice. One of the most critical issues is the memory usage when the number of signatures is huge. More specifically, each individual character in signatures can be referred to as a trie node, which provides plenty information and consumes a considerable amount of memory. For example, the Python implementation of AC requires more than 240 GB of memory to process 24 million signatures whose lengths range from 135 to 151. Especially for signatures with fewer and shorter common prefixes, nodes tend to have more child nodes. The greater width leads to the increase of memory usage.

To reduce both the number of nodes and the width of the automaton, we propose the thinned automaton with binarized pattern matching. More formally, each signature $p[1 \dots |p|] \in \mathbf{P}$ is transformed into a binarized pattern $p'[1 \dots |p|]$ before being added into the automaton. The i -th character $p'[i]$ of p' is defined as follows:

$$p'[i] = \text{binarize}(p[i]), \text{ where } \text{binarize}(c) = \begin{cases} 0 & , c \in \{\mathbf{A}, \mathbf{G}\} \\ 1 & , c \in \{\mathbf{C}, \mathbf{T}\} \end{cases} .$$

Note that these four characters can be randomly divided into two groups. From the analysis

presented in Table S3, we use a balanced partition which groups **A,G** together. Compressing two characters into one bit 0 or 1, binarized patterns improve the representation capability of a depth- d node in the trie from 1 to 2^d unbinarized pattern(s), thereby reducing both the width of the automaton and the number of nodes. In this work, the automaton with binarized patterns is named *thinned automaton* because of its reduced width. Here, we conduct a theoretical analysis of the improvement of the thinned automaton against the plain AC. For convenience, we assume that each character in a signature is uniformly distributed. To estimate the worst-case scenario, we assume that every signature has the largest length m observed in the set. While inserting a signature into a trie, the number of newly added nodes depends on the presence of its prefixes in the trie. Proposition 7.1 gives an expectation of finding prefixes for n signatures with c possible characters.

Proposition 7.1 (Proved in Appendix D.1). Given n signatures with c possible characters to be added into a trie, the expected number of signatures that fail to find their length- i prefixes along the trie during its insertion is $c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right) - c^{i-1} \left(1 - \left(\frac{c^{i-1}-1}{c^{i-1}}\right)^n\right)$, where $0 \leq i \leq m$.

Based on Proposition 7.1, we derive the expected number of nodes in a trie in Proposition 7.2.

Proposition 7.2 (Proved in Appendix D.2). Given n signatures of length m with c possible characters to be added into a trie, the expected number of trie nodes is $\sum_{i=1}^m \left[c^i - c^i \left(\frac{c^i-1}{c^i}\right)^n \right]$.

Following Proposition 7.2, Proposition 7.3 derives the expected improvement on the number of trie nodes when the number of signatures is approaching to a large number.

Proposition 7.3 (Proved in Appendix D.3). When the number of signatures in the automaton is approaching to a large number, the expected number of nodes in the thinned automaton is only $\frac{3}{2} \cdot \frac{1}{2^{m+1}}$ of those in the plain AC.

As shown in Proposition 7.3, the improvement with the thinned automaton is guaranteed under the assumption mentioned above. However, DNA sequences are biased. In this sce-

	Signature p				Sequencing Reads T	
Original	ATT	GA	TTG	AGAT	TC	AATTGAGAT ATTGACATCG
Binarized	011	10	111	0101	10	001110101 0111000101

Figure 7.3: The binarized representations for five patterns and two sequencing reads. Two signatures GA and TC share the same binarized pattern (red). A substring in a sequencing read ATCG has the identical binarized form to the signature AGAT (blue).

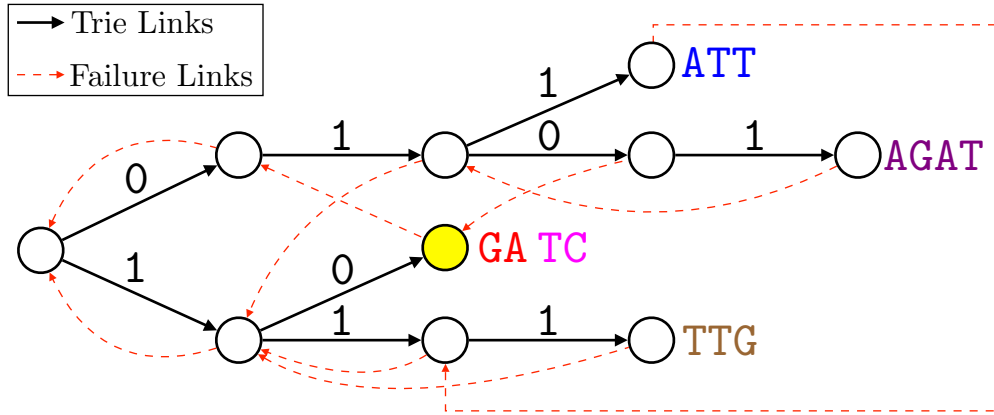


Figure 7.4: An example of the thinned automaton of AC with five signatures. Black solid links are trie links, and red dashed links are failure links. The *yellow* node represents two signatures GA and TC.

nario, where the characters of each signature are not uniformly distributed, the improvement can be more pronounced because more duplicated segments lead to fewer trie nodes.

Even though the thinned automaton reduces the number of nodes, compressed representations may lead to collisions. Figure 7.3 shows an example of binarized results for five patterns and two sequencing reads, and Figure 7.4 further illustrates the corresponding thinned automaton. Two signatures CA and TG share the same binarized pattern 10 and result in a collision when reaching the *yellow* node in Figure 7.4. Substrings with identical binarized representations may also lead to false matches. For instance, ATGC in the second read, which is not a signature, has the same binarized representation 0101 as the signature ACAT. To maintain the correctness of signature profiling, each match to a binarized pattern needs to be verified with the original signatures. In other words, it is very time-consuming if there are serious collisions in certain nodes. A naïve comparison costs $O\left(\sum_{p \in \{p|p'=h, p \in P\}} |p|\right)$ time to verify signatures with the same representation h .

7.2.4 Acceleration by Rolling Hash

Using hash functions is an intuitive idea to speed up comparisons between strings. As the lengths of signatures vary, arbitrary substrings of the read $t \in \mathbf{T}$ is required to compute hash values during verification. However, on-the-fly computation of hash values takes an additional linear time $O(|t|)$ for each checkup; precomputing all possible substrings is also infeasible due to dispensable computations and extensive $O(|t|^2)$ additional memory.

To accelerate verification, we propose to apply rolling hash [74] that alleviates the time complexity for each checkup from linear to constant with a linear-time preprocessing and an additional linear memory consumption. Rolling hash is a family of hash functions where the input is hashed with a window that moves through the input. A new hash value can be rapidly calculated from the given old hash value in $O(1)$ time. It also allows $O(1)$ query time on the hash value of any substring in the input with content-based slicing. We implement the Rabin-Karp algorithm [188] as the rolling hash function. Formally, the hash value of a length- L input $t[1 \dots L]$ is defined as follows:

$$H(t[1 \dots L]) = t[1]a^{L-1} + \dots + t[L-1]a^1 + t[L]a^0 \pmod{q},$$

where $t[i]$ is the i -th character of the input; a is a constant multiplier; q is a constant prime modulus. The hash value of a length- i prefix of t can be recursively calculated through the hash value of the length- $(i-1)$ prefix:

$$H(t[1 \dots i]) = \begin{cases} H(t[1 \dots i-1]) \cdot a + t[i] & , \text{ if } i > 1 \\ t[1] & , \text{ if } i = 1 \end{cases} \pmod{q}.$$

With bottom-up computation, hash values of all prefixes $H(t[1 \dots i])$ can be preprocessed in both $O(L)$ time and space complexity. Given the hash values of all prefixes, the hash value of a substring $t[i \dots j]$ can be derived in $O(1)$ as follows:

$$\begin{aligned}
& H(t[i \dots j]) \\
&= \begin{cases} H(t[1 \dots j]) - H(t[1 \dots i-1]) \cdot a^{j-i+1} & , \text{ if } i > 1 \\ H(t[1 \dots j]) & , \text{ if } i = 1 \end{cases} \pmod{q}.
\end{aligned}$$

As a theoretical analysis, Proposition 7.4 gives a theoretical upper-bound of the collision probability. The larger the prime modulus q , the smaller the hash collision probabilities. Note that we employ the Rabin-Karp algorithm instead of cyclic polynomials for hashing. This is due to the fact that the Rabin-Karp method has been demonstrated to be more efficient than cyclic polynomials for general applications [213].

Proposition 7.4 (Gonnet and Baeza-Yates [129]). The probability of two different random strings of the same length having the same hash value in Rabin-Karp rolling hash is $P(\text{collision}) \leq 1/q$, where q is the prime modulus in computations of the Rabin-Karp algorithm.

To apply rolling hash for acceleration, each node contains a hash table that maps a hash value onto the original signature. When transitioning to the node, the hash value of the matching substring in the read can be rapidly calculated and verified for its presence in the hash table. As a result, the average time complexity of each checkup reduces to $O(1)$. The overall time complexity of TahcoRoll is $O\left(\sum_{p \in \mathbf{P}} |p| + \sum_{t \in \mathbf{T}} |t| + \sum_{p \in \mathbf{P}} c_p\right)$, including the construction of the automaton and the matching process. The only memory overhead is hash tables with exactly $|\mathbf{P}|$ values, which is an amortized $O(|\mathbf{P}|)$ space.

7.2.5 Implementation Details

TahcoRoll is purely implemented by C++14 for universal usage in different platforms. It builds a thinned automaton on a set of signatures represented by their canonical form (i.e. the lexicographical minimum of itself and its reverse complementary sequence). Each node of the automaton holds an unordered map for an average constant time complexity of searches and insertions after querying binarized patterns on the structure. The memory consumption of all automaton nodes is simultaneously pre-allocated for efficient memory operations after

estimating the number of nodes by conducting binary searches on sorted patterns. The profiling process scans each read twice, one for its forward sequence and the other for its reverse complementary sequence. The operations of rolling hash are optimized by pre-computing the powers of the prime modulus. In addition, the paralleled version of TahcoRoll using multiple cores is also available for the scalability. More specifically, the paralleled TahcoRoll applies the multi-threading capability of C++14 for implementation.

7.2.6 Software Adaptation

Since most of the k -mer counters process reads with a fixed k , we use a Python script as a wrapper to handle different k -mer sizes and to call the appropriate functions from command line. We include all the k -mer counters mentioned in the Introduction. We implement two baseline methods. The first one is a naïve implementation in C++, denoted by “Naïve”. It uses a hash table to store k -mers and scans through the reads multiple times with different window sizes. Theoretically, Naïve is light in memory, but requires an extensive running time. The second baseline is the conventional Aho-Corasick algorithm. We test two publicly available implementations written in Python (PlainAC_Py; *pyahocorasick*1.1.3) and C++ (PlainAC_C++; *cjgdev/aho_corasick*). Details of different softwares are described in the Supplementary Materials and GitHub.

7.2.7 Synthetic K -mers Generation

To examine the effects of signature number and length, we generate four batches of k -mers with different lengths, denoted by *small* (15-31bp), *medium* (65-81bp), *large* (131-151bp), and *wide* (15-131bp). Each batch contains four sets of 1.2, 6, 12, and 24 million k -mers. These numbers are arbitrarily chosen to examine the scalability of different methods. The sequence of each k -mer is randomly assigned with four nucleotide characters, and a random length that falls in the appropriate range. These random signatures are designed to test the worst scenario as their characters are uniformly distributed and may not share as many common prefixes as in the real sequencing data. Each k -mer is represented by its canonical

form. Duplicated k -mers are removed from the list.

7.2.8 Synthetic Reads

We used polyester [119] to generate 15 sets of single-end RNA-Seq experiments, with read lengths of 75, 100, 125, 150, and 180bp. Each set contains 10-115 million reads from randomly selected transcripts based on Ensembl Human Genome GRCh38 [82]. The flexibility of synthetic data allows us to examine the effects of read number and length.

7.2.9 Real Datasets

We examined public datasets from a diverse range of sequencing platforms. The first dataset contains two experiments to study the transcriptomic analyses for lymphoblastoid cells [68]: SRR1293901 is a 2x262 cycle run from Illumina MiSeq and SRR1293902 is a 2x76 cycle run from Illumina HiSeq 2000. The second dataset, GSM1254204, aims to characterize the transcriptome of human embryonic stem cells using PacBio long reads [16]. The third set is generated by Oxford Nanopore to study the whole genome of breast cancer model cell line with different read lengths: SRR5951587, SRR5951588, and SRR5951600. For the RNA-Seq datasets, we use a list of 10,962,469 k -mers selected from transcript sequences that can distinguish different transcript isoforms. For the WGS datasets, 10,935,397 short sequences are randomly selected from the reference genome as signatures. Since long reads contain a higher error rate, we cannot set the k -mer size too long. It ranges from 25-60bp.

7.3 Results and Discussion

7.3.1 Automaton Construction

The memory of AC is sensitive to the composition of signature patterns such as k -mer lengths, the number of k -mers, and common prefixes shared by different k -mers. Figure 7.5 compares the computational resources used for automaton construction in TahcoRoll

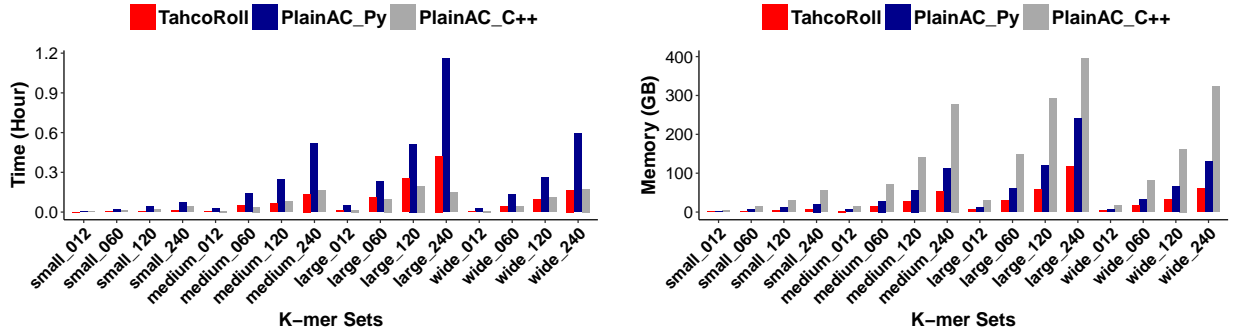


Figure 7.5: Run-time (left) and memory (right) for constructing the automaton given 16 sets of k -mer patterns. A lower value represents a more efficient approach. PlainAC_C++ maxes out the memory capacity while constructing the “large” batch of 24 million k -mers (large_240), and thus its recorded time and memory are truncated. TahcoRoll consistently requires less time and memory than PlainAC_Py

against PlainAC_Py and PlainAC_C++ over 16 sets of signatures. The implementation of PlainAC_C++ uses several additional data structures on each node to facilitate the traversal on an automaton, causing a huge memory overhead. As a result, PlainAC_C++ is fast in automaton construction, but requires twice and five times more memory than PlainAC_Py and TahcoRoll respectively. For the *large* batch of 24 million k -mers, PlainAC_C++ maxes out the memory capacity (>396GB) of our server, and thus the recorded run-time is truncated. Our thinned automaton consistently requires less time than PlainAC_Py in construction. As we increase the number of k -mers, the construction time rises. The memory of the thinned automaton is significantly reducing to nearly half of the memory required in PlainAC_Py. Larger k -mers are more diverse in their sequences, and often share shorter common prefixes with others. This phenomenon is reflected in our analysis, where the *large* batches of k -mers require more time and memory than others. The *wide* batches use less resource than the *large* ones because their k -mer sizes are widely spread from 15-151bp and contain fewer long k -mers.

7.3.2 Pilot Study of 13 Approaches

We perform a preliminary assessment of the memory footprint and run-time on 11 existing counters, together with Plain_AC and TahcoRoll. Since most of these counters are designed

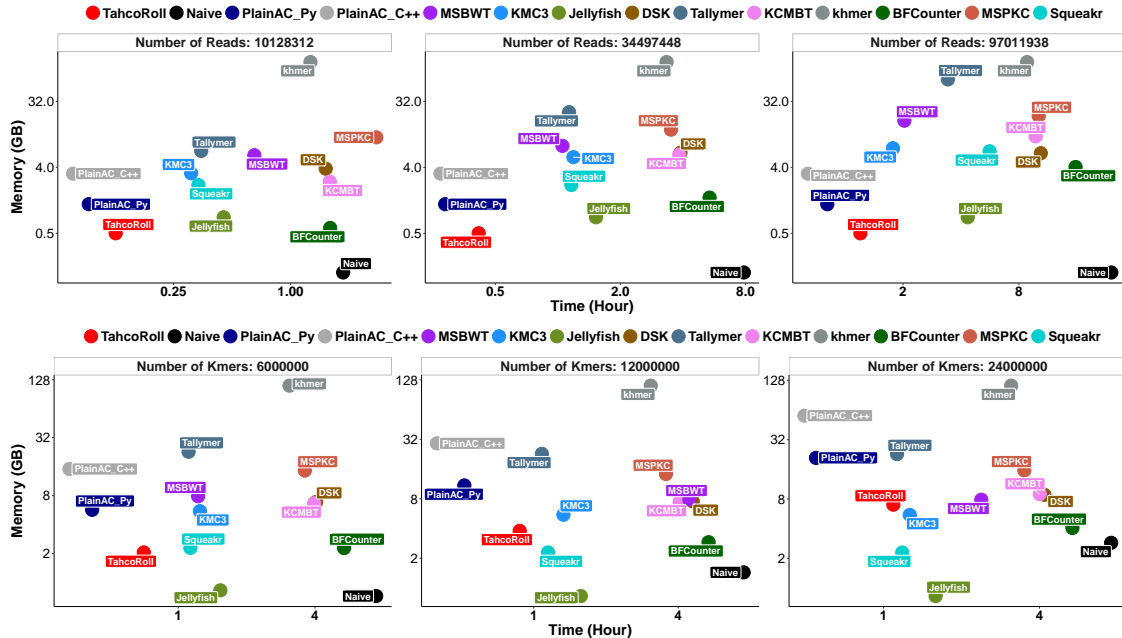


Figure 7.6: Run-time (x-axis) and memory (y-axis) of counting *small* batches of k -mers on synthetic reads of 75bp. Each point represents a pair of measurements (run-time and memory). Points located on the lower left corner of each plot indicate more efficient approaches. The top panel examines three different read sets with 1.2 million k -mers; the bottom panel examines three different k -mer sets with 34,497,448 reads.

to process data with a fixed k , they present a limitation on handling a wide range of k 's. Moreover, few of these algorithms limit the choice of k . For these reasons, we inspect the capability of different approaches with single thread using small datasets: synthetic reads with 75bp and *small* batches of signatures. For all of the Aho-Corasick-based approaches, we index the k -mers using their canonical representation. Each read is scanned twice, one for its forward sequence and the other for its reverse complementary sequence. This technique alleviates the memory burden of storing each k -mer twice in the automaton.

We separate the analyses into two panels as demonstrated in Figure 7.6. The top panel focuses on different number of reads, and the bottom panel focuses on different number of k -mers. Methods in the bottom-left corner of each plot indicate being both time and memory efficient. As we predicted, Naive uses very little memory, but takes a long time to complete. PlainAC is fast, but requires a large amount of memory when increasing the number of k -mers. Consistent with the analysis in Figure 7.5, PlainAC_C++ uses twice as

much memory as PlainAC_Py.

TahcoRoll is the most efficient approach in five out of these six analyses. KMC3 and Squeakr use less memory when there are 24 million k -mers, but requires more time than TahcoRoll. When we fix the number of k -mers (top panel), the memory and run-time for KMC3 and Squeakr increase with the number of reads, but the memory stays constant for both TahcoRoll and Jellyfish. Jellyfish is memory efficient when counting a given list of k -mers with the same size; however, repeating this process for different k 's makes it more time-consuming than TahcoRoll.

We also validate the accuracy by comparing the counts reported in each approach to Naïve. Probabilistic approaches, BFCOUNTER and khmer, neglect singleton k -mers. The exact counting mode of Squeakr is unable to count short k -mers if all of them exist in reads, due to its implementation design. All other approaches agree with Naïve. The results are omitted due to space limitations.

To avoid waiting on extremely slow counters, we remove Naïve, DSK, khmer, BFCOUNTER, MSPKC, and KCMBT for further analyses. From the memory usage prospective, we take out Tallymer and PlainAC_C++ since Tallymer does not scale well with more and longer reads and PlainAC_C++ uses up the memory for more and longer k -mers. Squeakr is also removed as it often fails to count shorter k -mers. The remaining analyses are carried out for PlainAC_Py, Jellyfish, KMC3, MSBWT, and TahcoRoll.

7.3.3 Extensive Study on Synthetic Datasets with Both Single and Multiple Threads

We use 1.2 million k -mers ranging from 15-151bp (*wide*) to evaluate the scalability on different read lengths and number of reads. We highlight the total run-time and memory consumption of each approach in Table 7.1. The run-time is further broken down into the automaton construction phase (Prep) and the read querying phase (Query) for TahcoRoll and PlainAC_Py. The two phases of MSBWT and KMC3 include indexing the reads (Prep) and querying the k -mers (Query). Read processing is performed in the querying phase of

Table 7.1: Time (Hour) and memory (GB) of synthetic signatures over different read sets. Dagger (†) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.

Read Length	Total Reads	TahcoRoll				PlainAC_Py				MSBWT				KMC3				Jellyfish	
		Prep	Query	Time	Mem	Prep	Query	Time	Mem	Prep	Query	Time	Mem	Prep	Query	Time	Mem	Time	Mem
75bp	10,128,312	0.006	0.09	0.10†	3.29*	0.02	0.11	0.13	6.75	0.40	0.01	0.41	5.86	1.49	0.02	1.51	3.30	1.83	4.74
	34,497,448	0.005	0.28	0.29†	3.29*	0.02	0.37	0.39	6.75	0.90	0.01	0.91	7.88	2.45	0.09	2.53	5.52	5.55	4.74
	97,011,938	0.005	0.78	0.78†	3.29*	0.02	1.04	1.06	6.75	3.26	0.02	1.95	17.57	5.82	0.68	6.50	8.00	15.24	4.74
100bp	11,397,007	0.005	0.13	0.14†	3.29*	0.03	0.17	0.20	6.75	0.45	0.01	0.46	6.40	2.42	0.33	2.75	3.83	3.32	4.74
	41,054,662	0.005	0.47	0.48†	3.29*	0.02	0.59	0.61	6.75	1.29	0.01	1.30	11.10	4.81	0.61	5.42	7.56	11.25	4.74
125bp	114,813,452	0.006	1.35	1.36†	3.29*	0.02	1.59	1.61	6.75	3.49	0.02	3.51	26.00	16.23	3.35	19.58	20.10	31.78	4.74
	10,822,319	0.004	0.15	0.15†	3.29*	0.03	0.19	0.22	6.75	0.63	0.01	0.65	6.83	2.81	0.81	3.61	4.15	5.26	4.74
	58,012,701	0.005	0.77	0.78†	3.29*	0.03	0.99	1.02	6.75	2.59	0.02	2.61	17.48	10.53	2.51	13.04	19.03	27.22	4.74
150bp	107,375,244	0.005	1.37	1.38†	3.29*	0.02	1.84	1.87	6.75	4.56	0.02	4.58	29.75	18.46	3.92	22.37	34.59	50.41	4.74
	27,628,054	0.006	0.35	0.36†	3.29*	0.02	0.55	0.57	6.75	1.69	0.01	1.71	11.46	9.09	1.88	10.97	14.87	18.78	4.74
	57,437,772	0.007	1.20	1.21†	3.29*	0.02	1.20	1.22	6.75	3.50	0.02	3.51	20.31	17.26	3.98	21.24	31.10	36.86	4.74
180bp	114,306,300	0.006	2.01	2.01†	3.29*	0.03	2.42	2.44	6.75	5.86	0.02	5.88	37.27	33.45	8.25	41.69	58.23	74.29	4.74
	16,197,631	0.006	0.35	0.35†	3.29*	0.03	0.40	0.43	6.75	2.43	0.01	2.45	9.30	7.45	1.99	9.44	14.61	15.51	4.74
	37,836,905	0.005	0.86	0.87†	3.29*	0.02	0.87	0.90	6.75	3.20	0.02	3.22	16.96	16.05	4.20	20.26	33.34	35.14	4.74

Table 7.2: Time (Hour) and memory (GB) of synthetic reads over different k -mer sets. Dagger (†) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.

K-mer Batch	Total K-mers	TahcoRoll				PlainAC_Py				MSBWT				KMC3				Jellyfish	
		Prep	Query	Time	Mem	Prep	Query	Time	Mem	Prep	Query	Time	Mem	Prep	Query	Time	Mem	Time	Mem
Small (15-31bp)	1,200,000	0.0004	2.56	2.56	0.51*	0.003	1.81	1.81†	1.25	5.39	0.01	5.40	34.35	3.99	0.35	4.34	14.61	11.15	0.83
	6,000,000	0.002	4.83	4.83	2.09	0.02	2.42	2.44†	5.70	5.39	0.06	5.46	34.31	5.39	0.84	6.23	14.61	11.11	0.83*
	12,000,000	0.003	5.48	5.48	3.85	0.03	2.74	2.77†	10.93	5.39	0.12	5.51	34.35	5.89	0.95	6.84	14.61	11.17	0.83*
	24,000,000	0.006	7.22	7.23	7.13	0.09	3.11	3.21†	20.93	5.39	0.23	5.63	34.35	5.94	0.96	6.91	14.61	11.15	0.83*
Medium (65-81bp)	1,200,000	0.005	2.01	2.01†	2.82	0.03	2.42	2.45	5.83	5.39	0.01	5.40	34.35	5.03	2.59	7.62	58.16	11.41	2.47*
	6,000,000	0.02	2.47	2.49†	13.49	0.13	4.77	4.90	28.59	5.39	0.06	5.45	34.35	4.90	1.91	6.81	58.16	11.37	2.47*
	12,000,000	0.09	3.53	3.62†	26.52	0.27	5.27	5.54	56.71	5.39	0.11	5.50	34.33	4.90	1.93	6.83	58.16	11.07	2.47*
	24,000,000	0.16	4.00	4.16†	52.11	0.75	5.25	6.00	112.5	5.39	0.22	5.61	34.35	4.87	1.49	6.37	58.16	11.36	2.47*
Large (131-151bp)	1,200,000	0.02	2.65	2.67†	6.10	0.06	2.98	3.04	12.24	5.39	0.02	5.41	34.35	3.51	2.35	5.87	67.27	6.66	4.74*
	6,000,000	0.08	3.51	3.59†	29.91	0.29	4.34	4.63	60.63	5.39	0.08	5.47	34.35	4.28	4.04	8.32	67.27	6.72	4.74*
	12,000,000	0.18	4.37	4.55†	58.43	0.55	4.98	5.53	118.97	5.39	0.16	5.55	34.33	5.14	4.50	9.65	69.19	8.59	4.38*
	24,000,000	0.42	4.42	4.84†	117.79	1.33	4.90	6.23	240.67	5.39	0.29	5.69	34.35	4.10	3.05	7.17	67.27	6.73	4.74*

TahcoRoll and PlainAC_Py, but in the preparation phase of MSBWT and KMC3. Therefore, the run-time of querying is not on the same scale across different approaches. For Jellyfish, we use its function to count the list of k -mers directly, so the run-time cannot be split in details. Its memory usage depends on the size of the list of k -mers, and can be as efficient as TahcoRoll. However, its run-time does not scale well with datasets containing more or longer reads. TahcoRoll consistently outperforms others across different read sets in both run-time and memory. Our thinned automaton is more compact, which makes it more efficient than the conventional automaton.

Next, we use 86,976,737 reads of 180bp to evaluate the scalability on different batches

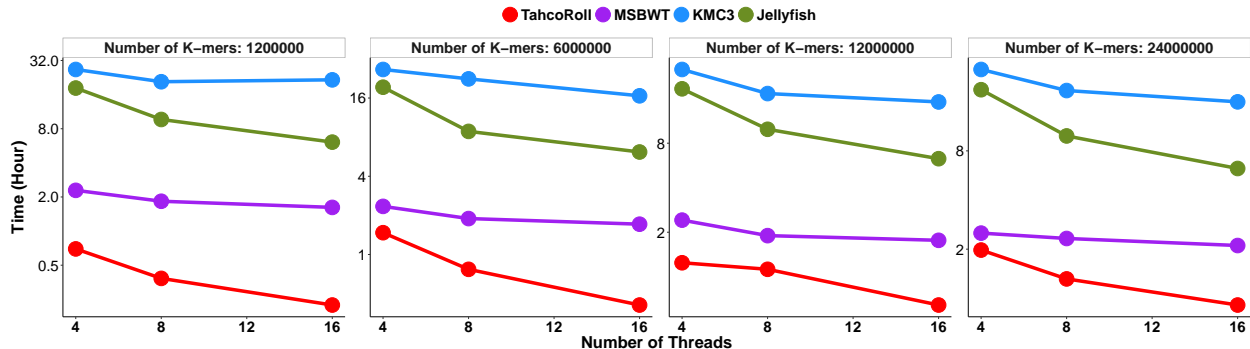


Figure 7.7: Multi-threads evaluation of MSBWT, KMC3, Jellyfish and TahcoRoll on profiling 86,976,737 synthetic reads of 180bp with *wide* batches *k*-mers.

of *k*-mers, which are designed to test the worst scenario. Table 7.2 shows that when the *k*-mers are short (*small*), PlainAC_Py uses the least amount of time. When *k*-mers get longer, TahcoRoll is the most efficient approach. This observation is due to less collision in the signature sets. Under a severe condition where there is a large number (12 and 24 million) of *k*-mers with uniformly distributed characters, TahcoRoll requires more memory than MSBWT in three out of six cases. It is worth mentioning that both MSBWT and KMC3 write a huge amount of intermediate files to disk (at least 16GB for MSBWT and 43GB for KMC3 in this dataset) to alleviate the memory bottleneck. In contrast, TahcoRoll is an in-memory approach that does not generate any intermediate data.

MSBWT, KMC3, and Jellyfish allow indexing reads in parallel, so we evaluate the parallel settings on the *wide* batches of *k*-mers. Figure 7.7 shows the run-time of analyzing 86,976,737 synthetic reads of 180bp across four sets of *k*-mers. Both Jellyfish and TahcoRoll scale well with the number of threads, but the improvement of MSBWT and KMC3 is marginal. This is mainly due to the limitation of I/O as these two approaches constantly read and write files to disk. The run-time of TahcoRoll remains faster than others across different experiments and threads. The four-thread TahcoRoll also demonstrates to be faster than others with 16 threads. Table 7.3 presents the run-time and memory usage in details for each setting. The memory usage does not vary with different number of threads.

Table 7.3: Time (Hour) and memory (GB) of profiling synthetic reads over different *wide* batch of k -mer sets. Dagger([†]) marks the most time efficient approach; asterisk (*) marks the most memory efficient approach.

Total K-mers	Methods	4-Thread (Hour)	8-Thread (Hour)	16-Thread (Hour)	Memory (GB)
1,200,000	TahcoRoll	0.70 [†]	0.38 [†]	0.22 [†]	3.50*
	MSBWT	2.29	1.83	1.62	30.83
	Jellyfish	18.29	9.66	6.10	4.74
	KMC3	26.68	20.79	21.61	72.83
6,000,000	TahcoRoll	1.47 [†]	0.77 [†]	0.41 [†]	16.07
	MSBWT	2.35	1.89	1.71	30.83
	Jellyfish	19.39	8.86	6.15	4.74*
	KMC3	26.55	22.47	16.65	72.77
12,000,000	TahcoRoll	1.24 [†]	1.12 [†]	0.64 [†]	31.49
	MSBWT	2.42	1.90	1.76	30.83
	Jellyfish	18.66	9.94	6.29	4.74*
	KMC3	25.22	17.33	15.21	73.08
24,000,000	TahcoRoll	1.98 [†]	1.32 [†]	0.91 [†]	61.86
	MSBWT	2.51	2.33	2.11	30.83
	Jellyfish	18.96	9.88	6.24	4.74*
	KMC3	22.22	18.74	15.97	73.08

7.3.4 Real Datasets from Different Sequencing Platforms

Synthetic studies demonstrate the worst case of signature sets. Here, we examine the practical usage by analyzing signatures from real DNA sequences with reads from different sequencing platforms. Experiments are conducted on a desktop machine of Coretm i7-3770 CPU@3.4.0GHz. Since TahcoRoll is more resistant to number of reads and different read lengths than other methods as demonstrated from synthetic studies, we examine its competency on four sets of long read data.

Table 7.4 summarizes the nature and analysis of each dataset. MSBWT, KMC3, and Jellyfish are run with eight threads; TahcoRoll is run with single thread and eight threads. For the measurement that is less efficient than TahcoRoll, we compute the fold-change to those reported by the eight-thread TahcoRoll. MSBWT is unable to finish indexing for the PacBio data within two days. KMC3 cannot index long reads from Nanopore as the data exceeds the buffer size automatically set by the program; it also uses up all the memory available on the machine (32G) for the PacBio data. Overall, the run-time of single-thread TahcoRoll is as efficient as Jellyfish with eight threads, and significantly outperforms KMC3 in short reads and MSBWT in long reads. In the parallel settings, TahcoRoll runs at least

Table 7.4: Evaluation of real datasets across different sequencing platforms. MSBWT, KMC3, and Jellyfish are run with eight threads. Fold-change is relative to the measurements reported by eight-thread TahcoRoll.

Dataset	SRR1293902	SRR1293901	GSM1254204	SRR5951587	SRR5951588	SRR5951600	
Source	RNA-Seq	RNA-Seq	RNA-Seq	WGS	WGS	WGS	
Platform	Illumina HiSeq	Illumina MiSeq	PacBio	Nanopore	Nanopore	Nanopore	
Number of Reads	38,278,052	9,524,186	3,239,918	205,685	171,398	161,148	
Average Read Length	75	262	1113	3kb	8kb	12kb	
Number of <i>Sig-mers</i>	10,962,469	10,962,469	10,962,469	10,935,397	10,935,397	10,935,397	
Lengths of <i>Sig-mers</i>	25-60	25-60	25-60	25-60	25-60	25-60	
Time (Hour)	TahcoRoll (1-thread)	1.20	1.40	1.17	0.27	0.44	0.65
	TahcoRoll (8-thread)	0.23	0.28	0.22	0.06	0.09	0.16
	MSBWT	0.95 (4.1X)	1.64 (5.8X)	NA	3.03 (53.4X)	2.79 (29.5X)	12.31 (77.7X)
	KMC3	15.85 (68.5X)	14.16 (50.7X)	19.26 (87.3X)	exceed buffer size		
	Jellyfish	0.94 (4.0X)	1.56 (5.6X)	1.13 (5.1X)	0.27 (4.8X)	0.48 (5.1X)	0.68 (4.2X)
Memory (GB)	TahcoRoll	4.18	4.17	4.18	10.5	10.5	10.5
	MSBWT	7.76 (1.8X)	70.10 (16.8X)	NA	1.89	3.16	4.65
	KMC3	28.76 (6.8X)	24.84 (5.9X)	31.34 (7.4X)	exceed buffer size		
	Jellyfish	1.79	1.79	1.79	1.79	1.79	1.79

Table 7.5: Evaluation of different binarized representations. Time is reported in hour and memory is reported in gigabyte. Nucleotides can be divided into balanced or unbalanced partitions. The p -values are computed through paired t -tests on time against the default setting: $\{\{A, G\}, \{C, T\}\}$, and adjusted by Bonferroni correction.

Mapping	$0=\{A, C\}; 1=\{G, T\}$		$0=\{A, T\}; 1=\{C, G\}$		$0=\{A\}; 1=\{C, G, T\}$		$0=\{C\}; 1=\{A, G, T\}$		$0=\{G\}; 1=\{A, C, T\}$		$0=\{T\}; 1=\{A, C, G\}$	
	Time	Mem	Time	Mem	Time	Mem	Time	Mem	Time	Mem	Time	Mem
SRR1293902	1.28	4.49	1.27	4.42	1.54	3.36	1.42	3.20	1.54	3.12	1.93	2.98
SRR1293901	1.35	4.49	1.39	4.42	1.72	3.36	1.71	3.20	1.79	3.12	1.96	2.98
GSM1254204	1.26	4.49	1.26	4.42	1.46	3.36	1.64	3.20	1.70	3.12	2.01	2.98
SRR5951587	0.32	11.11	0.30	10.84	0.34	9.85	0.47	7.93	0.57	7.85	0.54	9.27
SRR5951588	0.52	11.11	0.48	10.84	0.58	9.85	0.83	7.93	1.21	7.85	0.81	9.27
SRR5951600	0.76	11.11	0.74	10.84	0.78	9.85	1.46	7.93	1.36	7.85	1.11	9.27
p -value	0.3408		0.12714		0.033552		0.043938		0.008988		0.010212	

four times faster than MSBWT and Jellyfish, and demonstrates a drastic improvement over KMC3.

Lastly, we examine the impact of different binary representations of the nucleotides under the single-thread setting. The concise representation requires a many-to-one mapping between four nucleotides and two single binary values. The four characters can be divided into balanced partitions: $\{\{A, C\}, \{G, T\}\}$, $\{\{A, T\}, \{C, G\}\}$, and $\{\{A, G\}, \{C, T\}\}$, or unbalanced partitions: $\{\{A\}, \{C, G, T\}\}$, $\{\{C\}, \{A, G, T\}\}$, $\{\{G\}, \{A, C, T\}\}$, and $\{\{T\}, \{A, C, G\}\}$. The default setting groups $\{A, G\}$ together, and $\{C, T\}$. Table 7.5 summarizes the comparison of alternative mappings using real datasets. On average, the default mapping runs faster (shown in Table 7.4) than alternative mappings. We use two-tailed paired t -tests to compare the

run-time of each mapping against the default choice. The p -values are adjusted by Bonferroni correction [39] to account for the issue of multiple hypothesis testing. Among all balanced partitions, the default setting uses the least amount of memory, but its run-time is not significantly different from others (p -values > 0.05). Unbalanced partitions provide more compact representations as revealed by their memory usages, but require more time to resolve collisions than the default setting (p -values < 0.05).

7.4 Conclusion

In this chapter, we propose a novel task of variable-length k -mer profiling in genomic sequences. While the necessity of diversifying k -mer lengths has already been shown in many studies [55, 184, 364], most of these works only support fixed-length k -mers and need an enormous amount of memory, disk space, and time to profile k -mers with a wide range of k 's.

By leveraging the techniques of binarization and rolling hash for Aho-Corasick automaton, we construct a thinned Aho-Corasick automaton accelerated by rolling hash (TahcoRoll) to profile variable-length k -mers in genomic data. The main advantage of TahcoRoll is its in-memory property which does not require any disk space.

A pilot study first gives a comprehensive overview of the strengths and limitations of 13 different k -mer counting approaches. Additional experimental results show that TahcoRoll scales well with both longer reads and a larger number of reads, especially that its memory usage is independent of the read data. It is the only approach that can efficiently process data from different sequencing platforms. In the evaluation of k -mer sets, Aho-Corasick approaches use less time than others in all cases. Between the two implementations, TahcoRoll requires half of the memory needed in PlainAC_Py.

Although all of our experiments focus on counting the frequency of a set of k -mers, the structure of this thinned automaton can be expanded to store essential information for each k -mer, such as its explicit positions in a genome for occurrence profiling. TahcoRoll

opens up the opportunity to profile a set of variable-length k -mers, especially in long read datasets. It can be used as a stand alone software or to be integrated into existing pipelines for transcript quantification and microbial community profiling. We also plan to explore applying TahcoRoll in other sequencing applications, such as error correction for long read sequencing and profiling epigenetic marks through Bisulfite-Seq, to address various challenges in computational biology.

CHAPTER 8

Identifying Users behind Shared Accounts

In the previous chapters, we learn how to derive and manage robust representations for different heterogeneous data, but the knowledge in those representations could be disconnected across different data types. As an example to address this challenge, in this chapter, we present our work to unify the knowledge in diverse domains by aggregating them into a heterogeneous network for interdisciplinary knowledge discovery. We also demonstrate the impacts of our work in user identification for online streaming services.

8.1 Introduction

Online streaming services, such as Netflix¹ and Spotify², have become popular and accumulated massive user bases. Premium users have the privileges to enjoy high-quality contents and real-time streaming events, but these services usually come at a fee. Because of the membership fee of the premium accounts, it is not rare that users share a premium account to split the cost between themselves. However, illegal sharing may compromise not only the service provider's financial interests but also the service quality in general. First, account sharing implies loss of potential customers who may bring additional revenue to the service provider. Second, current customer profiling and recommendation systems operate under the assumption that each account is used by a single user and hence cannot accurately model individual user preferences from a mixture of activities by multiple users. This may impair its ability to provide high quality recommendations to users. Consequently, unsatisfied users

¹Netflix: <https://www.netflix.com/>

²Spotify: <https://www.spotify.com/>

may decide to switch to other providers.

To detect account sharing and enhance the quality of recommender systems in the presence of account sharing, we aim to identify individual users behind shared accounts. The goal of our work is three-fold. First, given a list of registered accounts, along with the corresponding *session logs* that record the activities of the accounts, we aim to accurately identify the set of users behind each account based on its session activities from the set of users who are using this account. Then we can accordingly predict whether an account is shared by multiple users. Second, given a newly-coming session issued by a certain account, we aim to identify the corresponding user from the identified users of that account. Third, we will enhance the performance of item recommendation by integrating account-level and user-level item recommendation. The session log of an account contains lists of entries. Each entry records the item requested and the timestamp of such request. We organize the log of each account into a list of consecutive sessions. In addition, each item may be associated with several metadata attributes (e.g., a song may have genres, artists, albums, and published years).

It has been shown in the literature that modeling multi-user behaviors in shared accounts [4, 323, 337, 355, 373] and session-based recommendations [150, 319] successfully improve the performance of item recommendation. However, these studies do not attempt to identify individual users. To the best of our knowledge, Zhang et al. [363] is the first and only attempt to identify users in shared accounts by a specialized subspace clustering, which is employed as a baseline in our experimental studies. In addition, the information of metadata is not taken into account in their approach.

Since we do not know the accounts that may be shared by multiple users and the users that share the same account, we propose an unsupervised learning-based framework, *Session-based Heterogeneous graph Embedding for User Identification* (**SHE-UI**). The main idea is to model the preference of individual users via a novel technique of session embedding that learns a unique feature representation for each session. We first create a heterogeneous information network to represent the relationships among items and their meta information.

Then, by applying a specialized random walk mechanism, the feature representation of each node can be derived using the skip-gram learning architecture [240, 263]. Subsequently the item-based session embedding is learned through the node embedding. For each account, the user identification problem is then mapped to the problem of session clustering. We develop a clustering algorithm based on Affinity Propagation [120] to simultaneously determine the number of clusters and group sessions into clusters. Each cluster represents the sessions issued by the same user, and the number of clusters represents the number of users sharing this account. For any incoming session of this account, we may find the potential user who issues the session by computing its representation (from the first few items in the session) in the space of session embedding and finding its nearest cluster. Last, to boost the performance of recommender systems, we propose a hybrid recommender, termed **AURec**, which combines conventional account-level and user-level (derived by SHE-UI) item recommendation.

We summarize the contributions of this work in the following.

- We propose to deal with two research tasks: (1) identifying users behind a shared account based on historical sessions and metadata of the items, and (2) given a new session initiated by a multi-user account, identifying which user issued this session. The former can benefit the service provider to detect multi-user accounts so that new pricing strategies can be established, while the latter boosts the performance of item recommendation. It is also worth mentioning that no prior knowledge about the mappings between users and accounts are given here.
- We develop an unsupervised framework SHE-UI that cannot only identify users in shared accounts, but also learn the preferences of individual users. Through a novel session embedding technique, SHE-UI effectively learns feature representations from a heterogeneous graph that represents the relationships between items and their metadata.
- Experiments conducted on two large-scale datasets of online streaming services, Last.fm and KKBOX, demonstrate that SHE-UI clearly outperforms existing item-based and embedding-based methods on both tasks of user identification. A study of parameter sensitivity also manifests the robustness of SHE-UI.

- Based on the identified users behind accounts, we devise a hybrid recommender AU-Rec that combines account-level and user-level item recommendation. Experiments on KKBOX data show that AURec is able to significantly outperform the state-of-the-art account-level recommendation methods by 39% in terms of Precision@1.

8.2 Problem Statement

In this section, we first formally define the problem of user identification in online streaming services. Let I be the set of items, e.g., songs and movies. For each item $i \in I$, it may have multiple attributes, e.g., artist(s) and genre(s) of a song, denoted by M_i , as its metadata. We denote the collection of all metadata as $M = \bigcup_{i \in I} M_i$. Here the metadata can be any discrete attribute that can describe items. Relationships may exist between attributes in metadata. For example, an album m_j may include a song performed by an artist m_k . These relationships can be denoted as $R = \{(m_j, m_k) \mid (m_j, m_k) \in M^2, m_j \neq m_k, m_j \text{ is related to } m_k\}$.

Let A be the set of accounts. For each account $a \in A$, the set of users of the account is denoted as $U(a)$, which is unknown in advance. In the following discussion, we refer to accounts with only one user as *single-user accounts* and the remaining ones as *multi-user accounts*. The activity log of each account $a \in A$ contains a sequence of sessions $S(a)$. Each session $s \in S(a)$ is a sequence of T_s items (successively requested by a user u_s without a long period of inactivity): $s = \langle i_1, i_2, \dots, i_{T_s} \rangle \in I^{T_s}$. For multi-user accounts, we assume that each session may be issued by one user. Note that the actual user u_s of every session s is also unknown to the system. The two goals of this work are as follows:

1. **User Identification in Past Sessions (UI-Past):** Given a set of accounts A and their corresponding sessions, for each account $a \in A$, the first goal is to group sessions $S(a)$ into K_a clusters (i.e., users), $C(a) = \{c_1^a, c_2^a, \dots, c_{K_a}^a\}$ such that the sessions from the same user are grouped into the same cluster where $1 \leq K_a \leq |S(a)|$. K_a is also unknown and needs to be estimated from the data. In other words, we would like to estimate the ideal clusters $C^*(a)$ grouping sessions by their actual users.

2. **User Identification for New Sessions (UI-New):** Given the identified users $C(a)$ of an account a , for any new incoming session $s \notin S(a)$ of account a , the next goal is to predict which user is the actual issuer of this session as early as possible. Based on the first few items in s , we want to identify the cluster c_k^a to which s belongs.

8.3 Session-based Heterogeneous graph Embedding for User Identification

In this section, we present the proposed framework, Session-based Heterogeneous graph Embedding for User Identification (SHE-UI).

8.3.1 Framework Overview

Figure 8.1 shows the framework of SHE-UI. A heterogeneous graph is constructed to represent the relations among items and metadata. We first compute node embeddings from which we generate session embeddings. Then an algorithm based on affinity propagation [120] is proposed to simultaneously determine the cluster number K_a for each account a and to group sessions $S(a)$ into clusters.

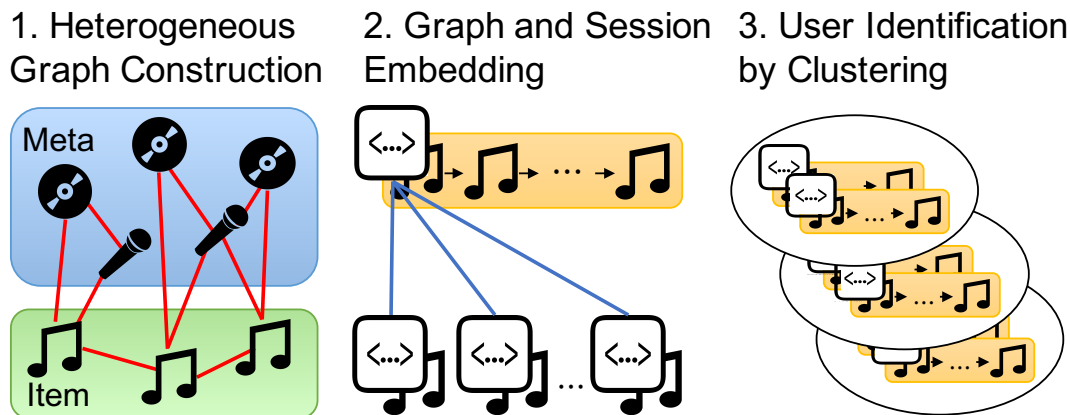


Figure 8.1: The framework overview of SHE-UI.

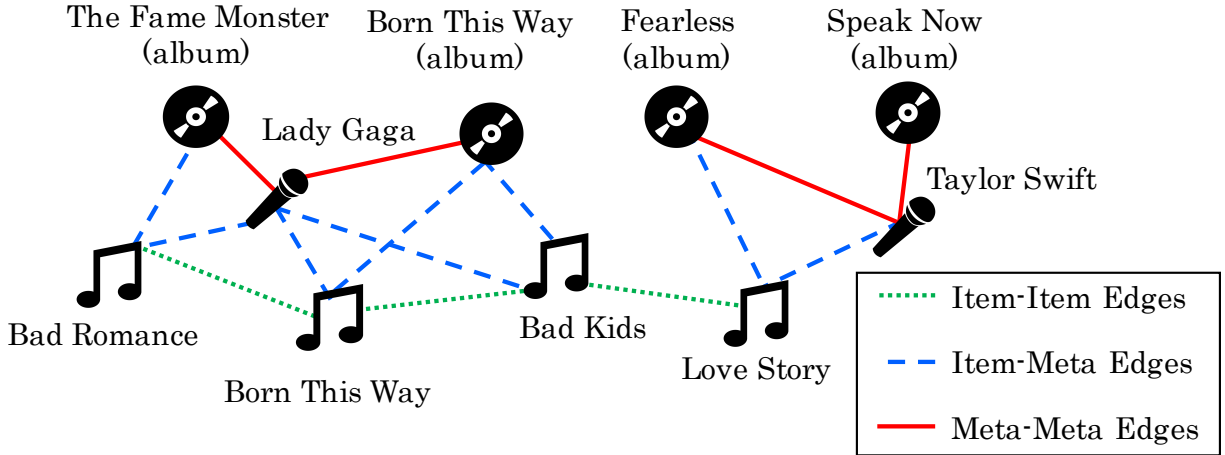


Figure 8.2: An example of heterogeneous graph construction with a session that begins with three songs by Lady Gaga and ends with a song by Taylor Swift.

8.3.2 Node Embedding in Heterogeneous Graph

As the first stage of SHE-UI, we encode items and metadata into an undirected graph $G = (V, E)$ with heterogeneous nodes $V = \{I, M\}$ and edges E . Specifically, each node in the graph represents an item or an attribute in metadata; each edge represents a relationship between nodes. The set of edges E can be constructed in the following three manners. Figure 8.2 shows an example of these three types of edges within a session.

1. **Item-Item Edges:** Any two items consecutively requested in the same session are linked to each other;
2. **Item-Meta Edges:** The node of each item is connected to nodes representing attributes in its metadata;
3. **Meta-Meta Edges:** Each meta relationship $(m_j, m_k) \in R$ is represented by an edge between the two corresponding nodes.

Given the heterogeneous graph $G = (V, E)$, we will first compute a low-dimensional feature representation for each node. We aim to find a mapping function $f : V \rightarrow \mathbb{R}^d$ from nodes to their low-dimensional feature representations, where d is the number of dimensions of the feature representation, and f can be considered as a $|V| \times d$ matrix, where $|V|$ denotes the number of nodes.

8.3.3 Learning Node Features

We extend the skip-gram architecture [240, 263] from natural language processing to learn the feature representations of nodes in a heterogeneous graph. In natural language processing, the skip-gram architecture learns relations between words and their context. Here each node in the network is treated as a word, and some random walk paths are sampled as sentences. We define $N_S(v) \subseteq V$ as the neighbor nodes for each node v via a sampling method S . Here we use a sampling method based on normalized random walk, which is presented in Section 8.3.4. The skip-gram model is extended to optimize the log-likelihood of the observed $N_S(v)$, conditioned on node v 's feature representation $f(v)$ as follows:

$$\max_f \sum_{v \in V} \log P(N_S(v) | f(v)).$$

To make optimization more efficient, we adopt two standard assumptions [135]. First, we assume that, given node v 's feature representation, v 's neighbor nodes $N_S(v)$ can be observed conditionally independent of each other. Then $P(N_S(v) | f(v))$ can be factorized by the neighbor nodes as follows:

$$P(N_S(v) | f(v)) = \prod_{n \in N_S(v)} P(n | f(v)).$$

Second, we assume that any pair of neighboring nodes symmetrically affect each other in the d -dimensional space of feature representation. Therefore, given a node v , the conditional likelihood of every neighbor node $n \in N_S(v)$ can be modeled as a softmax unit [12] by reversing the previous formula:

$$P(n | f(v)) = \frac{\exp(f(n) \cdot f(v))}{\sum_{u \in V} \exp(f(u) \cdot f(v))}.$$

Algorithm 8.1: LearningNodeFeatures(G, t, d, l)

Input: the graph $G = (V, E)$, walks per node t , the feature dimensions d , the fixed length l

Output: the embedding function f

```
1  $walkset = \emptyset$ 
2 for  $iter = 1$  to  $t$  do
3   foreach  $v \in V$  do
4      $W = \text{NormalizedRandomWalk}(v, l, G)$ 
5      $walkset = walkset \cup \{W\}$ 
6  $f = \text{StochasticGradientDescent}(walkset, d)$ 
7 return  $f$ 
```

With these assumptions, the objective function can be rewritten as:

$$\max_f \sum_v \left(-\log Z_v + \sum_{n \in N_S(v)} f(n) \cdot f(v) \right),$$

where $Z_v = \sum_{u \in V} \exp(f(u) \cdot f(v))$ can be approximated by negative sampling [241]. In addition, this objective function can be optimized by stochastic gradient descent [42]. Algorithm 8.1 presents the detailed procedure to learn node features. Each node in the graph will be treated as the source of t random walks. These generated $t \times |V|$ random walks will be exploited to learn the node features by stochastic gradient descent. After learning node features, we will use the feature representations of items to compute session embeddings in Section 8.3.5.

8.3.4 Normalized Random Walk

We now present our sampling method based on normalized random walk. Random walk is one of the most popular solutions for graph-based embedding [135, 264]. However, traditional random walk that treats every edge equally important is not suitable for the heterogeneous graph constructed above in which popular items and metadata attributes may have much higher node degrees than the rest. For example, popular songs can be requested by more than ten thousand sessions while others are requested by ten sessions. Consequently, a

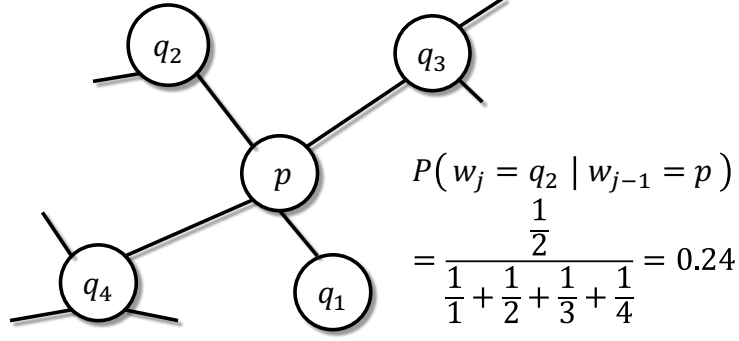


Figure 8.3: An illustration for the procedure of normalized random walks. The walk is going to be transited from p and evaluating transition probabilities of neighbor nodes.

random walk is likely to be confined to a small number of high degree nodes and ignores the rest of the graph. To solve this problem, we use the normalized random walk to learn node embedding in the heterogeneous graph.

Consider a source node $r \in V$ and a random walk W from r with a given length l . Let w_j be the j -th node in the walk, where $w_0 = r$ is the first node in W . The degree of node v is denoted as $d(v)$; $N(v)$ denotes the set of neighbors of node v . Then node w_j can be generated by the normalized probability $P(w_j \mid w_{j-1})$:

$$P(w_j = q \mid w_{j-1} = p) = \begin{cases} \frac{1/d(q)}{Z_p} & \text{if } (p, q) \in E \\ 0 & \text{otherwise} \end{cases},$$

where $Z_p = \sum_{q' \in N(p)} \frac{1}{d(q')}$ is the term for normalization. Figure 8.3 shows an example. Node p has four neighboring nodes $N(p) = \{q_1, q_2, q_3, q_4\}$; the degree of q_3 is $d(q_3) = 3$, and the probability of transiting to q_2 is $P(w_j = q_2 \mid w_{j-1} = p) = 0.24$. The detailed procedure of generating a l -length random walk from r is provided in Algorithm 8.2. We will investigate how l affects the performance in Section 8.4.3.

8.3.5 Item-based Session Embedding

Since each session $s = \langle i_1, i_2, \dots, i_{T_s} \rangle \in I^{T_s}$ consists of a sequence of items, the feature representation of a session can be computed by a combination of item features. A naïve way

Algorithm 8.2: NormalizedRandomWalk(r, l, G)

Input: the source node r , the fixed length l and the graph G

Output: the random walk W

```
1  $w_0 = r$ 
2  $W = [w_0]$ 
3 for  $j = 1$  to  $l$  do
4   | Draw  $w_j \sim P(w_j | w_{j-1})$ 
5   |  $W.append(w_j)$ 
6 return  $W$ 
```

to derive session features is to simply compute the average features over all item occurrences. However, a user’s affinity to an item may not be linearly correlated with the number of occurrences of the item in the session. For example, a user may play a song 100 times and another song 10 times. The affinity to the latter song is underestimated if we treat each play equally important.

To alleviate this problem, we model user’s affinity to an item in a session by a kernel function of the number of item occurrences in this session. It has been shown [132, 133] that item sequences modeled after user behaviors tend to follow a Poisson distribution. Then we can adopt a square-root function to approximate the variance-stabilizing transformation to model user’s affinity [112]. Let $\Gamma(s)$ be the set of distinct items in session s , and $Occ(s, i)$ be the number of occurrences of item i in session s . The feature representations of session s can be defined as follows:

$$f(s) = \frac{1}{\sum_{i \in \Gamma(s)} \sqrt{Occ(s, i)}} \sum_{i \in \Gamma(s)} \sqrt{Occ(s, i)} \cdot f(i).$$

These session features $f(s)$ can appropriately represent the characteristics of items in the session.

8.3.6 User Identification

After obtaining session features, we want to detect the number of users of each account a and group sessions by their actual issuers automatically. While most of the clustering

Algorithm 8.3: UserIdentification($S(a), f$)

Input: the set of sessions $S(a)$ and the embedding function f

Output: The set of cluster exemplars $C(a)$

1 Initialize X and Y as $|S(a)| \times |S(a)|$ matrices with zeros

2 **repeat**

3 **for** $j = 1$ to $|S(a)|$ **do**

4 **for** $k = 1$ to $|S(a)|$ **do**

5 $\Delta_{jk} = \max_{k' \neq k} \{Y_{jk'} + \text{Score}(s_j, s_{k'})\}$

6 $X_{jk} = \text{Score}(s_j, s_k) - \Delta_{jk}$

7 **for** $j = 1$ to $|S(a)|$ **do**

8 **for** $k = 1$ to $|S(a)|$ **do**

9 **if** $j \neq k$ **then**

10 $Y_{jk} = \min\left(0, \sum_{j' \notin \{j,k\}} \max(0, X_{j'k})\right)$

11 **else**

12 $Y_{kk} = \sum_{j' \neq k} \max(0, X_{j'k})$

13 **until** Convergence;

14 **return** $C(a) = \{s_k \mid \forall s_k \in S(a), X_{kk} > 0\}$

algorithms require the number of clusters K_a as an input parameter, we propose using affinity propagation [120] algorithm to automatically discover the appropriate clustering number.

Specifically, we propose to cluster these sessions via a message passing mechanism between sessions, in which the exemplars are found and considered as the cluster representatives. The algorithm passes messages between sessions and iteratively updates two $|S(a)| \times |S(a)|$ matrices: *responsibility* matrix X and *availability* matrix Y . The responsibility value X_{jk} represents how session s_k is suitable to be the exemplar of session s_j compared to other exemplars. The availability value Y_{jk} estimates how appropriate for session s_j to pick s_k as its exemplar. Both X and Y are log-probability matrices. At the beginning, they are initialized to zero. In each iteration, all elements in X are estimated by Y_{jk} and a score function $\text{Score}(s_j, s_k)$ between features of two sessions s_j and s_k . Here $\text{Score}(s_j, s_k)$ is defined as the L2-distance between two feature vectors. Then we update Y_{jk} by summing up responsibilities in X . We iteratively update X and Y until convergence. The sessions which remain positive responsibilities are the exemplars. The cluster number is the number of exemplars found in an account. This procedure is described in Algorithm 8.3. These

exemplars will be used for user identification in past and future sessions.

User Identification using Cluster Exemplars. Recall that Section 8.2 introduced two goals of user identification, UI-Past and UI-New. To identify users from past sessions (i.e., UI-Past), Algorithm 8.3 can directly output the clusters of past sessions issued by an account, and each cluster corresponds to a user. To predict the user of a new session (i.e., UI-New) in account a , we need to, from all users detected for account a from UI-Past, find the one who is most likely to issue the new session only using the first few (say, ρ) items in the new session. These ρ items are treated as a shorter session from which we derive its feature representation follow the same procedure in Section 8.3.5. Then we can obtain its corresponding exemplar and cluster assignment by computing the L2-distance between feature vectors. The cluster with the shortest distance to the given new session is considered as the corresponding user. Here, ρ should be a small integer because our task is to identify the user as soon as he/she issues a new session. We will also investigate how ρ affects the performance in Section 8.4.3.

8.3.7 Complexity Analysis

Here we analyze the time and space complexity of SHE-UI.

Time Complexity. It costs $O(|R| + \sum_{i \in I} |t(i)| + \sum_{a \in A} \sum_{s \in S(a)} T_s)$ time to construct the heterogeneous graph. Assume that the number of metadata $|t(i)|$ for each item i and the length T_s of each session s are small constants. It becomes $O(|R| + |I| + |S|)$, where $S = \bigcup_{a \in A} S(a)$ is the set of all sessions of all accounts. Then SHE-UI spends $O(t \cdot |V| \cdot l) = O(|I| + |M|)$ time on collecting normalized random walk paths, where t and l are also treated as small constants. To obtain feature representations of nodes, the stochastic gradient descent process takes $O(|V|^2) = O(|I|^2 + |M|^2)$. Finally, the affinity propagation-based method determines the cluster number and clusters sessions in $O(\sum_{a \in A} |S(a)|^2)$ time. In addition, user identification cluster each account independently, so the algorithm is parallelizable. In summary, the time complexity of SHE-UI is acceptable as $O(|R| + |I|^2 + |M|^2 + \sum_{a \in A} |S(a)|^2)$.

Space Complexity. The heterogeneous graph occupies $O(E) = O(|R| + |I| + |S|)$ to store the edges. The random walk paths need $O(|V|) = O(|I| + |M|)$. Node and session

features takes $O(|V| + |S|) = O(|V| + |I| + |M|)$ space. Finally, the affinity propagation costs $O(|S(a)|^2)$ space to create the responsibility and availability matrices for each account. If the algorithm runs sequentially, it takes $O(\max_{a \in A} |S(a)|^2)$; otherwise, the parallelized algorithm has to spend $O(\sum_{a \in A} |S(a)|^2)$ on storing matrices for all accounts simultaneously. Therefore, the space complexity of SHE-UI is $O(|R| + |I| + |M| + \max_{a \in A} |S(a)|^2)$ or $O(|R| + |I| + |M| + \sum_{a \in A} |S(a)|^2)$.

8.4 Experiments

8.4.1 Datasets and Experimental Settings

The experiments are conducted on two datasets:

- **Synthetic Last.fm (Last.fm).** Last.fm [53] provides publicly available datasets for music recommendation. The Last.fm-1K dataset contains streaming (or listening) history of 1K users from Feb 2005 to May 2009. Since Last.fm-1K does not provide any information about account sharing, we manually create *synthetic accounts* by merging several users' history together following a similar procedure to that in [323]. 25% of accounts have 1, 2, 3, and 4 users respectively. Each user only belongs to one account.
- **Real data from KKBOX (KKBOX).** The dataset comprises listening logs of 100K accounts from the KKBOX music streaming service from December 1, 2014 to November 30, 2015. In this dataset, we use the device ID as the bronze standard to evaluate the accuracy of our user identification. Sessions with the same device ID are treated as being issued by the same user. Figure 8.4 shows the percentage of accounts over different number of sharing users. More than 75% of accounts in the real-world dataset are shared by multiple users. Figure 8.5 further illustrates the percentage of session pairs sharing common items or metadata. We observe that, among all sessions of a given multi-user account, sessions issued by the same user are much more likely to share common items or metadata than that of different users.

Data Preprocessing. In online streaming services, the log of each account is a sequence

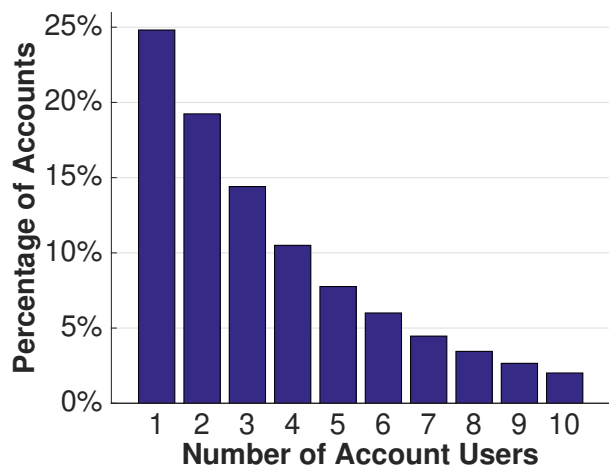


Figure 8.4: The percentage of accounts over different numbers of account users in the real-world KKBOX dataset.

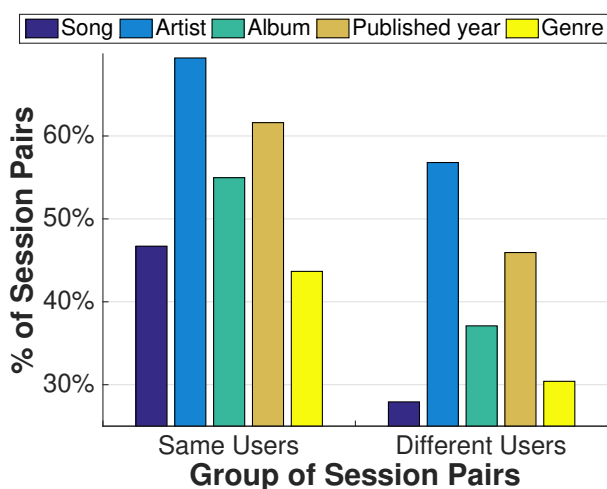


Figure 8.5: The percentage of session pairs sharing items or metadata within accounts in the KKBOX dataset. A pair of sessions can be issued by the same user or two different users.

Table 8.1: Statistics of Two Datasets.

(a) Session Information

	Last.fm	KKBOX
training sessions	209,313	10,783,556
testing sessions	209,925	10,782,507
accounts	370	88,399
unique users	922	343,723
items	314,763	564,164

(b) Metadata

Last.fm	
artists	60,410
KKBOX	
artists	43,157
albums	253,896
published years	77
genres	48

of entries, each of which contains a selected item with a timestamp. The log of each account can be partitioned into a list of sessions. We use 30 minutes of inactivity to define session boundaries. We exclude items of fewer than ten occurrences and accounts and sessions with fewer than ten entries. For the Last.fm dataset, the only available metadata are the artists of songs. For the KKBOX dataset, the available metadata include artists, albums, published years and genres. For both Last.fm and KKBOX datasets, we use 50% sessions for training and 50% sessions for testing for the UI-Past task (clustering sessions into groups of users sharing an account) and the UI-New task (identifying the user of an incoming session),

respectively. Table 8.1 shows the statistics of the two datasets after data cleaning and preprocessing.

Default Parameter Settings. Unless we specify otherwise, we set the length of random walks l to 5, and the dimension of features d to 512. For each node in the graph, 10 random walks are generated (i.e., $t = 10$ in Algorithm 8.1). For each session in UI-New, the first 5 items are used to derive session features (i.e., $\rho = 5$ in Section 8.3.6). The effects of these parameters are analyzed in Section 8.4.3.

Evaluation Tasks. There are three evaluation tasks as follows.

- **(1) User Number Estimation:** we examine whether SHE-UI can accurately determine the number of users using the same account (i.e., the number of clusters among sessions of an account). Note that this evaluation can be regarded as *Multi-user Account Detection*. That says, accounts with two or more estimated users can be treated as multi-user ones. We consider this task as a regression problem, and thus use *Mean Absolute Error* (MAE) and *Root Mean Squared Error* (RMSE) [165] as the evaluation metrics to measure the difference between the number of users $|U(a)|$ and the number of estimated clusters $|C(a)|$ for every account $a \in A$. We examine the performance of our approach with affinity propagation (AP) [120], which can automatically determine the number of clusters (i.e., users).
- **(2) Performance in UI-Past and (3) Performance in UI-New:** we aim to evaluate the performance of SHE-UI for the tasks of UI-Past and UI-New, compared with a series of baseline methods. To further evaluate the effectiveness of session embedding in SHE-UI, experiments were conducted two times, with and without giving the number of users (i.e., session clusters) as input. If the numbers of users are known, K-Means++ [14] is applied to cluster sessions; otherwise, Algorithm 8.3 is applied. We compare the performance of SHE-UI using three conventional clustering evaluation metrics, including *Normalized Mutual Information* (NMI) [301], *Macro F-Score* (MAF) and *Micro F-Score* (MIF) [269].

Baseline Methods. To evaluate the performance in UI-Past and UI-New, we compare SHE-UI with several state of the art item-based clustering and embedding-based clustering

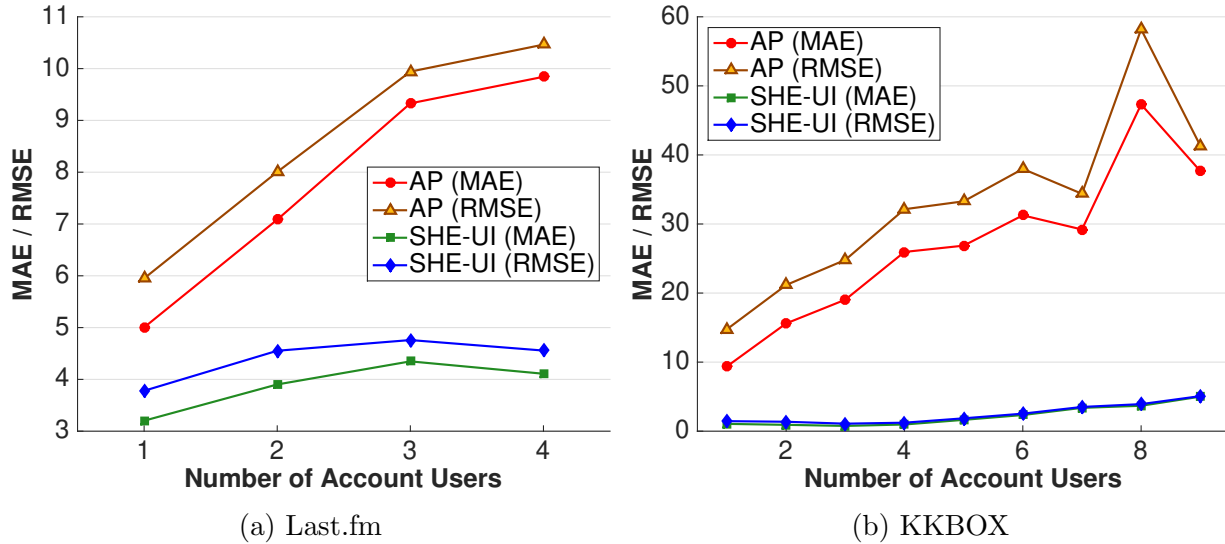


Figure 8.6: The performance of determining the cluster number. The lower MAE and RMSE indicate the better performance. All improvements are statistically significant at 95% confidence level by a paired t-test.

methods. The item-based clustering methods treat items in sessions as features, which include KMeans++ (KM) [14], affinity propagation (AP) [120] and subspace clustering (SS) [363]. The embedding-based clustering methods derive d -dimensional representations of sessions for clustering, which include word2vec (W2V) [241], LINE [313] and DeepWalk (DW) [264]. Note that these embedding-based clustering methods only derive the feature representations of items and need to apply our approach in Section 8.3.5 and Section 8.3.6 to obtain session embedding and the clustering results. We do not furnish a comparison with node2vec [135] since it is too time-consuming to compute probabilities for alias edges in a large dense graph.

8.4.2 User Identification Performance

User Number Estimation. Figure 8.6 shows the performance of detecting cluster numbers for accounts with different numbers of users. SHE-UI significantly outperforms AP, especially in accounts with multiple users. Such results prove the usefulness of session embedding in SHE-UI. The improvement is more significant in KKBOX dataset. KKBOX has much richer metadata that translate into a much larger and more diverse graph, which creates significant

Table 8.2: The results of user identification behind shared accounts. The higher values in metrics indicate the better performance. All improvements of SHE-UI against DW [264] are statistically significant at 95% confidence level by a paired t-test.

Dataset	Synthetic Last.fm						Real Data from KKBOX					
	UI-Past			UI-New			UI-Past			UI-New		
Metric	NMI	MAF	MIF	NMI	MAF	MIF	NMI	MAF	MIF	NMI	MAF	MIF
Known Numbers of Users												
KM [14]	0.2956	0.6109	0.7400	0.2802	0.6106	0.7400	0.3640	0.5710	0.6516	0.3286	0.5644	0.6592
SS [363]	0.2954	0.6109	0.7405	0.2793	0.6105	0.7403	0.3627	0.5707	0.6612	0.3258	0.5642	0.6585
W2V [241]	0.4865	0.7022	0.7982	0.4428	0.6823	0.7769	0.3828	0.5855	0.6524	0.3571	0.5739	0.6488
LINE [313]	0.2667	0.5611	0.6544	0.2622	0.5724	0.6768	0.3830	0.5874	0.6463	0.3456	0.5634	0.6183
DW [264]	0.5597	0.7372	0.8162	0.5148	0.7161	0.7947	0.3995	0.5976	0.6656	0.3587	0.5775	0.6419
SHE-UI	0.6108	0.7613	0.8393	0.5718	0.7455	0.8236	0.4281	0.6111	0.6804	0.3880	0.5948	0.6625
Unknown Numbers of Users												
AP [120]	0.1677	0.3413	0.3474	0.1546	0.4825	0.5408	0.1884	0.4828	0.4978	0.1783	0.5225	0.5569
KM [14]	0.1189	0.5842	0.7003	0.1061	0.5622	0.6697	0.1856	0.5264	0.5849	0.1516	0.5041	0.5642
SS [363]	0.1518	0.5838	0.6856	0.1312	0.5616	0.6582	0.1927	0.5312	0.5904	0.1841	0.5151	0.5851
W2V [241]	0.2981	0.6413	0.6587	0.2560	0.6148	0.6347	0.2081	0.5337	0.6025	0.1807	0.5149	0.5818
LINE [313]	0.0813	0.5641	0.6687	0.0964	0.5546	0.6552	0.1955	0.5365	0.6083	0.1010	0.4782	0.5394
DW [264]	0.3053	0.6286	0.6557	0.2669	0.5966	0.6244	0.2158	0.5508	0.6249	0.1941	0.5322	0.6024
SHE-UI	0.3375	0.6563	0.6782	0.3214	0.6323	0.6568	0.2426	0.5610	0.6309	0.2218	0.5451	0.6117

challenges for AP. We can further observe that both MAE and RMSE of AP fluctuate as the number of users increases. In contrast, SHE-UI consistently has lower errors. Such results demonstrate that SHE-UI can make accurate prediction even for accounts shared by a large number of users (e.g., 8, in Figure 8.6b).

Performance in UI-Past. Table 8.2 shows the results. Note that the result of AP is not reported under the section "Known Numbers of Users" since it cannot take a predefined cluster number. Also note that under "Unknown Numbers of Users", all compared methods except AP utilize our method (Section 8.3.6) to determine cluster numbers because they need it as an input parameter. We observe that SHE-UI consistently outperforms other methods in every metric for the real-world data and almost all metrics for the synthetic dataset, especially the improvement in NMI is more than 10% over the best competitor DW. Note that NMI values are relatively low (compared with MAF and MIF) since NMI is sensitive to the cluster size. As expected, when the cluster number is unknown, the performance of all methods degrades (comparing to the case where the cluster number is unknown). Nevertheless, SHE-UI can still perform reasonably well since it can accurately estimate the numbers of users in accounts (as proved in Figure 8.6). Furthermore, the embedding-based methods generally outperform item-based methods. Note that LINE performs the worst

in Last.fm dataset because the information of metadata is too sparse for LINE to capture appropriate features through edge sampling. Such result not only shows the effectiveness of embedding-based approaches but also verifies the capacity of SHE-UI to accurately derive session embedding.

Performance in UI-New. Table 8.2 shows the results of UI-New as well. We observe similar performance to that in UI-Past: SHE-UI outperforms other competing methods in every metric for the real-world dataset and almost all metrics for the synthetic dataset. For instance, SHE-UI outperforms the best competitor (i.e., DW) by around 15% of NMI in Last.fm and by around 12% in KKBOX on average for both cases with known and unknown numbers of users. In addition, the performance in UI-New is generally worse than that in UI-Past because only the first ρ items of a new session are used. In summary, SHE-UI is able to efficiently identify the user of any incoming session (among all users sharing an account). We will demonstrate its utility in improving the performance of online item recommendation customized for the identified user in Section 8.4.4.

8.4.3 Study of Parameter Sensitivity

This section presents how parameter settings in SHE-UI may affect its performance in user number estimation and user identification. We present only the performance using the KKBOX dataset, as we observe a similar performance on Last.fm.

Figure 8.7a and 8.7e exhibit the performance of user identification measured by NMI as a function of the length of normalized random walk (i.e., l in Algorithm 8.2) for the scenarios of known and unknown numbers of users respectively. Even though the performance generally improves as the length increases, we observe that it saturates when the length reaches five. Longer random walk does not necessarily warrant additional benefit. We thus set our default length to 5.

Figure 8.7b and 8.7f show the performance by varying the feature dimensions (i.e., d in Algorithm 8.1). The results show that, while higher dimensions can in general lead to more accurate user identification, the performance plateaus at 512 dimensions. We thus choose

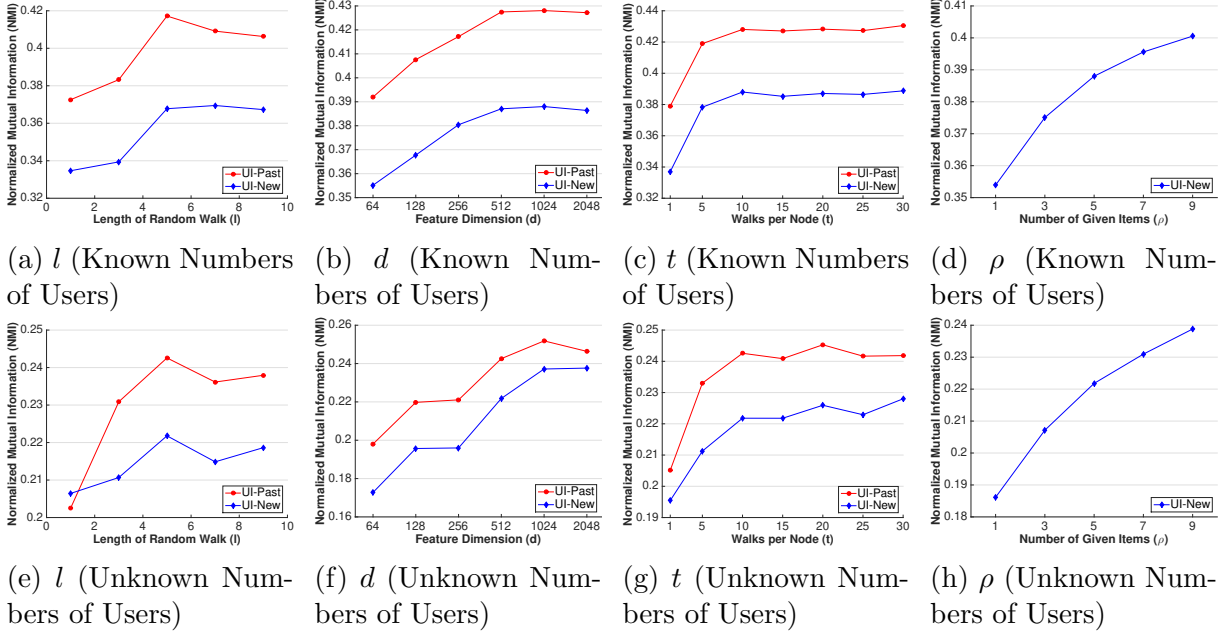


Figure 8.7: Results of Parameter Sensitivity Study.

512 as our default setting.

Figure 8.7c and 8.7g exhibit how the performance of SHE-UI is affected by the number of random walks generated for each node (i.e., t in Algorithm 8.1). The results are generally consistent with the intuition that more sampled walks lead to better performance. However, the performance saturates when the number reaches to 10, which is our choice of default value. This is because too many random walks starting from a node tend to bring duplicate information into feature representation.

Figure 8.7d and 8.7h demonstrate the performance of SHE-UI in UI-New by varying the number of items used to derive session features (i.e., ρ in Section 8.3.6). The performance is significantly better when the number of users is known than otherwise. It is also reasonable that more items seen in a new session lead to better performance of identifying the corresponding user. We set the default value to 5.

8.4.4 Item Recommendation with SHE-UI

Conventional recommender systems [35, 53, 149] do not attempt to distinguish users sharing the same account and thus can only recommend items to “accounts”, termed *account-level recommendation* (ARec) here. The preferences of individual users are not adequately captured. It is expected that by adding *user-level recommendation* (URec), we can effectively model the user preferences. Therefore, we propose a hybrid recommender that linearly combines the Account-level and User-level item RECommendation, which is termed AURec.

Let $\overline{R}_A(a, i)$ and $\overline{R}_U(u, i)$ be item i 's recommendation scores that ARec gives to account a and URec gives to the identified user u of account a , respectively. We employ the Bayesian personalized ranking matrix factorization (BPRMF) [278] model to compute the \overline{R}_A and \overline{R}_U scores. For a session s issued by the user u in the account a , AURec estimates the score of the item i by

$$\overline{R}_{AU}(a, u, i) = (1 - \alpha) \cdot \overline{R}_A(a, i) + \alpha \cdot \overline{R}_U(u, i),$$

where α is the parameter to control the weights of URec and ARec, and we set $\alpha = 0.6$ by default.

Evaluation Settings. The evaluation is conducted by using KKBOX data, in which items are songs. The split of training and testing is the same as in Section 8.4.1. We ran these experiments under the setting of UI-New with unknown numbers of users. We consider item recommendation as a ranking task. For each session s in the testing data, we want to examine how well we can predict the remaining items in the session based on the first 5 items. We first compute the \overline{R}_{AU} for every item and sort them by descending order of their \overline{R}_{AU} scores. We then examine the rankings of the items appear in the remainder part of the session s . Ideally, we want to see that these items are top ranked by the \overline{R}_{AU} scores. To quantitatively measure the recommendation performance, we use standard evaluation metrics [81], including *Mean Reciprocal Rank* (MRR), *Mean Average Precision* (MAP) and *Precision at k* ($P@k$) to compare AURec with four ARec recommender systems: recommendation by item popularity (PopRec), maximum margin matrix factorization (MMMF) [338], Bayesian

Table 8.3: Results of item recommendation with AURec.

	PopRec	MMMFM [338]	BPRMF [278]	CLiMF [292]	AURec
MRR	0.1242	0.1421	0.1353	0.1400	0.1727 (+22%)
MAP	0.0317	0.0331	0.0330	0.0337	0.0439 (+30%)
P@1	0.0597	0.0608	0.0577	0.0597	0.0846 (+39%)

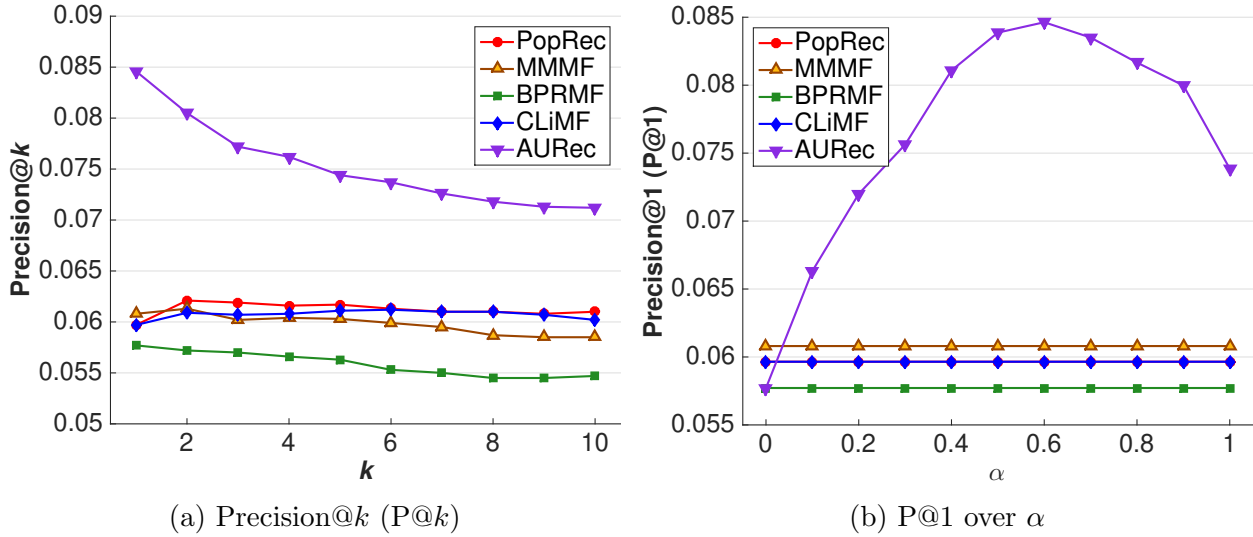


Figure 8.8: Results of recommendation by varying k and α .

personalized ranking matrix factorization (BPRMF) [278], and collaborative less-is-more filtering (CLiMF) [292].

Experimental Results.. Table 8.3 shows the results. It is clear that AURec significantly outperforms the best ARec competitor (i.e., MMMF) by a wide margin (from 22% to 39% in the three metrics). Such result reveals that user identification using SHE-UI can truly enhance the accuracy of item recommendation. To understand how parameters influence the recommendation performance, we further vary k in $P@k$ and α in AURec. The results are shown in Figure 8.8. We observe that AURec consistently outperforms others as k increases. In Figure 8.8b, the best performance is achieved at $\alpha = 0.6$ which demonstrates the need for combining URec and ARec. Using either ARec ($\alpha = 0$) or URec ($\alpha = 1$) along will produce substantially worse performance. It is worth noting that this task is extremely challenging. A session may be relatively short, consisting only a small number of items. A user may have broad interests and may not always play all of his/her favorite songs in one single session.

In the above setting, a favorite item is counted as a negative instance if it does not appear in the current testing session.

8.5 Conclusions and Discussions

This chapter investigates the problem of identifying individual users behind shared accounts in two settings: for historical data (UI-Past), and for incoming sessions (UI-New). An unsupervised learning-based framework, Session-based Heterogeneous graph Embedding for User Identification (SHE-UI), is proposed. Experiments conducted on KKBOX and Last.fm datasets demonstrate that SHE-UI can not only outperform the best competitor by at least 10% in UI-Past and 12% in UI-New in terms of NMI, but also significantly improve the performance of music recommendation by 39% measured by Precision@1. Accurate user identification is beneficial to both users and service providers. Users can enjoy a “true” recommendation while service providers can establish more desirable marketing strategies according to the behaviors of sharing accounts. The content providers (e.g. artists and publishers) can also gain insights in the taste and trend of different user groups.

The Session-based Heterogeneous graph Embedding (SHE) learns the feature representation for each session, which may be applicable to inferring account attributes, predicting session life-cycle, and detecting abnormal accounts, in addition to user identification. Moreover, the proposed SHE-UI can be applied to any activity and event sequences, allowing for incorporation of metadata. For example, one may study activity traces from smart devices and apply SHE-UI for activity recognition.

CHAPTER 9

Modeling Heterogeneous Data in Social Media Posts for Sponsorship Detection

In this chapter, we demonstrate the first example of our proposed aspect attention for interdisciplinary knowledge discovery as shown in Chapter 1. For social media posts with heterogeneous resources, we project them into a universal latent space and learn an aspect attention function for deriving ultimate aspect-attentive representations. As a real-world use case, we demonstrate significant improvements of our approach in sponsorship detection as a post classification task.

9.1 Introduction

Influencer marketing has been gaining significant attention from marketers as an essential advertising method recently [229]. As the rapid growth of the influencer marketing industry results in numerous paid advertisements in social media, the transparency issue of advertising posts has been raised. According to the regulations from the Federal Trade Commission (FTC) [76], the Advertising Standards Authority (ASA) [19], and the Organisation for Economic Co-operation and Development (OECD) [117], influencers are required to conspicuously disclose sponsorship when they publish paid advertisements. That is, mentioning brand names and the relationship between a mentioned brand and an influencer in paid advertisements, thereby having transparency in advertising posts. However, a noticeable number of influencers fail to disclose paid partnerships with brands in their advertising posts, either because they are not aware of the regulations [3] or because they are concerned

about lowering the effectiveness of the advertisement [110]. Surprisingly, the recent survey [3] reveals that only 52% of influencers and 60% of marketers have a good understanding of the regulation. This implies that the lack of legal knowledge and education for social media users can lead to social issues amid the rapid growth of social media. Figure 9.1 shows an example of a paid media where the influencer advertises the product of the brand in the absence of mentioning sponsorship.



Figure 9.1: An example of paid media that fails to disclose sponsorship. Despite the influencer advertises a product and mentions a certain brand name, no sponsorship is disclosed.

The sponsored posts without disclosing the sponsorship may cause the following problem. Audiences will be increasingly skeptical toward the influencers' posts, and hence influencers lose the trust. Influencer marketing can only become effective when people think that influencers are trusted sources of information [223]. Furthermore, the lack of transparency in the advertising posts can negatively impact brand image [110]. For the steady growth of the influencer marketing industry with proper advertising practice, the FTC has monitored and warned a few famous celebrities in social media who violated the endorsement regulations. However, it is impractical to monitor the millions of influencers on social media.

Although influencer marketing has gained noticeable attention recently, only a limited number of studies focused on sponsorship disclosure in influencer marketing. Some previous works examine the effect of the presence of sponsorship in social media [110, 302, 354] and suggest that disclosing sponsorship helps audiences to identify the post as an advertisement but lowers purchase intention. Moreover, Wojdyski et al. [342] attempt to measure the sponsorship transparency of paid advertisement by considering audiences' perceptions. While the previous works discuss the importance of detecting undisclosed sponsorship of social media posts, no study has proposed a method yet. Additionally, the previous works solely

rely on a small number of survey results, thereby lacking evaluation with a large dataset.

In this chapter, we propose a learning-to-rank based model, *Sponsored Post Detector (SPoD)*, that can detect sponsorship of social media posts. Our model incorporates three different aspects (i.e., modalities) on social media including graph, text, and image to represent the social media posts. We first employ the Graph Convolutional Networks (GCNs) [198] to leverage the characteristics of posts and the social relationship among influencers and brands. To adopt GCNs, we construct a heterogeneous network that connects influencers, posts, and brands. Besides the graph features, we also generate image and text features of each post to further describe the characteristics of the posts. Particularly, we use the pre-trained Inception-V3 model to obtain the image object features which have 1,000 categories [310] and utilize BERT [94] to create contextualized features of social media post captions. Moreover, we apply attention [321] over the three sets of features to estimate the importance of each aspect of social media posts, thereby utilizing more important aspects to detect hidden sponsorship. In addition to the attentive post features, we conduct a manifold regularization method to optimize the model performance. More specifically, we propose to exploit posting time and mentioned brands from social media posts for temporal regularization. For example, we place more weight on posts created at similar times and mentioning the same brand, that is, posts that likely belong to the same marketing campaign. With the proposed temporal regularization, our model takes the attentive post features as input to rank given social media posts by their sponsorship scores.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first attempt to rank social media posts by their sponsorship scores. We believe that our proposed SPoD can be beneficial for marketers and government organizations to find brands and influencers in violation of endorsement guides [19, 76, 117], thereby protecting consumers. Besides, social media platforms can utilize SPoD to help users recognize the sponsorship disclosing regulation. As SPoD exploits the prevalent social media features, our model can be practically adopted to any social media.

- We propose a novel learning-to-rank model, SPoD, that incorporates with texts, images, and relationships among influencers and brands. Moreover, we estimate the importance of the different modalities by using attention to acquire decent post representation. Besides, we employ the redundancy between the posts and their published times to learn temporal sponsored relationships. We conduct extensive experiments on a real-world dataset collected from Instagram which is known as the most popular social media for influencer marketing [229]. Our extensive experimental results demonstrate that SPoD improves performance by 54.3% in detecting undisclosed sponsorship compared to the best baseline method.
- Our analysis further reveals that the text features significantly improve the ranking performance. We find that contextualized features improve SPoD by 53.8%. We also observe that SPoD properly ranks the posts which are in various caption lengths. Particularly, SPoD obtains 216% improvement over the state-of-the-art text baseline method with very short caption posts by taking advantage of the graphical structural information.

9.2 Problem Statement

In this section, we formally define the goal of this paper. Suppose we have a set of posts $P = \{p_n\}_{n=1}^{|P|}$ published by a set of users $U = \{u_m\}_{m=1}^{|U|}$. Each post can be represented as $p_n = (t_n, a_n, b_n, l_n)$ where t_n , a_n , b_n , and l_n denote text, image(s), mentioned brand(s), and posting time of the post, respectively. Note that a post mentions at least one brand where the brand is in the set of brands $B = \{b_k\}_{k=1}^{|B|}$. Given text and images of a post, we extract text features and image features, $\mathbf{X}^T \in \mathbb{R}^{n \times f}$ and $\mathbf{X}^I \in \mathbb{R}^{m \times g}$, respectively, where f and g are the numbers of features. Moreover, we have graph features, $\mathbf{X}^G \in \mathbb{R}^{n \times h}$, where h is the number of features. The graph features can be generated from a heterogeneous network $\mathcal{G} = (E, V)$ where the vertices $V = (U, P, B)$ are composed of users, posts, and brands, respectively. Given a set of posts P , we aim to rank the posts by learning the distinctive features of sponsored posts so that sponsored posts with the absence of sponsorship disclosure can be discovered.

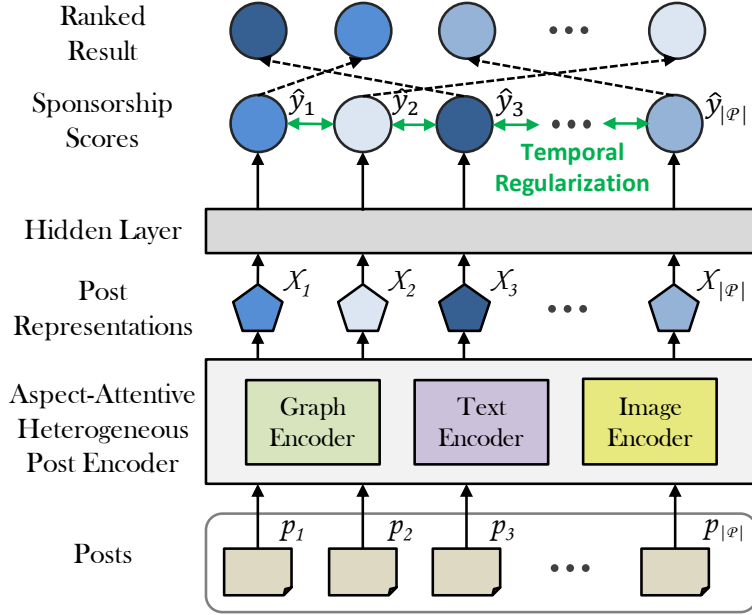


Figure 9.2: The overall framework of the SPoD. The aspect-attentive heterogeneous post encoder generates post representations. The estimated sponsorship scores are optimized by conducting temporal regularization for detecting sponsored posts.

9.3 Methodology

9.3.1 Framework Overview

Here we briefly give an overview of our proposed framework, SPoD, as shown in Figure 9.2. To leverage multimodal inputs of social media posts, we utilize three encoders, including graph encoder, text encoder, and image encoder. The graph encoder takes the heterogeneous network that includes users, posts, and brands as an input. Moreover, each node in the graph has a set of features as contextual representations to indicate the entity characteristics. Based on the heterogeneous structures of different entities and their features, graph convolutional networks (GCNs) are applied to derive appropriate node representations. In addition to GCN-encoded features, the text and image of each post are encoded by a contextualized text encoder and an image encoder, respectively. We then apply attention [321] over the three sets of features to estimate their importance and generate the post representations. The sponsorship scores of candidate posts are then computed based on all of the corresponding features and ranked for discovering sponsored posts. Finally, we optimize the sponsorship

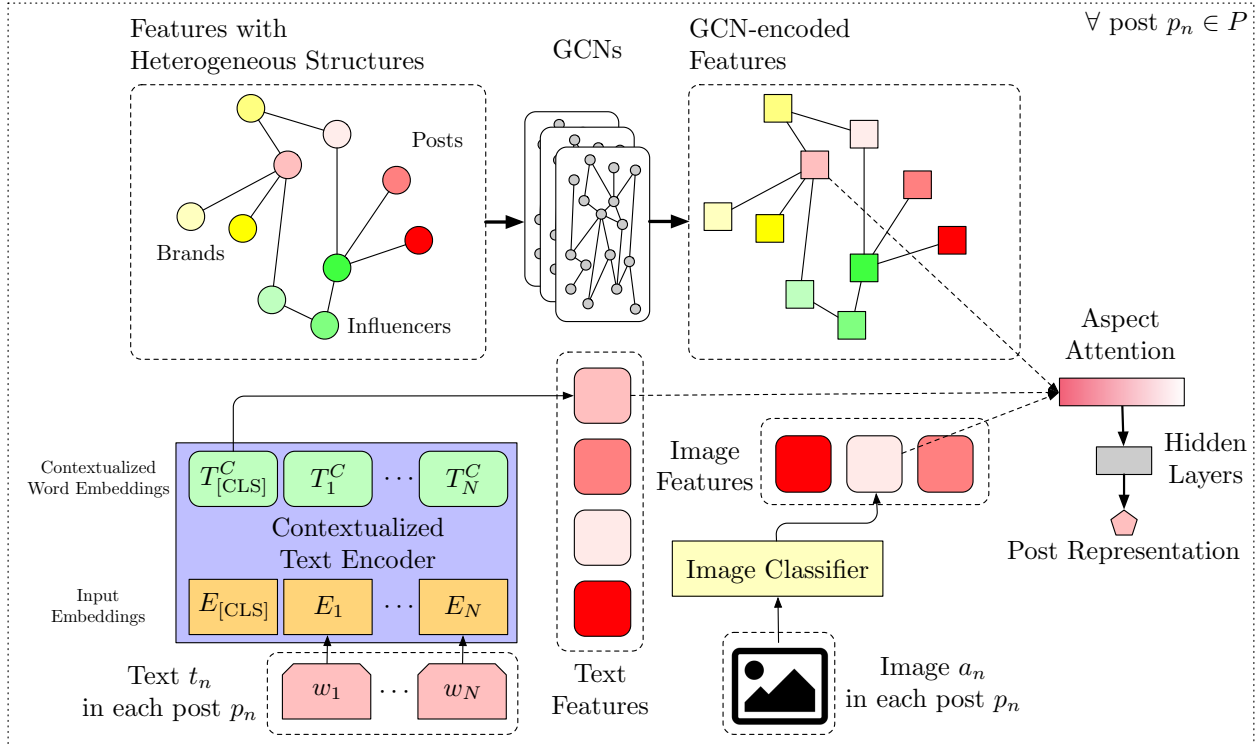


Figure 9.3: The illustration of the post encoder using heterogeneous information.

scores by conducting temporal regularization based on posting time and mentioned brands from the input post set.

9.3.2 Aspect-Attentive Heterogeneous Post Encoder

To acquire decent representations of posts, we utilize three encoders including the graph, text, and image encoders to capture knowledge from different aspects, and apply aspect-attention over the three sets of features as shown in Figure 9.3.

9.3.2.1 Graph Encoder

To model posts with the graphical structure, we first construct a heterogeneous network and then apply graph convolutional networks (GCNs) [198] to derive GCN-encoded features for each candidate post.

Heterogeneous Network Construction. To construct a heterogeneous network, we con-

sider three different entities including posts, users (i.e., influencers), and brands mentioned in posts. Note that the constructed heterogeneous network in our framework can be flexibly expanded with any additional relevant entities. The edges in the heterogeneous network indicate the interactions between entities behind nodes. The node of each post is linked to the node of the author user. If a brand is mentioned in a post, the post node has an edge to the brand node. Note that since more than one brand can be mentioned in a post, a post node can have multiple edges to brand nodes. More specifically, the edges of the network are represented by a sparse matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $A_{ij} = 1$ if the i -th and j -th nodes are connected; otherwise, $A_{ij} = 0$.

Node Features. Each node in the network has a set of features while all of the features can be represented as

$$\mathbf{Z} = [\mathbf{Z}^P; \mathbf{Z}^U; \mathbf{Z}^B] \in \mathbb{R}^{N \times d},$$

where \mathbf{Z}^P , \mathbf{Z}^U , and \mathbf{Z}^B are the features of nodes for posts, influencers, and mentioned brands, respectively; N and d are the number of all nodes in the network and the number of features for each node, respectively. The detailed features in this paper are defined in Section 9.3.2.5.

Graph Convolutional Networks To leverage the knowledge of structural information, we propose to apply GCNs to encode node representations with both node features and network structures. First, the adjacency matrix \mathbf{A} is transformed into a normalized adjacency matrix $\hat{\mathbf{A}}$ as follows:

$$\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}},$$

where \mathbf{D} is the diagonal matrix of node degrees. GCNs can then be operated based on the normalized adjacency matrix $\hat{\mathbf{A}}$ and the feature matrix \mathbf{Z} .

To model complicated network structures, we consider multi-layer GCNs by propagating information through different layers. Formally, the outputs of the i -th layer in GCNs, $H^{(i)} \in$

$\mathbb{R}^{N \times k}$, can be computed as follows:

$$\mathbf{H}^{(i)} = \sigma \left(\hat{\mathbf{A}} \mathbf{H}^{(i-1)} \mathbf{W}^{(i-1)} \right),$$

where k is the number of hidden dimensions in GCNs; $\mathbf{H}^{(i-1)}$ is the outputs of the previous layer; $\mathbf{W}^{(i-1)}$ is a matrix of layer-specific trainable weights; $\sigma(\cdot)$ is a nonlinear activation function. Note that $\mathbf{H}^{(0)} = \mathbf{Z}$ as the base case. Finally, the GCN-encoded representations \mathbf{X} can be computed by concatenating the outputs of different layers as follows: $\mathbf{X}^{\mathbf{G}} = [\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \dots, \mathbf{H}^{(L)}]$, where L is the number of layers in GCNs.

9.3.2.2 Text Encoder

To model contextualized knowledge from text in posts, we encode the text t_n in a given post p_n . Instead of learning from scratch, we apply the pre-trained state-of-the-art neural language model, BERT [94]. Note that any potential language model can be applied to the text encoder. Given the text in a post t_n , a length j sequence of words $c = [c_1, c_2, \dots, c_j]$, BERT adds an initial token [CLS] to alleviate the positional bias in input embedding $c' = [\text{[CLS]}, c'_1, c'_2, \dots, c'_j]$. The input word sequences of n posts, $\mathbb{Q}^{(j'+1) \times n}$, then goes to the transformer \mathbf{F}_T with o layers. We generate the output of the o -th transformer $\mathbf{F}_T(\mathbf{F}_T^{o-1}) = D^o = [d_{[\text{CLS}]}^o, d_1^o, d_2^o, \dots, d_j^o]$ as the representation of each word in the word sequences. Finally, we only adopt the $d_{[\text{CLS}]}^o$ to form the text representation $\mathbf{X}^{\mathbf{T}}$.

9.3.2.3 Image Encoder

In addition to the graph and text, we use images attached in the posts since images are known as one of the most important factors in effectively advertising products in social media marketing [231]. For example, the influencer in Figure 9.1 holds the product in the image for advertising purposes. We apply the pre-trained Inception-V3 model trained with one-million images in 1,000 object categories [310] to avoid training images from scratch. Particularly, the image encoder takes input images from a post and generates a feature vector with g

dimensions where each dimension represents the probability that the image contains the corresponding object. That is a list of s_n images from a post p_n , $a_n = [a_{n1}, a_{n2}, \dots, a_{ns}]$. We use the maximum value for each dimension while aggregating the feature vectors as follows: $\mathbf{F}_I(a_n) = \text{max-pool}(\{a_{ni} \mid 1 \leq i \leq s_n\})$, where the function $\text{max-pool}(\cdot)$ remains the maximum value for each dimension over the feature vectors of s_n images. Finally, the image representation \mathbf{X}^I contains the image object vectors from n posts.

9.3.2.4 Aspect-Attention

To estimate the importance of features from different aspects, including the heterogeneous graph, texts, and images, we apply the attention mechanism over the three sets of features. We first apply a fully-connected hidden layer to each set of features to have the same dimension of the features as:

$$\mathbf{V}^G = \mathcal{F}_G(\mathbf{X}^G), \mathbf{V}^T = \mathcal{F}_T(\mathbf{X}^T), \mathbf{V}^I = \mathcal{F}_I(\mathbf{X}^I),$$

where $\mathcal{F}_G(\cdot)$, $\mathcal{F}_T(\cdot)$, and $\mathcal{F}_I(\cdot)$ are fully-connected layers for the graph, text, and image features, respectively. The sets of features from different modalities can be represented as:

$$\mathbf{V} = [\mathbf{V}^G, \mathbf{V}^T, \mathbf{V}^I].$$

The importance α_i of feature \mathbf{V}_i can be estimated as:

$$\alpha_i = \frac{\exp(\mathbf{r}_i \cdot \mathbf{r}^c)}{\sum_j \exp(\mathbf{r}_j \cdot \mathbf{r}^c)},$$

where $\mathbf{r}_i = \tanh(\mathcal{F}(\mathbf{V}_i))$ is the hidden representation of the feature \mathbf{V}_i ; $\mathcal{F}(\cdot)$ is a fully-connected layer; $\tanh(\cdot)$ is the activation function; \mathbf{r}^c is the context vector for importance estimation. The estimated score α_i is multiplied with corresponding features \mathbf{V}_i to get weighted values, and the representation of the post can be derived by taking all the weighted

Table 9.1: Node features that represent characteristics of each type of node in GCNs.

Category	Feature	Description
Node	Node Type	Node type in the heterogeneous network.
Influencer	Keywords	The normalized frequency of keywords.
	Followers	Number of followers.
	Followees	Number of followees.
	Posts	Number of published posts.
	Influencer Category	Major interest of the influencer.
Posts	Likes	Number of likes in a post.
	Comments	Number of comments in a post.
	Hashtags	Number of hashtags(#) in a post.
	Usetags	Number of usertags(@) in a post.
	Caption Length	Length of text in a post.
	Images	Number of images in a post.
	Posting Day	The day a post was published.
Brand	Followers	Number of followers.
	Followees	Number of followees.
	Posts	Number of published posts.
	Brand Category	Business type of the brand.

values of the graph, text, and image features as follows:

$$\mathbf{X} = \sum_i \alpha_i \cdot \mathbf{V}_i.$$

9.3.2.5 Node Features in GCNs

To represent characteristics of nodes in the network, we incorporate four types of node features, including node type, influencer, post, and brand as shown in Table 9.1. Note that our proposed framework is not limited to these features, therefore, any potential feature can be additionally applied to the model.

- **Node type features.** To indicate one of the three types of nodes, including post, influencer, and brand, we apply the one-hot coded node type feature in this category.
- **Influencer features.** We exploit the normalized frequency of keywords to capture textual patterns of influencers. We select the frequently used keywords based on their Chi-square

values. Note that we use the top 100 keywords in this study. We also use the number of followers, followees, and published posts features that represent the reputation of influencers [191]. Moreover, we exploit the major interest of the influencers such as Food and Interior [192].

- **Post features.** To represent post characteristics, we exploit the features which are widely used in any social media. We first obtain the numbers of likes and comments in a post which can represent the popularity of the given post. Since it is well known that people tend to avoid advertising [67, 190], such post popularity features can help provide a distinguishable representation of sponsored posts. We also employ the numbers of hashtags and usertags that are particularly used for disclosing names of brands, products, or marketing campaigns in paid advertisements [354]. Additionally, we use the number of images in a post while most social media accept multiple images in a post, and the day a post was published (e.g., Sunday, Monday) since publishing time affects the popularity of advertising posts in social media [244].
- **Brand features.** To characterize the brand nodes, we have the business type of the brands³. Additionally, we use the number of followers, followees, and published posts to measure brand awareness [354].

9.3.3 Sponsorship Estimation and Ranking

To estimate the sponsorship score of a post, all of the GCN-encoded features, text features, and image features can be useful because many aspects of the post are considered.

For a post i , all of the features are concatenated as the ultimate representation \mathbf{X}_i . The predicted sponsorship score \hat{y}_i of the post i can then be generated by a linear unit with a fully-connected hidden layer as follows:

$$\hat{y}_i = \mathcal{F}_p(\sigma(\mathcal{F}_h(\mathbf{X}_i))),$$

³<https://business.instagram.com/>

where $\mathcal{F}_h(\cdot)$ and $\mathcal{F}_p(\cdot)$ are two fully-connected hidden layers; $\sigma(\cdot)$ is a nonlinear activation function. Therefore, the candidate posts can be ranked by the predicted sponsorship scores.

9.3.4 List-wise Learning to Rank

Since our goal is to rank posts by their likelihood scores to be sponsored posts, it is intuitive to apply learning to rank approaches to deal with the problem. More specifically, in this paper, we modify the ListMLE [348], which is list-wise learning to rank approach that can benefit overall ranking performance.

Suppose that \mathbf{X} is the set of features for posts to be ranked; Y is the output space of permutations of the posts; P_{XY} is an unknown but fixed joint probability distribution of X and Y . If a ranking function can be represented by $\hat{\mathbf{y}} : X \rightarrow Y$, the expected loss $R(\hat{\mathbf{y}})$ to be optimized can be derived as follows:

$$R(\hat{\mathbf{y}}) = \int_{X \times Y} L(\hat{\mathbf{y}}(\mathbf{X}_i), \mathbf{y}) \partial P(\mathbf{X}_i, \mathbf{y}),$$

where $\mathbf{y} \in \mathbf{Y}$ is a permutation; $\mathbf{X}_i \in \mathbf{X}$; $L(\hat{\mathbf{y}}(\mathbf{X}_i), \mathbf{y})$ is the 0-1 loss between the ranked result $\hat{\mathbf{y}}(\mathbf{X}_i)$ and the position in the permutation \mathbf{y} such that

$$L(\hat{\mathbf{y}}(\mathbf{X}_i), \mathbf{y}) = \begin{cases} 1 & , \text{ if } \hat{\mathbf{y}}(\mathbf{X}_i) \neq \mathbf{y} \\ 0 & , \text{ if } \hat{\mathbf{y}}(\mathbf{X}_i) = \mathbf{y} \end{cases}$$

To make the training process more efficient, candidate lists with n labeled posts are sampled from the whole training space in each iteration. Given independently and identically distributed samples in a candidate list $S = \{(X_i, \mathbf{y}_i)\}_{i=1}^n \sim P_{XY}$, we minimize the empirical loss R_S as follows:

$$R_S(\hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^n L(\hat{\mathbf{y}}(\mathbf{X}_i), \mathbf{y}_i),$$

where \mathbf{y}_i is the ground truth permutation.

9.3.5 Temporal Regularization

Timing is important for publishing posts in influencer marketing [244], so the redundancy between different posts with similar published times can be leveraged to improve the performance for sponsorship estimation. In addition, brands usually hire a number of influencers for a marketing campaign at a time. For example, the posts that are published at a similar time and mention the same brand name are more likely to be sponsored posts. In this paper, therefore, we conduct manifold regularization [176] by using the redundancy between the posts and their published times. Formally, the regularization loss $Q(\hat{\mathbf{y}})$ can be defined as:

$$Q(\hat{\mathbf{y}}) = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n (|\hat{y}_i - \hat{y}_j| \times \frac{w_b(i, j)}{\max(|l_i - l_j|, 1)}),$$

$$w_b(i, j) = \begin{cases} 1 & , \text{ if } b_i = b_j \\ 10^{-1} & , \text{ if } b_i \neq b_j \end{cases},$$

where \hat{y}_i and \hat{y}_j are the estimated sponsorship scores of the posts p_i and p_j mentioning the brands b_i and b_j at time l_i and l_j ; $w_b(i, j)$ indicates the brand-based regularization weight. Note that the posting time difference, $|l_i - l_j|$, is measured in days. Finally, the ultimate objective for discovering sponsorship L can be a combination of two loss functions as;

$$L(\hat{\mathbf{y}}) = R_S(\hat{\mathbf{y}}) + w_l \cdot Q(\hat{\mathbf{y}}),$$

where w_l is the weight for manifold regularization.

9.4 Experiments

9.4.1 Experimental Dataset

Dataset Construction. To evaluate our proposed model, our dataset samples influencer posts from Instagram, which is the most popular social media platform for influencer marketing [229]. Note that we implement the data collection method in [192] and comply with the

Instagram policy⁴. To find posts that mention brand names, we first collect a set of brands on Instagram by searching branded content, i.e., sponsored posts. Note that Instagram provides the branded content tool for influencers to disclose sponsorship by showing a partnered brand name on the top of a post⁵. From the searched sponsored posts, we obtain 26,910 brand names. Next, we find brand mentioning posts that contain at least one brand name by searching user tags in the corresponding caption. To reduce noises in the dataset, we filter out a post if it is published by a user with less than 1,000 followers which is a generally required number of followers to be considered as an influencer⁶. Finally, we collect 1,601,074 brand mentioning posts that are published from 2013 to 2019 by 38,113 influencers. Note that the number of posts is exponentially grown over time; the average follower count of the influencers is 127,279.

Sponsorship Labeling. Since our goal is to find sponsored posts that do not disclose paid partnerships, we first classify the posts in the dataset into two classes, including “Sponsored” and “Unknown”. We label the posts as ‘Sponsored’ if the posts explicitly disclose sponsored relationships by using certain keywords. More specifically, a given post is labeled as ‘Sponsored’ if the post either uses the branded content tool from Instagram or has one of the following hashtags, #ad, #sponsored, and #paidAD, which are widely used hashtags for sponsorship disclosure in influencer marketing [110, 354]. The remaining posts that are not identified as sponsored posts are labeled as ‘Unknown’. That is, the ‘Unknown’ posts may contain non-sponsored and sponsored posts with no sponsorship disclosure. We label all posts in the dataset and finally have 221,710 ‘Sponsored’ posts and 1,379,364 ‘Unknown’ posts which account for 13.8% and 86.2%, respectively. After labeling the posts, we remove all of the sponsorship-related keywords and hashtags from the posts to prevent information leakage in the experiments. Therefore, our model can detect sponsored posts without relying on such keywords. To evaluate the performance of detecting sponsored posts from the

⁴<https://help.instagram.com/325135857663734>

⁵<https://business.instagram.com/a/brandedcontentexpansion>

⁶<https://www.digitalmarketing.org/blog/how-many-followers-do-you-need-to-be-an-influencer>

‘Unknown’ posts, we further manually investigate the unknown posts. The details of manual labeling procedure are described in Section 9.4.4.2.

Heterogeneous Network. We build the heterogeneous network by using the 1,601,074 posts which include 2,273,578 brand mentions. As a result, the network has 38,113 influencer nodes, 26,910 brand nodes, and 1,601,074 post nodes with 3,874,652 edges.

9.4.2 Experimental Settings

To measure the performance of the proposed SPoD, we treat the task as a one-class ranking problem and assign a relevance score for each post so that posts with higher scores are more likely to be sponsored posts. That is, the posts labeled as ‘Sponsored’ have relevance 1 while the ‘Unknown’ posts have the relevance 0. As SPoD ranks the candidate posts by their sponsorship scores, the relevances are used to evaluate the rank quality. More specifically, we use mean average precision (MAP), mean reciprocal rank (MRR), and average precision (AP) as our evaluation metrics.

We use TensorFlow [1] to implement our model. We split the dataset into three partitions for training, validation, and testing with a ratio of 7:1:2 by randomly selecting the posts. Therefore, the ratios of sponsored posts and unknown posts on three partitions are the same. Additionally, we ensure that the same influencers are not included across the training, validation, and testing sets to avoid information leakage from learning relationships between influencers and brands (e.g., a certain influencer repetitively advertises a certain brand). We tune the parameters with the validation set and set a single hidden layer with 128 hidden nodes. The learning rate and the dropout probability are set as 10^{-3} and 0.5, respectively. We set the regularization weight, w_l , as 10^{-4} .

9.4.3 Comparative Baseline Methods

We compare the performance of the proposed model with the baseline methods in three different categories, including *Ranking*, *Graph*, and *Text*.

Ranking Baselines. As our model applies the learning-to-rank approach, we apply the identical feature sets of the proposed model for the ranking baselines, therefore, we can evaluate the model capability for the ranking task of the proposed model. We deploy three ranking baseline methods as follows: *ListNet* (LN) [52] is a list-wise learning-to-rank algorithm that exploits gradient descent on neural networks to optimize a list wise loss function. *MART* [121] is a pair-wise learning-to-rank algorithm that uses gradient boosted decision trees for prediction tasks. *LambdaMART* (LM) [45] directly optimize rank cost functions by using gradient boosted regression trees based on *MART*.

Graph Embedding Baselines. The baseline methods in this category only exploit the graphical structure without other information. We deploy the LINE [313] and the GCN [198] as two graph baseline methods. The baselines use the heterogeneous network as input features and disregard the text and image features.

Text Modeling Baselines. In addition to the ranking and the graph baselines, we also have two text baseline methods since influencers usually reveal paid partnerships using text. As the baselines, we deploy two state-of-the-art language models, Embeddings from Language Models (ELMo) [266] and Bidirectional Encoder Representations from Transformers (BERT) [94].

9.4.4 Experimental Results

In this section, we evaluate the performance of our proposed SPoD compared to the baseline methods with the following two steps: (i) We first examine the sponsored post ranking performance without taking into account sponsored posts in the unknown posts. (ii) We then investigate highly ranked unknown posts to evaluate the performance of detecting hidden sponsored posts in the unknown posts.

Table 9.2: Performance comparison with the baseline methods. SPoD significantly outperforms all types of baseline methods. The temporal regularization and aspect-attentive components improve the ranking performance.

Method	MAP	MRR	AP@ <i>k</i>			
			10	100	1000	10000
LN [52]	0.250	0.500	0.714	0.643	0.487	0.380
LM [45]	0.269	1.000	0.867	0.451	0.461	0.395
MART [121]	0.290	1.000	0.507	0.398	0.432	0.421
LINE [313]	0.317	1.000	0.894	0.701	0.587	0.473
GCN [198]	0.370	1.000	0.935	0.744	0.709	0.566
ELMo [266]	0.352	1.000	0.926	0.751	0.714	0.608
BERT [94]	0.376	1.000	0.947	0.788	0.755	0.653
SPoD w/o regularization	0.558	1.000	1.000	0.967	0.956	0.902
SPoD w/o aspect-attention	0.573	1.000	1.000	0.973	0.960	0.913
SPoD	0.592	1.000	1.000	0.994	0.984	0.941

9.4.4.1 Sponsored Posts Ranking Performance

Table 9.2 shows the performance of the proposed SPoD and the baseline methods. We find that ranking baselines show poor ranking performance compared to the graph and the text baselines. Despite the graph and the text baselines only exploit the network features and the contextualized text features, respectively, these baselines outperform the ranking baselines which use all the proposed features. This is because the text features can be easily over-fitted over the complicated structures in ranking baseline methods. In other words, the ranking baselines fail to leverage the contextualized features. We also find that, in the graph and the text baselines, GCN and BERT have better rank quality than LINE and ELMo, respectively, which are adopted to our proposed post encoder.

Finally, our proposed model, SPoD, significantly outperforms all of the other baseline methods. More specifically, SPoD obtains a 57.45% improvement in mean average precision over BERT. Unlike ranking baselines, SPoD separates text and image features from GCN features and then applies attention over the features, thus effectively utilizing more important post features for ranking. Furthermore, SPoD shows a more robust ranking performance than other baselines. Note that average precision at 10,000 of SPoD is 0.941 while the performance other baseline methods tend to decrease if the ranked list size increases. Figure 9.4 shows

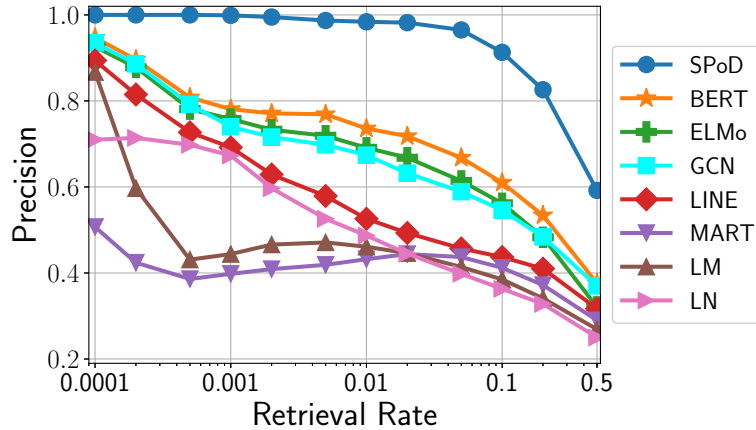


Figure 9.4: Precision at retrieval rates of the proposed model and the baseline methods. SPoD shows efficient and robust ranking performance compared to the baseline methods.

precision at retrieval rates of SPoD and the other baseline methods. While the precision of the baseline methods significantly drops as the retrieval rates increase, SPoD shows highly robust performance. That is, SPoD has high ranking accuracy even while finding a large number of sponsored posts. Furthermore, we perform an ablation study by removing the temporal regularization and the aspect-attention. As shown in Table 9.2, SPoD loses a 5.74% and 3.21% rank performances in the measure of MAP without using the proposed regularization and the aspect-attention, respectively. This suggests that the proposed regularization that exploits the redundancy between the posts and their published times plays an important role in discovering sponsorship of social media posts. This also reveals that the aspect-attention effectively generates post representations by finding more important features.

9.4.4.2 Detecting Unlabeled Sponsored Posts

Since the goal of SPoD is to detect sponsored posts which do not clearly disclose sponsored relationship with brands, we evaluate the performance of detecting such sponsored posts by investigating the ranking results from Section 9.4.4.1. To this end, we first extract a set of highly ranked unknown posts and then examine their images and captions to manually label the posts. To ensure the quality of our labeling procedure, two authors of this paper have carefully read and understood the FTC’s endorsement regulation, then investigated

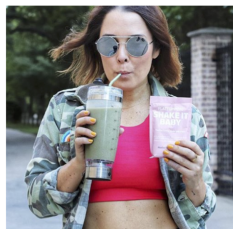
Table 9.3: Precision results on the top-ranked unknown posts. SPoD outperforms other baseline methods in detecting undisclosed sponsorship of the unknown posts.

Precision@ k	10	50	100	150	200
GCN [198]	0.600	0.540	0.420	0.380	0.355
BERT [94]	0.800	0.820	0.750	0.673	0.580
SPoD w/o regularization	1.000	0.960	0.910	0.873	0.810
SPoD w/o aspect-attention	1.000	1.000	0.950	0.920	0.860
SPoD	1.000	1.000	0.990	0.967	0.920

the top 200 unknown posts from each ranking result of SPoD, SPoD without regularization, BERT, and GCN. More specifically, we decide an unknown post as a sponsored post when an influencer exclusively promotes a certain product or service by expressing appreciation for sponsorship indirectly in text and holding the product to show brands in images. Cohen’s kappa coefficient of our labels is 0.784 which suggests that our labeling result is highly reliable [232]. Note that there are only 9 disagreements out of the 200 posts. We consider an unknown post as a sponsored post only if both labelers agree.

Table 9.3 shows the precision of detecting sponsored posts from the unknown posts. The result demonstrates that SPoD is very effective in discovering the sponsored posts with the absence of sponsorship disclosure compared to the baseline methods. Note that SPoD gains 58.6% and 159.2% improvements in precision scores at 200 over BERT and GCN, respectively by achieving 0.920 precision score. This implies that the proposed post representations are remarkably useful to identify sponsorship of social media posts even if the paid partnership is not explicitly disclosed. On the other hand, using only graph structural or contextualized information may fail to detect such posts. We also find that SPoD loses 11.96% and 6.52% performance in the precision at 200 without using the temporal regularization and the aspect-attention, respectively. This suggests that proposed temporal regularization significantly improves the performance of detecting undisclosed sponsorship by detecting advertising posts in the same marketing campaigns. This also reveals that the aspect-attention is useful to obtain more important knowledge from different aspects of social media posts.

Figure 9.5 showcases the example posts that are in the top 200 of the rank result of SPoD. The posts in Figure 9.5a and 9.5b are the sponsored posts with absence of sponsorship



You guys know I've got Game Day down, but what about the day after? @flattummyco has you covered following the big day with their yummy Shakes! They are packed with nutritious ingredients and only 130 calories - making them the perfect meal to give you a boost of energy and get you back on track. You guys really need to check them out, I'm hooked! #shakeitbaby

(a) Sponsored Post 1



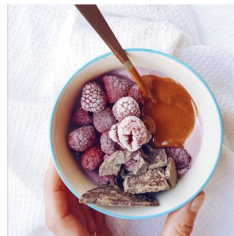
my chapped lips are thanking Raw Sugar for these amazing products; my new lip care go to! the lip scrub is perfect for refreshing my lips and the balm is so soothing + nourishing. the best part — for every product that you buy, Raw Sugar donates a fresh bar of soap to a family in need #rawthankyou stop by your local @target and try them yourself! #rawlovin

(b) Sponsored Post 2



The end of an era! Feels so good to say all the hard work has paid off - graduating @teessideuni with a First Class Honours Degree in Marketing. Feeling super proud!

(c) Non-sponsored Post 1



Rounding the weekend up like 🍒🍇🍓🍌🍦 Hope you've all had a wonderful weekend and are looking forward to the week ahead 😊 This little bowl was an absolute delight! We've got @alpro more fruit cherry yoghurt, @doisyanddam maple, toasted rice & pink salt chocolate snaps, @pipandnut chocolate orange almond butter, and very frosty frozen raspberries.

(d) Non-sponsored Post 2

Figure 9.5: Examples of successfully detected sponsored posts with absence of sponsorship disclosure, and highly ranked non-sponsored posts.

disclosure. In the sponsored posts, the influencers tend to describe details of the products by sharing their experience and recommending the products in the captions. In addition to the text, we observe the evidence of sponsorship in the image as they hold the products for advertising. However, the sign of sponsorship disclosure is not found from the text and hashtags. On the other hand, Figure 9.5c and 9.5d show the non-sponsored posts that are in high-rank positions. These posts may have been highly ranked due to an object in the images (e.g., products), contextual similarity, and social relation between influencers and brands.

9.4.5 Analysis and Discussions

In this section, we first study the effectiveness of the proposed model with different feature sets to understand the impact of features to discover sponsored posts. We then evaluate the performance of SPoD with the sets of posts that have different caption lengths.

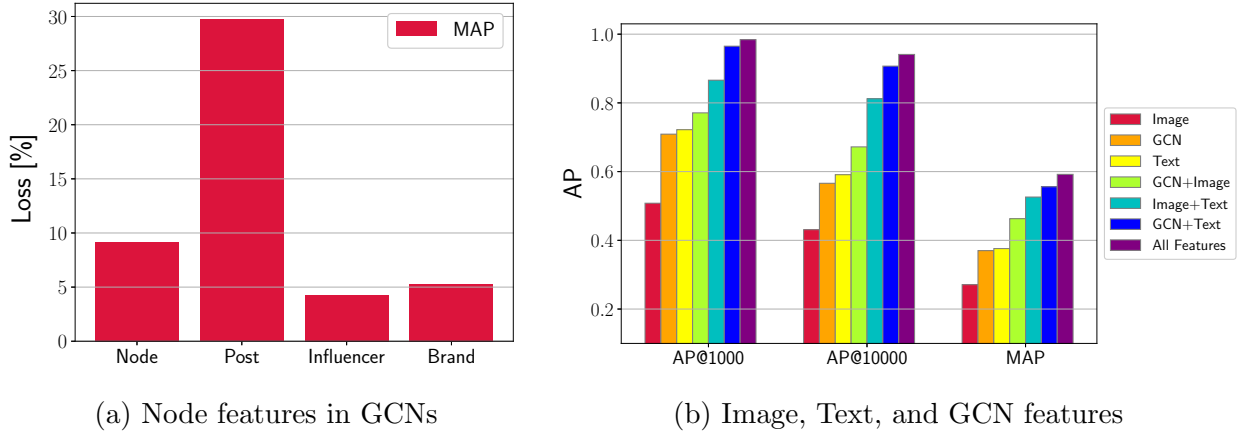


Figure 9.6: The post node features and the text features play an important role in detecting sponsorship of posts. SPoD significantly improves the performance in the short post set.

Feature Importance. To understand the importance of each feature set in the proposed SPoD for detecting sponsored posts, we evaluate the rank performance over different feature sets. Figure 9.6a shows the performance loss of MAP scores of the models trained with the features excluding one particular node feature against the full model as the leave-one-out analysis. We find that the post features have a larger loss value than other node features since sponsored posts have distinct characteristics from non-sponsored posts [354]. Figure 9.6a also shows that the node type features have a large loss value. This suggests that social relations between influencers and brands that are indirectly learned from node types provide valuable information to detect sponsorship of posts.

Figure 9.6b shows the average precision scores of the proposed SPoD over (i) only image features, (ii) only GCN features, (iii) only text features, (iv) GCN and image features, (v) image and text features, (vi) GCN and text features, and (vii) All features. The result reveals that the text features significantly improve the performance while the image features contribute to the slight improvement. Note that SPoD loses 27.86%, 12.55%, and 6.47% performance in MAP when it excludes text, GCN, and image features, respectively. This suggests that the contextualized information from captions is very useful for discovering sponsored posts. Due to the nature of paid advertisements, influencers try to recommend the products, convey detailed information of products, and to make a good impression on the

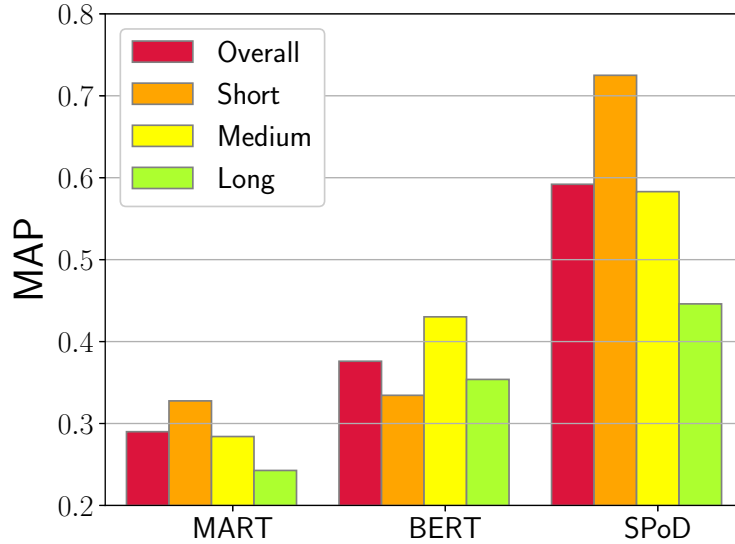


Figure 9.7: The MAP scores of methods with different caption lengths of posts.

brand in the text [342]. This consequently makes the paid advertisements to have distinct contextualized features from non-sponsored posts. The image features, on the other hand, have fewer benefits in discovering sponsored posts compared to the other features. Unlike the text features, which naturally show similar characteristics due to the commonality of language, images can be generated in various ways by different users, thereby making it difficult to rank the posts.

Caption Length of Posts. Since we find that textual information plays an important role in determining whether given social media posts are sponsored, we investigate the performance of the proposed model over different sets of posts with various caption lengths. We split the test dataset into three subsets, including short, medium, and long posts, which have less than 250 (34.5%) and 500 (31.9%), and more than 500 (33.6%) characters in a caption, respectively. Note that we use the same training set for the post sets with different caption lengths thus our model can be used for posts with various caption lengths to detect sponsorship. Figure 9.7 shows the performance of SPoD and two baseline methods over the post sets in different caption lengths. By comparing the performance of MART and BERT, we observe that BERT obtains noticeable improvement in medium and long length posts. This reveals that the contextualized information greatly help detect paid partnership in the posts.

However, because of the same reason, BERT fails to improve the performance on the short posts due to insufficient contextualized information from captions. As shown in Figure 9.7, the proposed SPoD outperforms the baseline methods over all the post sets in various caption lengths. Particularly, SPoD remarkably improves the performance in the short post set. This implies that SPoD effectively leverages the knowledge of both graph structures and the various features to detect the sponsorship of social media posts. For example, a sponsored post with a very short caption can be detected by our model by understanding the social relations and characteristics of adjacent influencers and brands.

9.5 Conclusion

In this paper, we propose a learning-to-rank model for discovering undisclosed sponsorship on social media. Our proposed model, SPoD, employs the aspect-attentive heterogeneous post encoder that incorporates graph, text, and image features to estimate their importance for representing the characteristics of social media posts, and then optimizes the ranking performance by using the regularization based on posting time and mentioned brands. As our proposed model effectively discover undisclosed sponsorship on social media, we strongly believe that SPoD can be deployed as follows: Companies and marketers can utilize SPoD to avoid influencers who frequently hide sponsorship in their advertising social media posts so they can collaborate with highly transparent influencers. SPoD also can be used by social media platforms to alert social media users to be conscious of undisclosed sponsorship, thereby helping them recognize the regulations on advertising posts. As a consequence, SPoD can help resolve the undisclosed sponsorship issue in influencer marketing by making influencers and brands comply with the regulations.

As future work, we intend to conduct the following studies: First, we plan to apply the proposed model to other datasets from various social media platforms as companies have started collaborating with influencers in video-based social media, including YouTube, and TikTok. In addition, we will further collect data from Instagram to include regular posts since the current dataset only contains brand mentioning posts. Next, our proposed SPoD

is designed to be a ranking model to discover a set of untransparent influencers and their advertising posts with high confidence. We aim at improving the proposed model by changing it from a ranking model to a binary classifier so that the model can decide the sponsorship of a given post rather than making a ranked list of posts.

CHAPTER 10

Social Media User Geolocation via Hybrid Attention

Similar to Chapter 9, we show the other example of using the aspect attention to model heterogeneous resources in this chapter. With aspect-attentive representations, we demonstrate great progress in identifying the geolocations of users based on social media posts and their social networks. The in-depth analysis also indicates that the aspect attention can effectively recognize resources that are more important for different situations.

10.1 Introduction

Nowadays, social media has become one of the most powerful tools for myriad real-world applications, such as online marketing [197] and event detection [339]. To facilitate those applications, the geolocations of social media users are usually required. For example, online marketing needs to decide the target audience based on their locations. A real-world event may be only related to the users within a certain geographical region. However, there are only a limited amount of social media posts annotated with posting geolocations because position sensors and services can be unavailable or prevented. In addition, most of the social media users also do not denote their locations in the user profiles due to the data privacy issue. Hence, it is important to identify user geolocations with only their behaviors on social media.

One of the most intuitive approaches for user geolocation is to analyze the natural languages utilized in social media posts. Users can mention specific entities or events related to geolocations and people living in a certain region may reveal noticeable habits or patterns in their languages. For example, Rahimi et al. [271] extract bag-of-words features from user

posts; Wing and Baldrige [341] estimate the word distributions for different regions; Han et al. [141] conduct feature selection to discover location indicative words. However, user language usage sometimes can be too vague and ambiguous to recognize their locations, especially in social media posts with only limited and noisy texts. Several less active users that rarely publish posts may also have insufficient data for geolocation.

In addition to social media posts published by users, social interactions with other users can also be applied to user geolocation. More precisely, a user can be more likely to reach out to the users living in closer areas. For instance, Davis Jr et al. [87] and Jurgens [186] exploit label propagation and rely on the location redundancy through user relationships. Wang et al. [327] derive node embeddings of social networks and location networks as features for user geolocation. Nevertheless, the sparsity of social networks can still lead to unsatisfactory performance. Social connections can be relationships with users living in other locations. Although previous studies utilize text frequency in social networks [272] and train machine learning models with heterogeneous features [97], conventional approaches are significantly affected by the network structures. Moreover, the importance of social networks can be distinct across different users.

In this chapter, the framework, Hybrid-attentive User Geolocation (HUG), is proposed to tackle the above issues. Social media posts of each user are first encoded by a hierarchical language attention network. The social network of users is modeled by a graph attention network so that the relations between users can be leveraged in representation learning. Finally, the hybrid attention mechanism is applied to automatically decide the individual importance scores of user posts and the social network for each user, thereby identifying her geolocation. To improve the prediction performance of tail locations, we also propose a novel location regularizer that leverages the knowledge from other locations. In the end, we conduct extensive experiments to show the effectiveness of HUG with in-depth analysis. We also demonstrate the interpretability of HUG with several concrete examples.

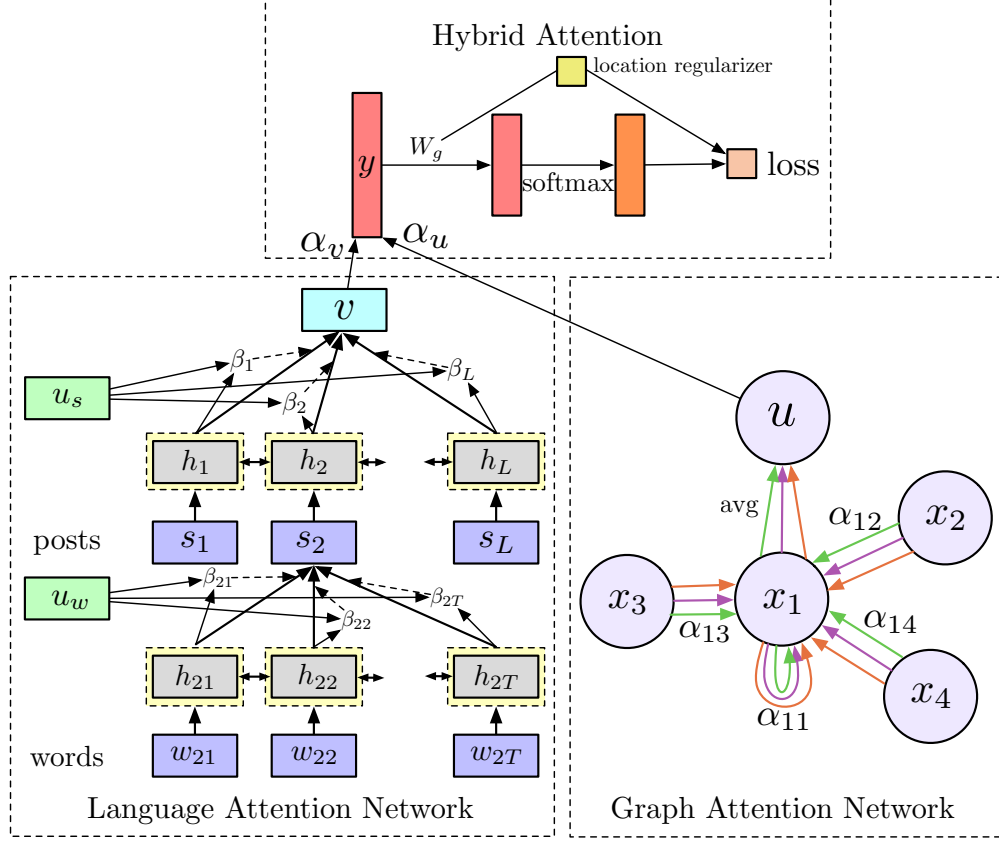


Figure 10.1: The overview of the proposed framework, HUG.

10.2 Hybrid-attentive User Geolocation

In this section, we first formally define the task of social media user geolocation and then introduce our proposed approach, Hybrid-attentive User Geolocation (HUG).

Problem Statement. Suppose we have a set of social media users U and their social network G . For each user $k \in U$, $W^k = \{w_{ij}^k\}$ denotes the social media posts published by the user, where w_{ij}^k represents the j -th word of the i -th post in W^k . The social network $G = (U, A)$ treats each user $k \in U$ as a node and models their relations with an edge set $A \subseteq U \times U$ that indicates the relations between users. Given the social media posts W^k of a user k and the social network G , the goal of this work is to predict the geolocation of the user $L_k \in \mathcal{L}$, where \mathcal{L} is a set of candidate locations.

Framework Overview. Here we propose HUG for social media user geolocation as shown in Figure 10.1. For each user, the language attention network encodes the social media posts of each user into a user language vector v when the graph attention network derives an informative node vector u based on the graphical structure of the given social network. Finally, the hybrid attention mechanism learns the importance scores of two resources, thereby predicting the geolocation. Moreover, the location-based model regularization further improves the model performance by leveraging knowledge across different locations.

10.2.1 Multi-head Graph Attention Network

To learn the structural knowledge from social networks, we employ the graph attention network [322] to derive node representations as user graph vectors.

Node Features. For each user k , the input node features x_k^{in} are the node attributes such as user profiles or bag-of-words features. We use a fully-connected layer to learn the hidden node features as $x_k^0 = \mathcal{F}(W^0 x_k^{in})$, where W^0 is the weight matrix and $\mathcal{F}(\cdot)$ is a nonlinear activation function.

Multi-head Graph Attention Layer. The graph attention network consists of several stacked graph attention layers passing node features x_k^i on different levels. For the i -th layer, the importance score s_{jk}^i of each edge $a_{jk} \in A$ between the users j and k can be estimated by the self-attention mechanism [376, 378] as $s_{jk}^i = \langle \mathcal{Q}^i x_j^{i-1}, \mathcal{Q}^i x_k^{i-1} \rangle$, where \mathcal{Q}^i is the weight matrix applied to every node. The node features in the i -th layer x_k^i can then be obtained as:

$$x_k^i = \sigma \left(\sum_{j \in \mathcal{N}(k)} \alpha_{jk}^i \mathcal{W}^i x_j^{i-1} \right), \alpha_{jk}^i = \frac{\exp(s_{jk}^i)}{\sum_{j' \in \mathcal{N}(k)} \exp(s_{j'k}^i)}$$

where $\mathcal{N}(k)$ indicates the neighbors of the user k in the social network; \mathcal{W}^i is a weight matrix for feature projection; $\sigma(\cdot)$ is a nonlinear activation function. Specifically, we utilize the multi-head attention mechanism to have a greater capability of modeling structural knowledge by concatenating the features generated by different weight matrices \mathcal{Q}_z^i and \mathcal{W}_z^i , where $1 \leq z \leq H$; H is the number of heads. Finally, in the last layer, we average the

multi-head features and delay the employment of the nonlinear activation to derive the user graph vector u_k as:

$$u_k = \sigma \left(\frac{1}{H} \sum_{z=1}^H \sum_{j \in \mathcal{N}(k)} \alpha_{jk,z}^N \mathcal{W}_z^i x_j^{N-1} \right),$$

where N is the number of graph attention layers.

10.2.2 Language Attention Network

We use a hierarchical language attention network [356] to encode the textual features for each user. The language attention model is composed of several parts, including a word embedding layer, a post encoder, and a user encoder.

Word Embedding Layer. We convert each word w_{ij} into a one-hot encoding representation \tilde{w}_{ij} and embed the words to vectors e with an embedding matrix E , where $e_{ij} = E \cdot \tilde{w}_{ij}$.

Post Encoder. For each post of a user, we feed the word embeddings to a bidirectional Recurrent Neural Network (BiRNN) to learn a hidden state of each word with sequential information as:

$$\overleftarrow{h}_{ij} = \text{GRU}(\overleftarrow{h}_{i,j+1}, e_{ij}), \overrightarrow{h}_{ij} = \text{GRU}(\overrightarrow{h}_{i,j-1}, e_{ij}), h_{ij} = [\overleftarrow{h}_{ij}, \overrightarrow{h}_{ij}],$$

where $\text{GRU}(\cdot)$ is the recurrent neural unit of the BiRNN. Here we choose GRU instead of LSTM because of its computational efficiency. To derive the post representation, we introduce an attention layer to obtain a weighted sum of the hidden states from the BiRNN layer. To be specific, we initialize a context vector u_w and calculate the attention scores β_{ij} for the words in the post as:

$$u_{ij} = \tanh(W_w \cdot h_{ij} + b_w), \beta_{ij} = \frac{\exp(u_{ij}^T \cdot u_w)}{\sum_j \exp(u_{ij}^T \cdot u_w)}, s_i = \sum_j \beta_{ij} \cdot h_{ij},$$

where W_w and b_w are the weight matrix and the bias to map each word into a hidden space for estimating importance.

User Encoder. Similarly, we employ a BiRNN model using GRU units to derive the hidden representations h_i for the posts of each user according to their published times as:

$$\overleftarrow{h}_i = \text{GRU}(\overleftarrow{h}_{i+1}, s_i), \overrightarrow{h}_i = \text{GRU}(\overrightarrow{h}_{i-1}, s_i), h_i = [\overleftarrow{h}_i, \overrightarrow{h}_i].$$

The other context vector u_s is then learned to estimate the importance score β_i for each post and aggregate the post representations h_i to form a user language vector v as follows:

$$u_i = \tanh(W_u \cdot h_i + b_u), \beta_i = \frac{\exp(u_i^T \cdot u_s)}{\sum_j \exp(u_j^T \cdot u_s)}, v = \sum_i \beta_i \cdot h_i,$$

where W_u and b_u are the weight matrix and the bias.

10.2.3 Hybrid Attention

To dynamically adjust the importance of two resources for a certain user, we propose the hybrid attention to jointly model texts and social networks. Precisely, a context vector c_h is applied to estimate the importance scores of graph and language user vectors as:

$$\alpha_v = \frac{\exp(o_v \cdot c_h)}{\exp(o_v \cdot c_h) + \exp(o_u \cdot c_h)}, \alpha_u = \frac{\exp(o_u \cdot c_h)}{\exp(o_v \cdot c_h) + \exp(o_u \cdot c_h)},$$

where $o_v = \tanh(W_h \cdot v + b_h)$; $o_u = \tanh(W_h \cdot u + b_h)$; W_h and b_h are the weight matrix and the bias for a nonlinear projection. Therefore, the ultimate feature vector produced by the hybrid attention can be obtained as $y = \alpha_v \cdot v + \alpha_u \cdot u$.

10.2.4 Location-regularized User Geolocation

User Geolocation. Based on the feature vector y , we use a fully-connected hidden layer without a bias to estimate the probability of being the geolocation of the user k for each location i as:

$$P(L_k = i) = \text{Softmax}(W_g y),$$

where W_g is the weight matrix for the hidden layer.

Location-based Regularization. To leverage the knowledge across different locations, we regularize the model weights W_g for inferring location probabilities by the corresponding distances. Specifically, we have the location-based regularization loss Loss_R as:

$$\text{Loss}_R = \sum_{j \in \mathcal{L}} \sum_{k \in \mathcal{L}-j} \frac{|W_g(j) - W_g(k)|^2}{D(j, k)},$$

where $D(j, k)$ denotes the distance between the locations j and k .

Learning and Optimization. Finally, the loss function of *HUG* for optimization can be derived by the cross-entropy loss for classification and the location-based regularization loss as:

$$\text{Loss} = \sum_i \mathbb{1}[L_k = i] \cdot P(L_k = i) + \gamma \cdot \text{Loss}_R,$$

where γ is the weight of regularization loss.

10.3 Experiments

10.3.1 Experimental Setup

Datasets. We employ three public Twitter user geolocation datasets: (1) *GEOTEXT* [105], (2) *TWITTER-US* [280] and (3) *TWITTER-WORLD* [141]. The datasets are pre-partitioned into training, development and test sets. In each dataset, user tweets are concatenated into single documents. The social graphs are extracted with the mention relations between users, where two users are connected if one mentions the other, or they co-mention a third user. The node attributes in the social graphs are the bag-of-words and TFIDF features. The labels are the discretized geographical coordinates of the training users using a k-d tree [280]. Dataset statistics are summarized in Table 10.1.

Baselines. We compare the proposed *HUG* against 6 baselines that are trained based on the text and network features to determine user geolocations, including: (1) MLP + k-d tree

Table 10.1: Dataset statistics.

	GeoText	Twitter-US	Twitter-World
Users	9,475	449,200	1,386,766
Classes	129	256	930
Train	5,685	429,200	1,366,766
Dev	1,895	10,000	10,000
Test	1,895	10,000	10,000

[271], a text-based multilayer perceptron model; (2) GCN-LP [272], a network-based model using one-hot neighbor encoding as the node attributes; (3) MENET [97], a multiview neural network model that utilize multi-entry data to infer users’ locations; (4) MLP-TXT+NET [272], a multilayer perceptron model with the concatenation of text features and adjacent lists as input; (5) GCN [272], a graph convolution network model [200] with the bag-of-words as node attributes; (6) HLPNN [159], a feature fusion model with city and country objectives.

Evaluation. We evaluate the models with three commonly used metrics: (1) **Acc@161**, the accuracy of predicting a user within 161km or 100 miles from the labeled location; (2) **Mean**, the mean error between the predicted and labeled location; (3) **Median**, the median error between the predicted and labeled location.

Implementation Details. We implement the proposed HUG in PyTorch framework for efficient GPU computation. The language attention network has bidirectional GRUs with hidden dimensions in $\{50, 100, 200\}$ and the word embeddings are initialized with the Glove vectors [263] pre-trained on the Twitter corpus. The entity-level aggregation network has two layers with the hidden dimension $D_e \in \{64, 128, 256\}$. The number of heads in multi-head graph attention is searched in $\{1, 2, 4, 8, 16\}$. We apply Adam optimizer for training and the initial learning rate is set as 5.0×10^{-4} . The activation functions are ELU [72].

Table 10.2: Twitter user geolocation prediction performance.

	GEO TEXT			TWITTER-US			TWITTER-WORLD		
	Acc@161 ↑	Mean ↓	Median ↓	Acc@161 ↑	Mean ↓	Median ↓	Acc@161 ↑	Mean ↓	Median ↓
MLP + k-d tree	38%	844	389	54%	554	120	34%	1456	415
GCN-LP	58%	576	56	53%	653	126	45%	2357	279
MENET	62%	532	32	66%	433	45	53%	1044	118
MLP-TXT+NET	58%	554	58	66%	420	56	58%	1030	53
GCN	60%	546	45	62%	485	71	54%	1130	108
HLPNN	-	-	-	71%	362	32	59%	828	60
HUG	64%	516	30	72%	359	31	62%	818	49

Table 10.3: Ablation study on TWITTER-US.

	Acc@161 ↑	Mean ↓	Median ↓
HUG	72%	359	31
w/o graph attention	51%	531	57
w/o language attention	58%	612	63
w/o location regularization	59%	562	51

10.3.2 Experimental Results

Table 10.2 summarizes the model performance of Twitter user geolocation prediction on all datasets. Overall, HUG is able to outperform other baselines across the three datasets on all metrics. We make the following observations. (1) Compared with MLP + k-d tree [271] and GCN-LP [272] that only utilizes a single source of data, e.g. text or network features, HUG outperforms by simultaneously learning the important language features and network structure features. (2) As the feature fusion models, MENET [97], MLP-TXT+NET [272] and HLPNN [159] incorporate the fixed network embeddings as features. In contrast, our proposed HUG can adaptively fine-tune both attention models to favor the geolocation prediction by the hybrid attention mechanism. (3) Compared with GCN [272], our attention-based approach can better understand the hierarchical language features and assign different importance to nodes of the same neighborhood. (4) We conduct the ablation study by removing the graph attention, language attention and location-based regularization one by one at a time. As the results on the TWITTER-US dataset shown in Table 10.3, each module contributes to the performance improvement and the proposed HUG benefits from the combination and the hybrid attention mechanism.

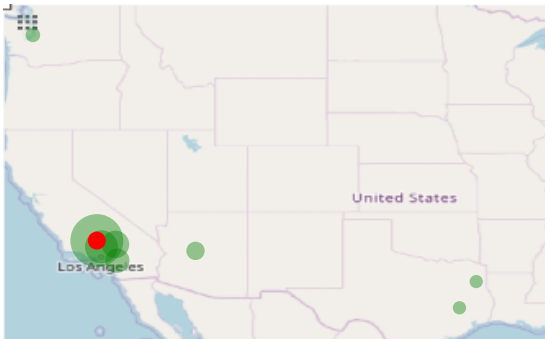
We further investigate the interpretability of the proposed HUG. Figure 10.2 shows the

I just found out I'm doin 2 shows tonite in **Louisville** becuz the 1st show sold out thats lol sorry I just saw your screen name and seen the area code! My bad for bothering u. lol so why gon' be in **louisville** tonight that's gon' be funny man! yeah well r spr break actually start the 13 but we dnt leave til the 17th so I'll omg they got me weak everything they keep sayin and doin is sooo hilarious

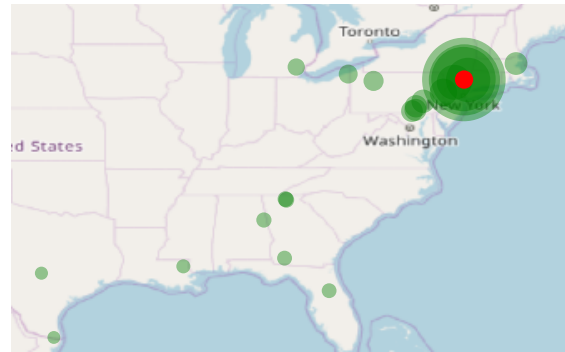
(a) User A

I can't wait to move bck to **LA** thanks i did that one a few hours ago i havent had no sleep but there stil more Why is everyone moving to **culver city** ugh I wanna move to **west hollywood** nxt omg 2mc Live today like its the last don't worry about thee future n don't dwell in the past just **USC** gon cost me \$42,000 smhshyt there prices went down it use to be 55,000

(b) User B



(c) User C



(d) User D

Figure 10.2: Attention weight analysis. (a)-(b) Documents of users A and B. (c)-(d) Geolocations and attention weights of users C and D with their one-hop neighbors.

text and graph examples from the GEOTEXT dataset. In (a) and (b), we show the social media posts of two users (A, B), whose hybrid attention weights for texts are $\alpha_v = 0.643$ and 0.794 , respectively. The blue blocks denote the tweet-level attention weights. The orange denotes the word attention weights and our model can select the words with a strong indication of geolocations like *Louisville*, *LA*, *USC* and *West Hollywood*. Figure 10.2 (c) and (d) demonstrate two users (C, D) with the geolocations and attention weights of their one-hop neighbors. The hybrid attention weights for graph vector are $\alpha_u = 0.844$ and 0.942 for user C and D, respectively. We plot the geolocations of user C and D in red dots. The green dots are the geolocations of the one-hop neighbors and the dot sizes denote the attention weights. Our proposed HUG also works in terms of the graph attention and location-based regularization, by assigning the higher weights to closer neighbors and lower weights to farther neighbors.

10.4 Conclusion

In this chapter, we propose a novel end-to-end framework, Hybrid-attentive User Geolocation (HUG), to jointly model the post texts and user interactions in social media and predict user geolocations. We introduce the hybrid attention mechanism to automatically determine the importance of texts and social networks while social media posts and interactions are modeled by a graph attention network and a language attention network. The experimental study on three benchmark geolocation datasets from Twitter shows that HUG consistently renders superior prediction performance against baseline approaches. We also demonstrate the interpretability of HUG with in-depth analysis of attention weights.

CHAPTER 11

Reformulation Inference Network for Context-Aware Query Suggestion

In this chapter, we demonstrate an example of modeling individual human behaviors in web search for query suggestion based on compositional and explainable behavior modeling as introduced in Chapter 1. We treat network representation learning as a blackbox to derive representations of query terms. Based on those query term representations, we propose the novel idea of homomorphic query embedding so that we can conduct compositional operations to compute representations of query reformulation that jointly reflect semantic and syntactic changes.

11.1 Introduction

Although search engines have already become indispensable in our daily life, the difficulty of deciding the ideal queries is everlasting. The search intents are often sophisticated while queries are usually not only short but also ambiguous [49]. As a consequence, in order to refine the search results, users have to articulate their information needs by reformulating the queries. The burden is on users. To make searching easier, most modern search engines turn to query suggestion that provides recommendations for next queries. Because users may be unfamiliar with the topics or the vocabulary for formulating queries, the concise suggestions can further accelerate search satisfaction. Moreover, query suggestion also benefits various applications such as query auto-completion.

To capture the search intents, it is intuitive to exploit the context information, including

previous queries and click-through information in the same search session. The idea of utilizing the context has been widely studied in both query suggestion [37, 50, 148, 160, 234, 297] and query auto-completion [29, 58, 294]. Many conventional context-aware methods solely rely on query association or similarity between queries. For example, reused terms [29, 160, 294] and query co-occurrence [37, 148, 182, 234, 297] are advantageous to discover promising queries. However, query association and similarity based methods suffer from data sparsity. Even though some works [49, 50] attempt to alleviate the hardship by clustering queries based on click-through data, query clusters can still be too sparse to raise the roof of the limitations. In addition, queries in search sessions are issued successively, but the order of queries is generally ignored by query co-occurrence and similarity. The search sessions are not only diverse but also highly complicated, so methods considering only query association or similarity may fail in learning how users reformulate queries.

Fully understanding query reformulations is the grail of context-aware query suggestion because the context of search sessions are affected by reformulating queries to refine search results. To analyze reformulation behaviors, pioneer researchers categorized query reformulations into predefined strategies [86, 161]. More specifically, these reformulation strategies can be typecast into *syntactic* and *semantic* reformulations. Syntactic reformulations consist of predefined syntactic changes between queries such as adding terms, deleting terms, and acronym expansion [38, 161]. The clear definitions of syntactic reformulation strategies come in handy to design features for machine learning models of query suggestions [174]. On the contrary, semantic reformulations incorporate strategies changing the meanings of queries such as generalization and specialization [6]. In practice, the semantic interoperability of queries is usually achieved by resorting to established ontologies, thereby exteriorizing semantic reformulations for query suggestion [6, 171, 172]. However, most of the reformulation-based methods are restricted and count on predefined reformulation strategies or assistance of ontologies, suffering from out-of-scope strategies and ambiguous queries.

The growth of deep learning, especially recurrent neural networks (RNNs) on sequences, provides the opportunity to generalize the use of reformulations in query suggestion. Fol-

lowed by the success in machine translation [22, 308], sequence-to-sequence (seq2seq) models based on RNNs are adopted for query suggestion [88, 298]. The seq2seq models read the previously issued queries as a sequence, and then freely generate a sequence of terms as the suggested query. The terms of each query can be also treated as a sequence to derive information through another RNN [298]. Furthermore, since diverse search sessions increase the difficulty of learning reformulations, Dehghani et al. [88] decompose the generation of a query into reformulation strategies of appending and copying terms. However, there are a few shortcomings of existing deep learning based methods in query suggestion. First, although reformulations have been guided during generation [88], reformulations in the context still remain an open problem. The lack of understanding of previous reformulations can result in unsatisfactory query suggestions as shown in previous studies [6, 174]. Second, predefined strategies are too restricted to learn sophisticated reformulations that can be hard to decompose or even out of the scope. A better representation of reformulation is needed. Last but not least, although to some degree RNNs consider the transition between queries in the search session, models can still have a hard time learning reformulations because relationships between queries are sparse and implicit. As a consequence, existing RNN-based models [88, 298, 345] rely on additional features for discriminative query suggestion and hence often fail in predicting the intended query.

Here we propose Reformulation Inference Network (RIN) to address the limitations. More precisely, we focus on modeling reformulation behaviors for query suggestion with the reformulation representations capturing both syntactic and semantic relations. The system of homomorphic query embedding based on term embeddings is first introduced to preserve the syntactic property of reformulations. In addition, a heterogeneous network embedding model integrates click-through data in search logs into the foundational term embeddings to capture semantic reformulations. Simultaneously reading both the query and the recent reformulation for each step, RIN encodes the search session into a context vector by an RNN with the attention mechanism [22]. The reformulation inferencer then improves the capability of the context vector by predicting the next reformulation. Based on the context

vector, multi-task learning is employed to train a query discriminator and a query generator with the reformulation inferencer. We then can utilize the model to suggest queries. Here we summarize our contributions in the following.

- To the best of our knowledge, this is the first work to model user behaviors based on queries and reformulations with homomorphic encoding that simultaneously preserves syntactic and semantic properties. The general and flexible representations can benefit the learning of how users reformulate queries along search sessions for query suggestion.
- We propose the framework RIN that deals with the data sparsity by inferring not only the intended query but also the next reformulation for query suggestion. More specifically, the reformulation representations enable the opportunity to leverage the knowledge across syntactically and semantically similar reformulations. When the model becomes more proficient in forecasting the next reformulations, the ability to discriminate and generate suggested queries can be also sharpened as well with multi-task learning.
- Experiments conducted on the publicly available AOL search engine logs demonstrate that RIN significantly outperforms existing methods for either discriminative or generative query suggestion. A study of parameter sensitivity then indicates the robustness of the proposed framework.

11.2 Problem Statement

In this section, we first formally define the objectives. A search session can be formally represented as a sequence of queries $\langle q_1, q_2, \dots, q_L \rangle$ submitted successively by a single user within a time interval. Each query q is composed of a set of terms $T(q)$ and associated with the corresponding click-through information u , which is a set of clicked URLs. Suppose the user intends to submit a query q_{L+1} after the search context $\langle q_1, q_2, \dots, q_L \rangle$. The two goals are listed as follows:

1. **Discriminative Query Suggestion:** Given a set of candidate queries Q_{can} , we would like to give a ranking of candidates $q_{can} \in Q_{can}$ so that q_{L+1} ranks as high as possible.

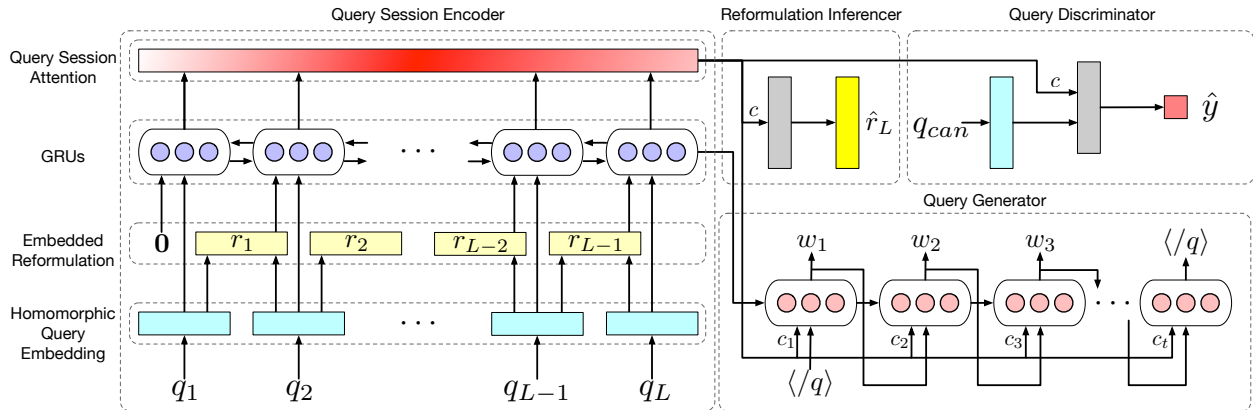


Figure 11.1: The schema of the proposed Reformulation Inference Network (RIN).

Some previous work [298] also call this task next-query prediction.

2. **Generative Query Suggestion:** Different from the discriminative task, the generative task does not rely on candidate sets. With only the search context, generative query suggestion aims to generate a query q'_{L+1} , which is expected to be as similar as possible to the intended query q_{L+1} based on some similarity measures, such as cosine similarity and position independent word error rate [88].

For simplicity, the context is referred to as the search context $\langle q_1, q_2, \dots, q_L \rangle$; the context length is the number of queries in the search context.

11.3 Reformulation Inference Network

In this section, we present the proposed framework, Reformulation Inference Network (RIN), for modeling queries in search sessions.

11.3.1 Framework Overview

Figure 11.1 demonstrates the general schema of RIN. The model mainly consists of four components, including query session encoder, reformulation inferencer, query discriminator, and query generator. Based on reformulation representations derived from homomorphic query embeddings, the query session encoder wraps the search session into a vector using

a recurrent neural network and session-level attention. The reformulation inferencer plays the main role in RIN to infer the next query reformulation with the encoded vector from the query session encoder. Finally, to solve either the discriminative or generative task, the query discriminator and generator can be learned with the reformulation inferencer by multi-task learning.

11.3.2 Distributed Reformulation Representation

We first formally define how to represent reformulations in our model. For syntactic reformulations, some works designed several discrete features to measure reformulations [174]; other works simplified reformulations as a few basic syntactic operations, such as copying a term from the context and appending a new term [88]. For semantic reformulations, the topics of queries can be utilized to observe the changes of concepts behind queries [172]. In this work, we propose to model reformulations from both perspectives.

Inspired by homomorphic encryption that allows computations on ciphertexts [246], we want to design an embedding system that can reflect reformulations with subtraction computations. Theoretically, every reformulation can be syntactically factorized into adding and removing terms. Based on this concept, we come up with the *homomorphic query embedding* to represent each query as follows:

Definition 11.1 (Homomorphic Query Embedding). Suppose that every term t has a representative embedding \mathbf{v}_t . The homomorphic embedding of a query q is defined as:

$$\mathbf{v}_q = \sum_{t \in T(q)} \mathbf{v}_t,$$

where $T(q)$ consists of all terms in the query.

Based on the homomorphic query embeddings of queries, the reformulation \mathbf{r}_i from q_i to q_{i+1} can be represented as the difference between embeddings as follows:

$$\mathbf{v}_{q_{i+1}} - \mathbf{v}_{q_i},$$

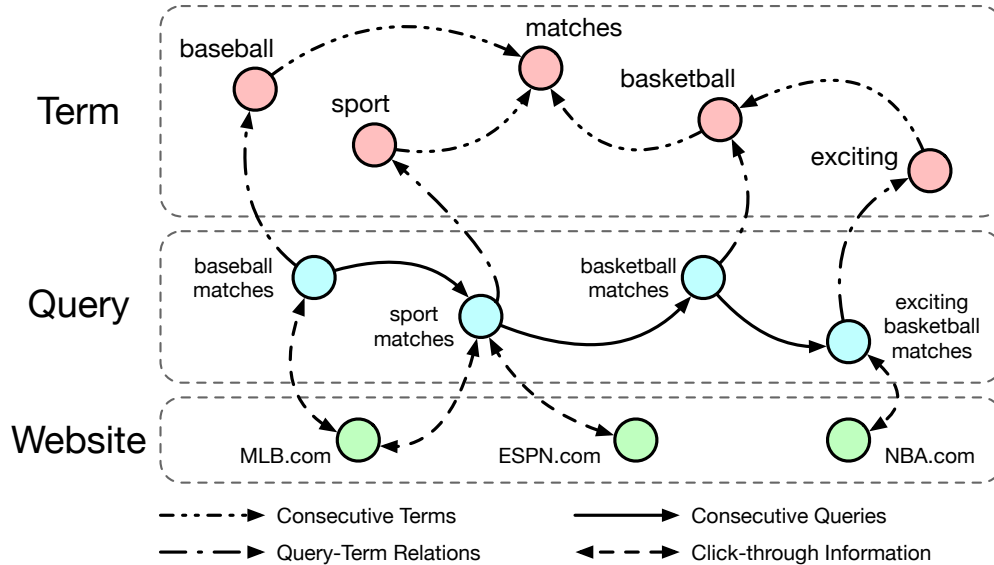


Figure 11.2: An example of the heterogeneous network constructed by a search session of four queries for deriving term embeddings. Note that the queries in the graph are auxiliary nodes connecting the domains of terms and websites.

which can be also considered as a vector from q_i to q_{i+1} in the hidden space of homomorphic query embeddings. There are at least three benefits to apply homomorphic query embeddings to perform reformulations. First, the syntactic relations are homomorphically preserved as adding and subtracting term embeddings. Second, the latent space of embeddings also implicitly captures the semantic information of queries. Last but not least, the linear substructures of embeddings [241, 263] are helpful to understand the semantic relationships between reformulations and offer high interpretability. For example, the reformulation from “Japan travel” to “Tokyo travel” can be shown as $\mathbf{v}_{\text{Tokyo}} - \mathbf{v}_{\text{Japan}}$, which is close to $\mathbf{v}_{\text{Rome}} - \mathbf{v}_{\text{Italy}}$ from “Italy hotel” to “Rome hotel” under the country-capital reformulation substructure.

In this work, we propose to exploit heterogeneous network embedding to learn the features of queries and terms. To learn the semantics of queries, term dependencies [209, 241, 263] and click-through data [172, 175, 218, 234] have been demonstrated to be useful. Here we propose to unify terms, queries, click-through data, and their relationships using a heterogeneous network. Figure 11.2 shows an example of the network constructed by a search session of four queries. Each of terms, queries, and websites has a representative node in the network.

Consecutive terms in each query and consecutive queries in each search session are connected. Each query also links to the first term and has bidirectional edges to the relevant websites clicked in the training logs. Finally, any of network embedding methods can be applied. Here node2vec [135], which is one of the state-of-the-art methods, is utilized to learn term embeddings as the base of homomorphic query embedding.

11.3.3 Query Session Encoder

The encoder of query sessions in RIN is a bidirectional recurrent neural network (Bi-RNN) with attention. The input of the encoder is a sequence $X = [x_1, x_2, \dots, x_L]$, where $x_i = [\mathbf{v}_{q_i}; \mathbf{r}_{i-1}]$ concatenates the homomorphic query embedding \mathbf{v}_{q_i} and the reformulation \mathbf{r}_{i-1} from the last query q_{i-1} for each query q_i in the search session. Note that \mathbf{r}_0 is set as a zero vector because the first query has no reformulation. The Bi-RNN reads the input sequence twice as the forward pass and the backward pass. During the forward pass, the Bi-RNN creates a sequence of forward hidden states $\vec{\mathbf{h}} = [\vec{\mathbf{h}}_1, \vec{\mathbf{h}}_2, \dots, \vec{\mathbf{h}}_L]$, where $\vec{\mathbf{h}}_i = \text{RNN}(\vec{\mathbf{h}}_{i-1}, x_i)$ is generated by a dynamic function such as LSTM [152] and GRU [69]. Here we use GRU instead of LSTM because it requires fewer parameters [183]. The backward pass processes the input sequence in reverse order. The backward hidden states are then generated as $\overleftarrow{\mathbf{h}} = [\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \dots, \overleftarrow{\mathbf{h}}_L]$, where $\overleftarrow{\mathbf{h}}_i = \text{RNN}(\overleftarrow{\mathbf{h}}_{i+1}, x_i)$. Finally, the forward and backward hidden states are concatenated as the encoder hidden representations $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_L]$, where $\mathbf{h}_i = [\vec{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$.

Since each query in the search session can have unequal importance for inferring the next query, the attention mechanism [22] is introduced to extract and aggregate representations that are more important than others. More specifically, the representation \mathbf{h}_i will first be transformed by a fully-connected hidden layer to u_i to measure the importance α_i as follows:

$$\mathbf{u}_i = \tanh(\mathcal{F}_s(\mathbf{h}_i)),$$

$$\alpha_i = \frac{\exp(\mathbf{u}_i^T \mathbf{u}_s)}{\sum_{i'} \exp(\mathbf{u}_{i'}^T \mathbf{u}_s)},$$

where \mathcal{F}_s is the fully-connected hidden layer; \tanh is chosen as the activation function for the convenience of similarity computation; \mathbf{u}_s is a vector to measure the importance by computing $\mathbf{u}_i^T \mathbf{u}_s$. Finally, the normalized importance α_i is obtained through a softmax function. The context vector \mathbf{c} can be represented as the weighted sum of the encoder hidden representations \mathbf{h} as follows:

$$\mathbf{c} = \sum_i \alpha_i \mathbf{h}_i.$$

11.3.4 Reformulation Inferencer

To enhance the capability of inferring the intended query, we assume that a model that accurately predicts the next reformulation can also correctly forecast the next query. More precisely, the intended query q_{L+1} is equivalent to the reformulation \mathbf{r}_L with the last query q_L . Additionally, inferring reformulations is more trainable than directly predicting next queries because similar reformulations may be shared across different queries.

The aim of the reformulation inferencer is to predict the next reformulation $\mathbf{r}_L = \mathbf{v}_{q_{L+1}} - \mathbf{v}_{q_L}$ established by homomorphic query embeddings. Taking the context vector \mathbf{c} , the reformulation inferencer first applies a fully-connected hidden layer for the non-linearity of prediction as follows:

$$\mathbf{u}_r = \text{ReLU}(\mathcal{F}_{h_r}(\mathbf{c})),$$

where $\text{ReLU}(\cdot)$ is the rectified linear unit [247] as the activation function; $\mathcal{F}_{h_r}(\cdot)$ is the fully-connected layer. The predicted reformulation $\hat{\mathbf{r}}_L$ can then be modeled as a linear combination of the result as follows:

$$\hat{\mathbf{r}}_L = \mathbf{W}_r \mathbf{u}_r + \mathbf{b}_r,$$

where \mathbf{W}_r and \mathbf{b}_r are the weights and the biases for the combination. If the predicted reformulation $\hat{\mathbf{r}}_L$ is close to the actual reformulation \mathbf{r}_L , the model should also have considerable potential to infer the next query q_{L+1} .

11.3.5 Query Discrimination and Generation

The reformulation inferencer predicts the next reformulation, but the tasks mentioned in Section 11.2 cannot directly be solved by reformulations represented as homomorphic embeddings. Hence, *query discriminator* and *query generator* are proposed to directly solve discriminative and generative tasks, respectively.

Query Discriminator. Given a candidate query q_{can} and the context vector \mathbf{c} from the query session encoder, the goal of the query discriminator is to assess how likely q_{can} is the intended query. More precisely, we want to predict a probabilistic score \hat{y} to approximate the probability of being the intended query

$$y = P(q_{can} = q_{L+1} \mid \langle q_1, q_2, \dots, q_L \rangle)$$

for each candidate query q_{can} .

The input of the query discriminator concatenates the homomorphic query embedding of q_{can} and the context vector \mathbf{c} as $\mathbf{x}_d = [\mathbf{v}_{q_{can}}; \mathbf{c}]$. The probabilistic score \hat{y} can then be generated by a sigmoid unit with a fully-connected hidden layer as follows:

$$\begin{aligned} \mathbf{u}_d &= \text{ReLU}(\mathcal{F}_{h_d}(\mathbf{x}_d)), \\ \hat{y} &= \sigma(\mathcal{F}_d(\mathbf{u}_d)), \end{aligned}$$

where $\mathcal{F}_{h_d}(\cdot)$ and $\mathcal{F}_d(\cdot)$ are two fully-connected hidden layers; $\text{ReLU}(\cdot)$ is the activation function for the hidden layer; $\sigma(\cdot)$ is the logistic sigmoid function [142]. **Query Generator.** Without any candidate query, the query generator aims to produce a sequence of terms as the generated query q'_{L+1} that estimates the intended query q_{L+1} . Inspired by seq2seq [22, 308] in machine translation, the query generator is designed as a decoder to generate a sequence of terms based on the output of the query session encoder.

The query generator as a decoder also relies on RNN. To generate the t -th term w_t , the

hidden state of RNN can be computed based on the last predicted query w_{t-1} as follows:

$$\mathbf{s}_t = RNN(\mathbf{s}_{t-1}, [w_{t-1}; \mathbf{c}_t]),$$

where \mathbf{c}_t is the context vector of the t -th term; similar to the query session encoder, the dynamic function $RNN(\cdot)$ is GRU in the experiments. Instead of always using the general context vector \mathbf{c} , \mathbf{c}_t estimates a more appropriate context vector because each query in the context may play different role in generating w_t . More precisely, the last hidden state of the decoder \mathbf{s}_{t-1} is taken into account to compute the importance and construct the dynamic context vector \mathbf{c}_t as follows:

$$\begin{aligned} \mathbf{u}_{t,i} &= \tanh(\mathcal{F}_g([\mathbf{s}_{t-1}; \mathbf{h}_i])), \\ \alpha_{t,i} &= \frac{\exp(\mathbf{u}_{t,i}^T \mathbf{u}_g)}{\sum_{i'} \exp(\mathbf{u}_{t,i'}^T \mathbf{u}_g)}, \\ \mathbf{c}_t &= \sum_i \alpha_{t,i} \mathbf{h}_i, \end{aligned}$$

where $\mathcal{F}_g(\cdot)$ is a fully-connected hidden layer. Based on \mathbf{c}_t , we further define the conditional probability for generating w_t as follows:

$$P(w_t | w_1, \dots, w_{t-1}, \mathbf{c}) = f(\mathbf{s}_t),$$

where $f(\cdot)$ is a projection layer that estimates the conditional distribution over the vocabulary.

11.3.6 Learning and Optimization

The multi-task learning is applied to simultaneously train different components in RIN. Each of the reformulation inferencer, the query discriminator, and the query generator has a corresponding loss function jointly optimized with other components.

For the reformulation inferencer, the loss function loss_R optimizes the distance between

the actual reformulation \mathbf{r}_L and the predicted reformulation $\hat{\mathbf{r}}_L$ as follows:

$$\text{loss}_R = \frac{1}{2} \|\mathbf{r}_L - \hat{\mathbf{r}}_L\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm [320].

For the discriminative tasks, the query discriminator solves a binary classification problem. Hence the loss function loss_D focuses on reducing the binary cross-entropy [151] between the predicted probabilistic score \hat{y} and the gold standard y as follows:

$$\text{loss}_D = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})),$$

where y is a binary indicator demonstrating if the candidate query q_{can} is the intended query q_{L+1} .

To solve the generative tasks, the query generator produces a probability distribution over the vocabulary, so the loss function loss_G can be modeled by the cross-entropy between the generated sequences of words and the actual intended query as follows:

$$\text{loss}_G = - \sum_{w_t} \log P(w_t | S_t),$$

where w_t is the t -th term in the intended query, and S_t is the preceding terms of w_t .

Finally, the objective of multi-task learning combines the loss functions of different components as follows:

$$\text{loss} = \text{loss}_R + \text{loss}_{\text{task}},$$

where $\text{loss}_{\text{task}}$ can be either loss_D or loss_G based on the task to be solved. Moreover, the loss can also consider both of the loss functions, such as $\text{loss}_D + \text{loss}_G$, to simultaneously solve both problems.

11.4 Experiments

In this section, we conduct extensive experiments and in-depth analysis to verify the performance and robustness of RIN for both discriminative and generative query suggestions.

11.4.1 Datasets and Experimental Settings

Datasets. We adopt the largest publicly available AOL search logs for experiments as previous studies [29, 172, 174, 294, 298, 345, 360, 361]. The 3-month dataset consists of queries submitted to the AOL search engine from 1 March, 2006 to 31 May, 2006. We first remove all non-alphanumeric characters and rare queries with less than ten occurrences in the whole log. The query logs are then segmented into sessions with a 30-minute threshold as the session boundary. Single-query sessions with only a single query are discarded because there must be at least one preceding query as the context for context-aware approaches. Every query with at least a preceding query as the context information is treated as the ground truth of the intended queries. The pre-processing process is consistent with previous studies [172, 174, 294]. To partition search sessions into training and testing sets, the first 2-month data are utilized for training while the remaining sessions are the testing data. Among the training data, 10% of sessions are randomly sampled as the validation set for parameter tuning. Finally, there are 1,357,500 training queries within 852,350 sessions and 647,559 testing queries within 403,772 sessions. Furthermore, to evaluate the performance using different context lengths, the testing set is partitioned into three subsets, including *Short Context* (1 query), *Medium Context* (2 to 3 queries), and *Long Context* (4 or more queries). As a result, Table 11.1 shows the statistics of queries with different context lengths in the training and testing datasets.

Evaluation. Similar to previous studies [88, 298, 345], the evaluation of discriminative query suggestion relies on candidate queries. The top-20 queries based on the frequency of the co-occurrences with the last query in the context are selected as candidate queries as *Most Popular Suggestion* (MPS) [88, 298] for re-ranking. The re-ranked results can then

Table 11.1: The statistics of queries with different context lengths in the training and testing datasets.

Dataset	Context Length		
	Short (1 query)	Medium (2-3 queries)	Long (4+ queries)
Training	852,350	386,970	118,180
Testing	403,772	184,843	58,944

be evaluated by mean reciprocal rank (MRR). For generative query suggestion, we follow the previous study [88] to employ the position independent word error rate (PER) [318] to estimate the word overlap based similarity between the generated queries and the actual intended queries.

Implementation Details. The model is implemented by TensorFlow [1]. The Adam optimizer [196] is adopted to optimize the parameters and perform back-propagation algorithm based on gradients. The initial learning rate and the dropout parameter are set as 10^{-3} and 0.5. After the parameter tuning with the validation set, the number of hidden neurons for GRUs is set as 128, and the number of dimensions for homomorphic embeddings is 256.

Baseline Methods. To evaluate the performance of RIN, we compare with the following baseline methods in different categories.

- *Most Popular Suggestion* (MPS) [88, 298] is a maximum likelihood method, which relies on “wisdom of the crowd” and ranks queries by the co-occurrence to the last query in the context. Note that the candidate queries in the experiments are generated by MPS.
- *Query-based Variable Markov Model* (QVMM) [148] learns the probability of query transitions over sessions with the variable memory Markov model implemented by a suffix tree.
- *Hybrid Suggestion* (Hybrid) [29] considers both the context information and the popularity by ranking candidate queries based on a linear combination between the popularity (i.e., MPS in our experiments) and the similarity to recent queries.
- In addition to the popularity of queries, *Personalized Completion* (PC) [294] also incorporates the personal query logs as long-term history to provide a personalized ranking model

based on LambdaMART [46], which is one of the state-of-the-art ranking models.

- *Reformulation-based Completion* (RC) [174] is the only non-deep learning baseline method exploiting query reformulation. 43 reformulation-based features are proposed to capture user reformulation behaviors over search sessions with LambdaMART.
- *Hierarchical Recurrent Encoder-Decoder* (HRED) [298] and *Seq2Seq with Copiers* (ACG) [88] are deep learning based query suggestion methods. HRED constructs a hierarchical encoder-decoder structure to model the sequential and hierarchical dependencies across terms and queries. ACG extends the seq2seq structure to read the terms in the search session and then learn whether to copy the used term or to add a new term.

Note that two deep learning baseline methods HRED and ACG solve the discriminative tasks by requiring an external feature set to train a LambdaMART model. Although our approach is also a deep learning based method, the query discriminator of RIN can address the task without any support of external features.

11.4.2 Experimental Results

Discriminative Query Suggestion. We first evaluate the performance for discriminative query suggestion. Table 11.2 shows the MRR performance of different methods over various context lengths. Note that a high MRR score indicates the actual intended queries are ranked more favorably.

The hybrid suggestion method (Hybrid) slightly boosts the suggestion performance of the popularity-based baseline method (MPS) by considering the similarity between candidate queries and the local context. On the contrary, the performance of the personalized completion method (PC) drops after considering the historical logs of users as long-term historical data. It is because an enormous amount of users in the search logs have only little or even no historical data so that the sparsity causes a severe over-fitting phenomenon. QVMM based on a variable-memory Markov model is also marginally better than MPS but not as well as Hybrid. The reason can be explained by the complicated search sessions that are too sparse to be modeled by query dependencies as shown by the lower performance

Table 11.2: The MRR performance of different methods in the testing sets with different context lengths for the task of discriminative query suggestion. All improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.

Dataset	MPS [88, 298]	Hybrid [29]	PC [294]	QVMM [148]	RC [174]	HRED [298]	ACG [88]	RIN
Overall Context	0.5471	0.5823	0.5150	0.5671	0.6202	0.6207	0.6559	0.8254
Short Context (1 query)	0.5680	0.5822	0.5343	0.5862	0.5960	0.6100	0.6471	0.8361
Medium Context (2 to 3 queries)	0.5167	0.5841	0.4865	0.5338	0.6689	0.6489	0.6542	0.8190
Long Context (4 or more queries)	0.4826	0.5768	0.4575	0.5026	0.6704	0.6122	0.6669	0.7611

with longer context. The reformulation-based completion method (RC) is the best non-deep learning method in the experiments. The promising results of the reformulation features used in RC show again that reformulations are helpful for modeling search sessions as previous studies [88, 174]. Two deep learning methods, HRED and ACG, outperform all of the other baseline methods because RNNs carefully capture the sequential information of terms and queries in search sessions. ACG is further better than HRED since two reformulation strategies are considered in the model. As the proposed approach, RIN surpasses all of the baseline methods. More precisely, RIN achieves 50.87% and 25.83% improvements in the dataset of overall context over MPS and ACG, respectively. It is also worth noting that the improvements are consistent across all datasets with different context lengths. All of these improvements are significant at the 95% confidence level in a paired t-test.

To discuss the performance with different context lengths, we investigate the improvements of all methods over the naïve baseline method MPS because the performance on different datasets is not comparable. Figure 11.3 shows the improvements of three best baseline methods and RIN over MPS with different context lengths. It is reasonable to see the improvements are generally greater with longer contexts because sequential information and reformulations are more sufficient in longer sessions. An interesting observation is that the improvements of RIN over MPS are similar in the datasets of medium and long contexts. It demonstrates that RIN needs fewer queries to understand the search intents of users. The other observation is that RC outperforms HRED and achieves similar performance to that of ACG while there are multiple queries in the context. The results indicate reformulations are hard to be instinctively captured by RNNs. As a consequence, as shown in Table 11.2, HRED

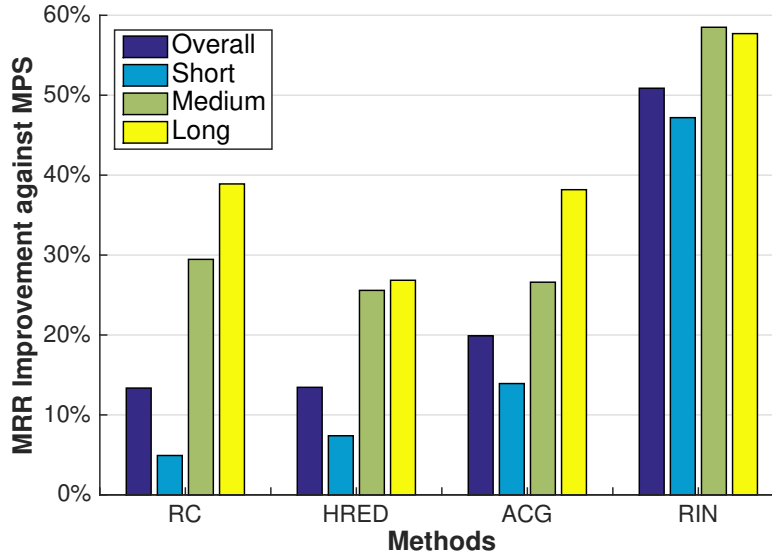


Figure 11.3: The MRR improvement of four methods over the MPS baseline method with different context lengths.

performs worse with long contexts, compared to other baselines considering reformulations as predefined features and strategies. Furthermore, RIN effectively learns reformulations and achieves the best performance.

Generative Query Suggestion.. Here we evaluate the performance for generative query suggestion. Note that only HRED and ACG are compared as the baseline methods because other methods are not generative models. PER as the evaluation metric treats each query as a bag of words and measures the difference between word sets. A low PER indicates high coverage of the predicted query to the intended query. Table 11.3 shows the PER performance of RIN and both baseline methods. Among two baseline methods, ACG outperforms HRED because of the consideration of reformulations. For HRED, it is interesting that PER dramatically improves when it comes to the long context dataset. It may be because longer contexts have more queries to capture the user behaviors. The proposed approach RIN outperforms both two baseline methods. More specifically, RIN obtains 22.02% and 4.72% improvements over HRED and ACG. For different context lengths, the improvements are greater with longer contexts. For instance, the improvement over ACG with short contexts is only 2.39%, but RIN improves ACG by 8.64% and 9.13% for the datasets of medium and long contexts, respectively. In addition, these results are also consistent with Figure 11.3 in

Table 11.3: The PER performance of three methods. The improvements of our approach over baseline methods are statistically significant at the 95% confidence level in a paired t-test.

Dataset	HRED [298]	ACG [88]	RIN
Overall	0.8069	0.6925	0.6612
Short (1 query)	0.8179	0.7015	0.6851
Medium (2 to 3 queries)	0.8338	0.6733	0.6197
Long (4 or more queries)	0.6753	0.6673	0.6115

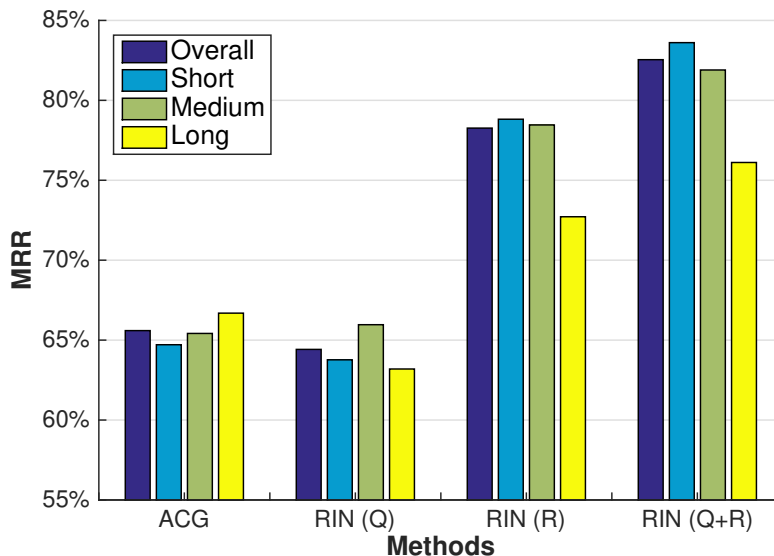


Figure 11.4: The MRR performance of RIN with either or both of homomorphic query and reformulation embeddings. Q denotes the homomorphic query embeddings while R indicates the embeddings of reformulations.

the experiments of discriminative query suggestion.

11.4.3 Analysis and Discussions

In this section, we first analyze the effectiveness of the proposed model and then study the sensitivity of parameters.

Effectiveness of Homomorphic Embedding. We first investigate the effectiveness of the proposed homomorphic embedding of queries reformulations for query suggestion. As a leave-one-out analysis, we train RIN with only query embedding \mathbf{v}_{q_i} or reformulation embeddings \mathbf{r}_{i-1} to verify their individual capability. Figure 11.4 shows the MRR performance

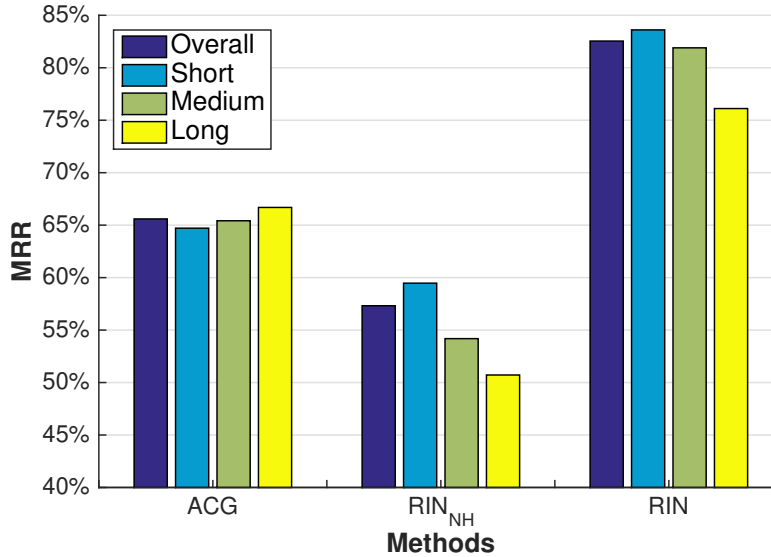


Figure 11.5: The MRR performance of RIN with homomorphic and non-homomorphic embeddings. Note that RIN_{NH} replaces the original homomorphic query embeddings with non-homomorphic query embeddings from node2vec.

of RIN with and without homomorphic query and reformulation embeddings, where Q and R denote the use of query embeddings and reformulation embeddings. It is obvious that although the query embeddings are excluded, RIN (R) only lowers MRR by 5.18% from the comprehensive RIN (Q+R). However, when RIN (Q) only considers the query embeddings, the MRR drops by 21.96%. Moreover, RIN (Q) performs even worse than the best baseline method ACG. Even though reformulations are much beneficial for query suggestion, they are so hard to learn without directly being represented. The other observation is that the gain from the reformulation embeddings, i.e., the improvement of RIN (Q+R) over RIN (Q), is more substantial for shorter contexts. For example, MRR improves by 23.73% for the short context dataset while the improvement in the long context dataset is only 16.97%. When the reformulations are clearly given, it becomes more convenient for models to understand user behaviors and predict the intended queries.

Homomorphic vs. Non-homomorphic Embeddings. In addition to the effectiveness of homomorphic embeddings, we also want to show its advantage against non-homomorphic embeddings. Here we replace the query embeddings v_q in RIN with the query node embeddings from node2vec (see Section 11.3.2), which are non-homomorphic. Note that the

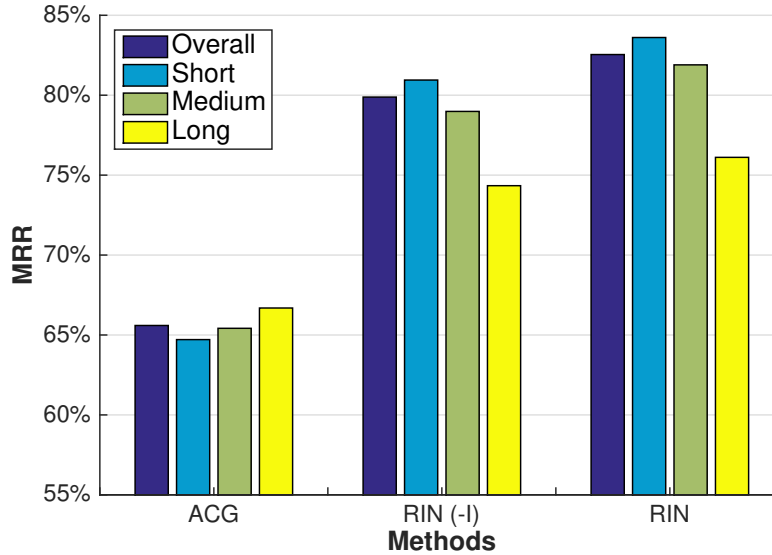


Figure 11.6: The MRR performance of RIN with and without the reformulation inferencer. Note that RIN (-I) removes the reformulation inferencer.

reformulation embeddings will be calculated based on the updated query embeddings after the replacement, although they are not theoretically homomorphic. Figure 11.5 shows the MRR performance of RIN with homomorphic and non-homomorphic embeddings. Note that RIN_{NH} represents RIN using non-homomorphic embeddings. When the query embeddings of RIN become non-homomorphic, the MRR drops substantially. It is because non-homomorphic query embeddings fail in representing how users reformulate queries, especially for syntactic reformulations. Although the previous work [243] demonstrates some examples of syntactic relationships, non-homomorphic embeddings still cannot adequately preserve the syntactic reformulations.

Effectiveness of Reformulation Inferencer. Here we examine how the reformulation inferencer works in RIN. Similar to the previous analysis, we remove the component of reformulation inferencer and observe the performance. Figure 11.6 depicts the MRR performance of RIN with and without the reformulation inferencer, where RIN (-I) is trained without the inferencer component. After removing the component, the MRR performance decreases by 3.22%. It shows that the reformulation inferencer is actually helpful to predict the intended queries. Moreover, although the reformulation inferencer is ignored, RIN (-I) still outperforms the best baseline method ACG by 17.89% with reformulation embeddings

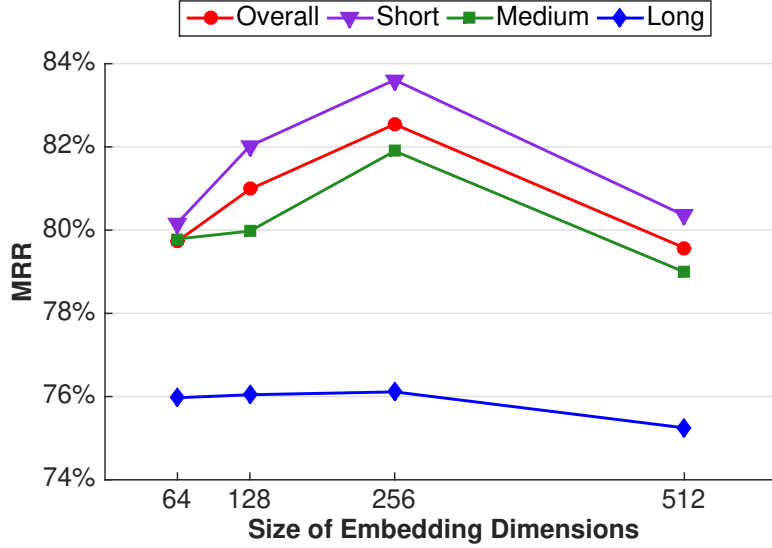


Figure 11.7: The MRR performance of RIN over different numbers of embedding dimensions.

as the inputs of query session encoder. Hence, the results also demonstrate the effectiveness of the proposed embedding method.

Size of Embedding Dimensions. Here we try to study how the size of reformulation embeddings (i.e., the number of embedding dimensions) affects the performance. Figure 11.7 shows the MRR performance of RIN over different numbers of embedding dimensions. When the dimension number increases from a small size, MRR improves and peaks at the dimension number of 256. With a larger size of dimensions, RIN becomes overfitted because of the sparsity of search sessions. There is another interesting finding about long sessions. MRRs of datasets with shorter contexts are more sensitive to the embedding size than MRRs of the long context dataset. When the context consists of multiple queries, the embeddings of all queries and previous reformulations are fed into the model. Therefore, even though the embedding size is small, the combination of multiple queries in the RNN can still lead to a high degree of freedom for the prediction capability.

11.5 Conclusion

In this chapter, we propose a novel approach for context-aware query suggestion by learning to represent query reformulations and model how users reformulate queries over search sessions. Inspired by homomorphic encryption, the *homomorphic query embedding* is introduced to reflect reformulations with computations based on semantic term embeddings. Our model, RIN, is then formulated as a neural network architecture to read the previous reformulations and infer the next reformulation in the embedding space, thereby addressing either discriminative or generative query suggestion. The extensive experiments demonstrate that our proposed model significantly outperforms seven baseline methods for both the discriminative and generative tasks of query suggestion. The improvements are consistent across different datasets of various context lengths. Moreover, the results of analysis also show the effectiveness and robustness of the proposed model. The reasons and insights can be concluded as follows: (1) reformulations are important to model search sessions, so homomorphic reformulations embeddings that precisely capture both syntactic and semantic reformulations can essentially benefit query suggestion; (2) the effectiveness of the reformulation inferencer in RIN implies that a model can be more capable of predicting the intended query if the next reformulation can be also anticipated; (3) longer contexts would not be necessarily required for understanding the search intents if the model can capture how users reformulate queries.

CHAPTER 12

Learning to Disentangle Interleaved Conversations

In this chapter, we discuss how to model interpersonal behaviors with compositional and explainable behavior modeling presented in Chapter 1. We take the task of conversation disentanglement as an example to model human interactions in multi-party conversations. Specifically, we treat the well-designed convolutional neural network as a blockbox for deriving the representation of each message. The construction of the similarity graph is considered compositional operations to leverage the redundancy of messages with similar semantics in the latent space. Therefore, messages in the same similarity graph can be treated as a conversational thread with the same topic.

12.1 Introduction

With the growth of ubiquitous internet and mobile devices, people now commonly communicate in the virtual world. Among the various methods of communication, text-based conversational media, such as internet relay chat (IRC) [340] and Facebook Messenger⁷, has been and remains one of the most popular choices. In addition, many enterprises have started to use conversational chat platforms such as Slack⁸ to enhance team collaboration. However, multiple conversations may occur simultaneously when conversations involve three or more participants. Aoki et al. [13] found an average of 1.79 conversations among eight participants at a time. Moreover, some platforms like chatrooms in Twitch may have more concurrent conversations [140]. Interleaved conversations can lead to difficulties in both grasping dis-

⁷Facebook Messenger: <https://www.messenger.com/>

⁸Slack: <https://slack.com/>

Thread	Message
⋮	⋮
T31	<i>Malcolm: If running as root, I need to set up a global config rather than ~/.fetchmailrc ?</i>
T38	<i>Elma: i'm sure i missed something but fonts rendering in my gimp works isn't at its best</i>
T39	<i>Sena: is there anyway to see what the CPU temperature is?</i>
T38	<i>Elma: is it because of gimp or i missed some tuning or such?</i>
T31	<i>Rache: Specify a non-default name run control file.</i>
T41	<i>Denny: so how does one enforce a permission set and ownership set on a folder and all its children?</i>
T31	<i>Malcolm: in the man page it doesn't mention any global fetchmailrc file... that is what was confusing me...</i>
T42	<i>Shenna: hi, are sata drives accessed as sda or hda?</i>
T41	<i>Elma: -R for recursive...</i>
T42	<i>Elma: sda</i>
⋮	⋮

Figure 12.1: A segment of real-world conversations involving six users and five (annotated) threads from the IRC dataset.

cussions and identifying messages related to a search result. For example, Figure 12.1 shows a segment of conversations from the real-world IRC dataset as an example. Five interleaved threads are involved in only ten messages. Messages in the same thread may not have identical keywords. Moreover, a user (i.e., *Elma*) can participate in multiple threads. Hence, a robust mechanism to disentangle interleaved conversations can improve a user’s satisfaction with a chat system.

One solution for conversation disentanglement is to model the task as a topic detection and tracking (TDT) [9] task by deciding whether each incoming message starts a new topic or belongs to an existing conversation. Messages in the same conversation may have higher similarity scores [230, 290] or similar context messages [333]. However, similarity thresholds for determining new topics vary depending on context. Embedding of earlier messages, resulting in duplication of parts of messages, can alter the similarity score. More specifically, the similarity scores obtained in previous work cannot well represent conversation-level relationships between messages.

Several studies have examined the use of statistical [101] and linguistic features [106, 107, 108, 230] for predicting user annotations of paired message similarity. These studies

employed bag-of-words representations which do not capture term similarity and cannot distinguish word importance and relationships between words in a message. Thus, better representations of messages and their relationships are needed.

Recent studies have demonstrated the effectiveness of deep learning methods in representation learning [32], aiming to infer low-dimensional distributed representations for sparse data such as text [151]. These representations can be derived not only for words [241] but also sentences and documents [209]. In particular, convolutional neural networks (CNNs) have been shown to efficiently and effectively preserve important semantic and syntactic information from embedded text sequences [34]. It has been demonstrated that CNNs produce state-of-the-art results in many NLP tasks such as text classification [194, 206, 368] and sentiment analysis [268, 311]. Existing approaches, however, do not take advantage of deep learning techniques to model relationships between messages for disentangling conversations. [233] defined many statistical features for use with a random forest for in-thread classification and used a recurrent neural network (RNN) only to model adjacent messages with an external dataset as a feature.

In this work, we aim to leverage deep learning for conversation disentanglement. Our proposed approach consists of two stages: (1) message pair similarity estimation and (2) conversation identification. In the first stage, we propose the Siamese hierarchical convolutional neural network (SHCNN) to estimate conversation-level similarity between pairs of closely posted messages. SHCNN is framed as a Siamese architecture [245] concatenating the outputs of two hierarchical convolutional neural networks and additional features. Compared to other conventional CNN-based Siamese networks [289, 357], SHCNN models not only local information in adjacent words but also more global semantic information in a message. In the second stage, the algorithm of conversation identification by similarity ranking (CISIR) ranks messages within a time window paired with each message and constructs a message graph involving high-rank connections with strong confidence. Although only high-confidence relations are represented in the constructed graph, the redundancy of pairwise relationships can capture the connectivity of messages within a conversation.

In summary, the main contributions of this work are threefold: **(1) Deep similarity estimation for conversation disentanglement:** To the best of our knowledge, this is the first study applying deep learning to estimate similarities between messages for disentangling conversations. SHCNN simultaneously captures and compares local and global characteristics of two messages to estimate their similarity. Message representations are also optimized towards the task of conversation disentanglement. **(2) Efficient and effective method:** The selection of message pairs posted closely in time and the proposed CISIR algorithm significantly reduces the computational time from $O(|M|^2)$ to $O(k|M|)$, where $|M|$ is the number of messages, and k is the maximum number of messages posted within a fixed-length time window. When many messages are posted over a long period, the computational time of our approach could be near-linear. **(3) Empirical improvements over previous work:** Extensive experiments have been conducted on four publicly available datasets, including three synthetic conversation datasets and one real conversation dataset from Reddit⁹ and IRC conversations. Our approach outperforms all comparative baselines for both similarity estimation and conversation disentanglement.

12.2 Conversation Disentanglement

In this section, we formally define the objective of this work and notations used. A two-stage approach is then proposed to address the problem.

12.2.1 Problem Statement

Given a set of speakers \mathcal{S} , a message m is defined as a tuple $m = (\mathbf{w}, s, t)$, where $\mathbf{w} = \langle w_1, w_2, \dots, w_n \rangle$ is a word sequence posted by the speaker $s \in \mathcal{S}$ at time t in seconds. Each message m is associated with a conversation $z(m)$. Messages in different conversations can be posted concurrently, i.e., conversations can be interleaved.

Following the settings of previous work [106, 107, 108, 230], a set of pairwise annotations

⁹Reddit: <https://www.reddit.com/>

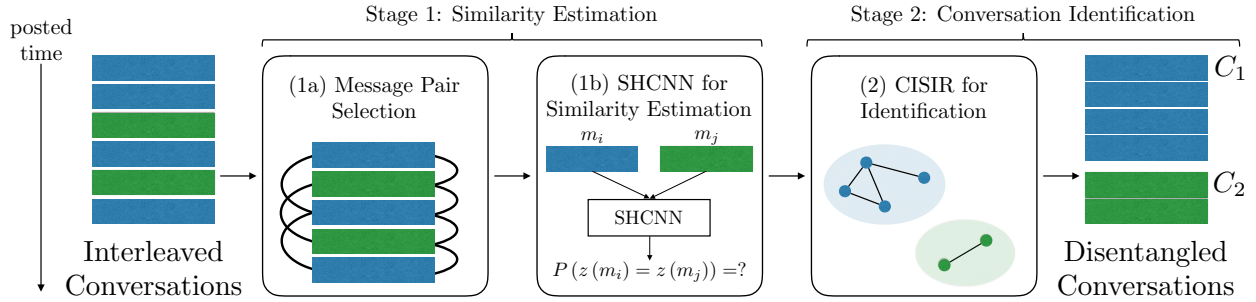


Figure 12.2: Illustration of our proposed two stage method. In the first stage, (1a) message pairs are selected for (1b) estimating pairwise similarity with a Siamese hierarchical CNN (SHCNN). In the second stage, (2) the algorithm of conversation identification by similarity ranking (CISIR) constructs a graph with strong relationships among messages and finds conversations as connected components.

$\mathbf{A} = \{(m_i, m_j, y)\}$, where $y \in \{0, 1\}$, is given for training the model. More specifically, a Boolean value y indicates whether two messages m_i and m_j are in the same conversation, i.e., $z(m_i)$ and $z(m_j)$ are identical.

Given a set of messages \mathbf{M} and the pairwise annotations \mathbf{A} as training data, the goal is to learn a model that can identify whether messages are posted in the same conversation $z(m)$. Note that the number of conversations $|\mathbf{Z} = \{z(m) \mid \forall m \in \mathbf{M}\}|$ is always unknown to the system.

12.2.2 Framework Overview

Figure 12.2 illustrates our two-stage framework. The first stage aims to estimate pairwise similarity among messages. Message pair selection is applied to focus on the similarity between messages that are posted closely in time and thus more likely to be in the same conversation. The Siamese hierarchical CNN (SHCNN) is proposed for learning message representations and estimating pairwise similarity scores. The overlapping hierarchical structure of SHCNN models a message at multiple semantic levels and obtains representations that are more comprehensive.

In the second stage, our conversation identification by similarity ranking (CISIR) algorithm exploits the redundancy and connectivity of pairwise relationships to identify conver-

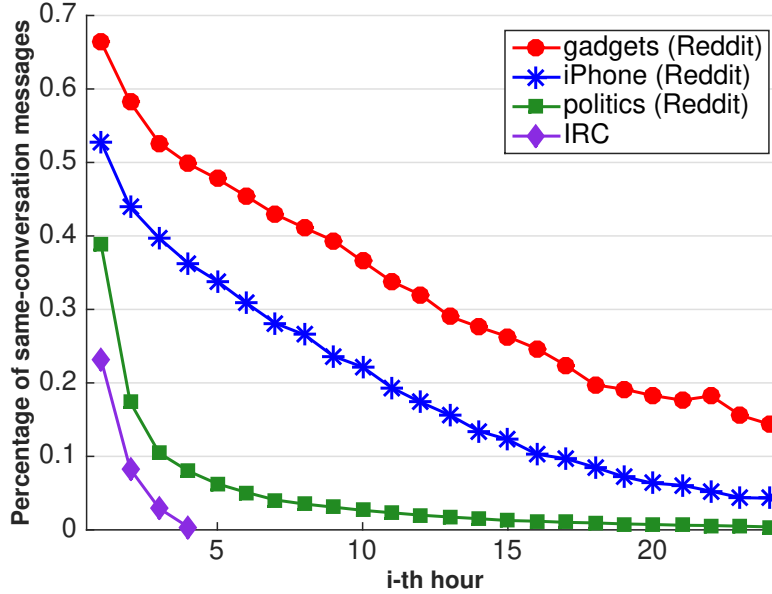


Figure 12.3: The percentage of messages in the same conversation as a given message with elapsed time between messages no greater than i hours for four experimental datasets.

sations as connected components in a message graph.

12.2.3 Message Pair Selection

Most of the previous work on conversation disentanglement focused on pairwise relationships between messages [230]. Especially for single-pass clustering approaches, all pairs of messages need to be enumerated during similarity computation [333]. However, if messages have been collected for a long time, the number of message pairs could be too mammoth to be processed in an acceptable amount of time. More precisely, it leads to at least $O(n^2)$ computational time, where n is the number of messages. As shown in Figure 12.3, the percentage of messages in the same conversation as a given message becomes significantly lower with a longer elapsed time between consecutive messages. In light of this observation, an assumption is made as follows:

Assumption 12.1. The elapsed time between two consecutive messages posted in the same conversation is not greater than T hours, where T is a small number.

More specifically, in our dataset every message m_i is posted within T hours earlier or later

than any other message m_j in the same conversation, i.e., $\frac{|t_i - t_j|}{3600} < T$ for all pairs (m_i, m_j) , where t is in seconds. For example, in the IRC dataset the average elapsed time between consecutive messages in a conversation is only 7 minutes. If a conversation is ongoing, there may not be an extended silence before a new message; conversely, an extended silence could be treated as the start of a new conversation. With this assumption, the number of pairs can be reduced to $O(kn)$, where k is the maximum number of messages posted in a T -hour time window. By default T is set to 1 hour in our experiments.

In addition, it is worth mentioning that it may be possible to include conversational structure, such as replied-to relations, into the model. For example, after using CISIR to identify conversational threads, structure inference may be performed using methods such as described in [18] or [332] and the structure used to refine the threads. In this study, we focus on only conversation disentanglement.

12.2.4 Similarity Estimation with the Siamese Hierarchical CNN (SHCNN)

Given a set of message pairs, we propose the Siamese hierarchical CNN (SHCNN) to estimate the similarity between a pair of messages.

12.2.4.1 Hierarchical CNN for Message Representation

The effectiveness of CNNs for representing text has already been addressed in previous studies. However, single-layer CNNs [194, 289] may not represent high-level semantics while low-level information could be diluted with multiple-layer CNNs [357]. The hierarchical CNN (HCNN) is designed to simultaneously capture low- and high-level message meanings as shown in Figure 12.4.

A message m_i is first represented by a $d \times |\mathbf{w}|$ message matrix $\mathbf{W} \in \mathbf{R}^{d \times |\mathbf{w}|}$, where d is the dimension of a word embedding, and $|\mathbf{w}|$ is the number of words in a message. For low-level information, we exploit single-layer CNNs [194, 289] with a set of $d \times k_L$ kernels, where L denotes “Low”, to extract n -gram semantics of k_L contiguous words. In our experiments, 64

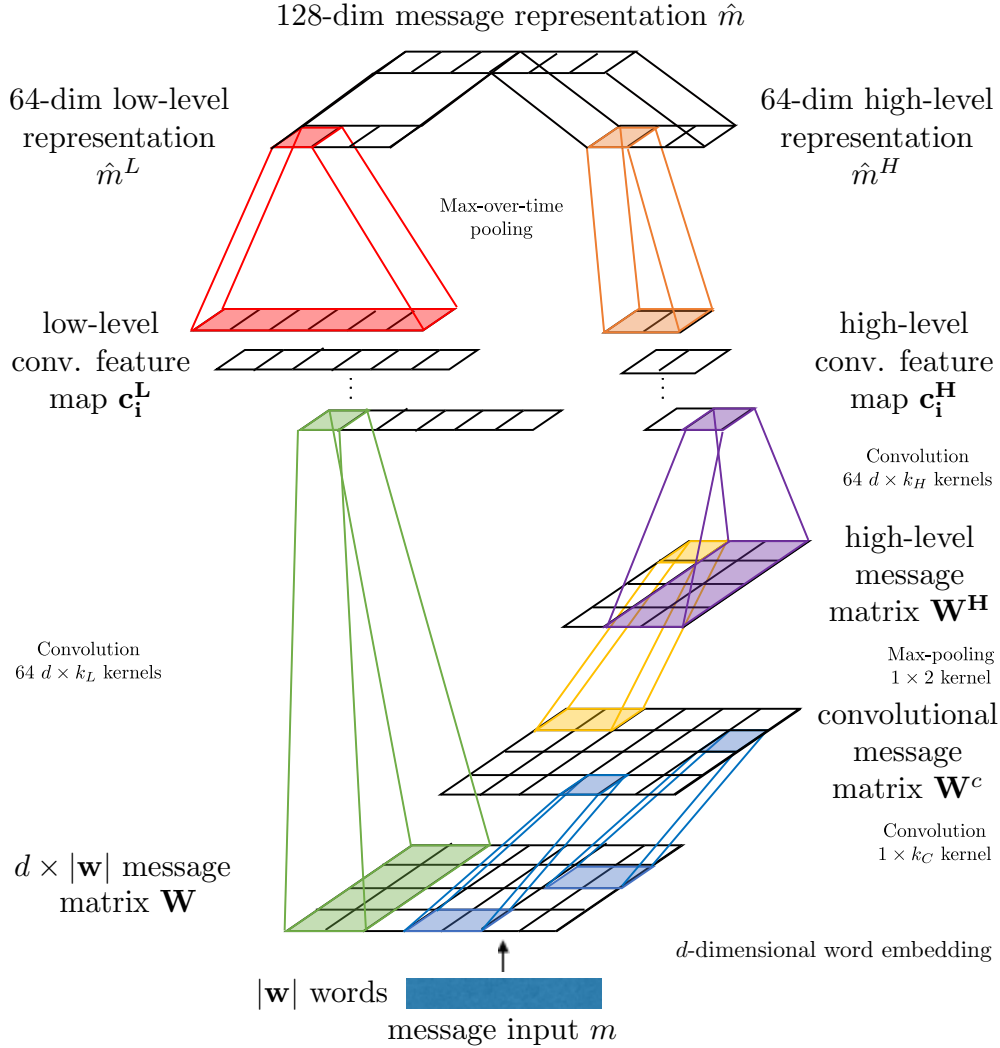


Figure 12.4: Illustration of hierarchical CNN (HCNN) for message representation. The labels with a larger font size indicate the corresponding tensors, and the labels with a smaller font size explain the operations between tensors.

$d \times k_L$ kernels, where $k_L = 5$, are applied to obtain 64 low-level features \hat{m}^L . Note that the kernel row dimension is identical to the word embedding dimension to jointly consider the full embedding vector. As a consequence, convolution with each kernel produces a vector \mathbf{c}_i^L , which is then aggregated by max-over-time pooling [75, 194].

To acquire high-level semantics across a message, HCNN uses another multiple-layer CNN for feature extraction. A $1 \times k_C$ kernel is applied to \mathbf{W} , thereby generating a convolutional message matrix \mathbf{W}^C . Features covering broader contents are computed by applying a 1×2

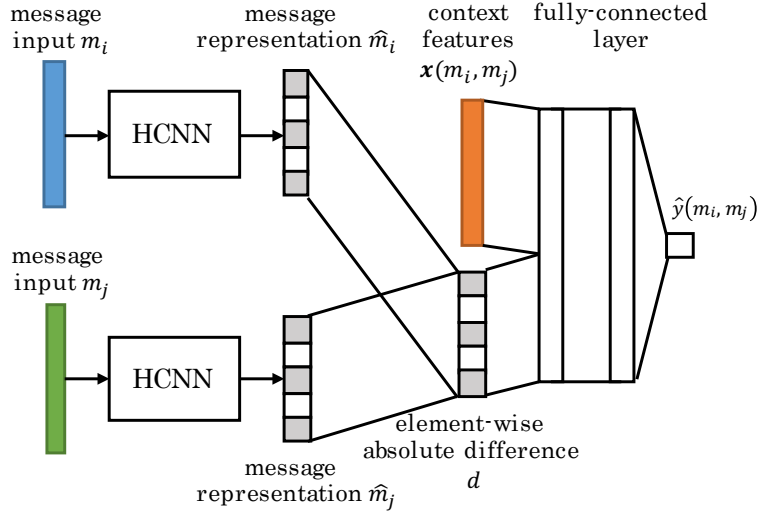


Figure 12.5: The siamese hierarchical CNN (SHCNN) for similarity estimation. Note that the model structure of an HCNN is shown in Figure 12.4.

kernel to a max-pooling layer with a stride of 2, producing a high-level message matrix \mathbf{W}^H . The row sizes of the two kernels are set to 1 to capture relations within each embedding dimension, and convolution is performed on \mathbf{W}^H with $64 d \times k_H$ kernels to capture relations across embedding dimensions. The generated convolutional feature maps \mathbf{c}_i^H are subject to max-over-time pooling, resulting in 64 features \hat{m}^H . Finally, a message representation \hat{m} is constructed by concatenating \hat{m}^L and \hat{m}^H , i.e., creating a 128-dimensional feature vector, for characterizing both low- and high-level semantics of a message m . In our experiments, both k_C and k_H are set to 5 while computing high-level representations.

12.2.4.2 Siamese Hierarchical CNN (SHCNN)

A Siamese structure with two identical sub-networks is useful to exploit the affinity between representations of two instances in the same hidden space [289, 334, 357]. For similarity estimation, we propose the Siamese hierarchical CNN (SHCNN) using a Siamese structure that blends the outputs from two HCNNs as well as some context features.

Figure 12.5 shows the structure of the SHCNN for estimating the similarity between two messages m_i and m_j where the message representations \hat{m}_i and \hat{m}_j are generated by

two sub-networks HCNNs (See Figure 12.4). There are many ways to deal with two sub-networks, such as using a similarity matrix [289] or an attention matrix [357]. However, both methods lead to an enormous number of parameters for long messages. We propose to independently compute the element-wise absolute differences [245] between a pair of message representations \hat{m}_i and \hat{m}_j , each from a sub-network. More formally, the absolute difference d is a vector where the k -th element is computed as $|\hat{m}_i(k) - \hat{m}_j(k)|$. This approach provides not only fewer parameters but also the flexibility to observe interactions among different dimensions in representations. Our experiments also show it outperforms the other two approaches in similarity estimation (See Section 12.3).

In addition to message contents, contexts such as temporal and user information were also usually considered in previous studies about conversation disentanglement [107, 108, 333]. Here we focus on the performance of message content representations and only incorporate four context features: speaker identity, absolute time difference and the number of duplicated words with and without weighting by inverse document frequency [70]. SHCNN concatenates the context features $\mathbf{x}(m_i, m_j)$ with the absolute difference d as the input of a fully-connected layer of the same size.

The final output of SHCNN $\hat{y}(m_i, m_j)$ is normalized by a logistic sigmoid function [142], representing the probability $P(z(m_i) = z(m_j))$.

12.2.4.3 Activation Functions

All convolutional layers and the fully-connected layer require activation functions, and the choice affects the performance [224]. Popular functions include rectified linear units (ReLU) [210], hyperbolic tangent units (tanh) and exponential linear units (ELUs) [73]. In this study, we conducted informal comparison experiments and ELU was finally chosen for all functions because it performed the best.

12.2.4.4 Optimization and Implementation Details

Given a set of annotated message pairs $\mathbf{A} = \{(m_i, m_j, y)\}$, where y is a Boolean value indicating whether two messages are in the same conversation, SHCNN is optimized with binomial cross entropy [131]. More formally, the objective function is as follows:

$$\sum_{(m_i, m_j, y) \in \mathbf{A}} y \cdot \log(\hat{y} + \epsilon) + (1 - y) \cdot \log(1 - \hat{y} + \epsilon) + \lambda \|\boldsymbol{\theta}\|^2$$

where \hat{y} simplifies $\hat{y}(m_i, m_j)$, and ϵ is a small number, i.e., 10^{-9} in our experiments, preventing underflow errors. The term λ serves as the weight for L2-regularization for the set of parameters $\boldsymbol{\theta}$.

In our experiments, SHCNN is implemented by TensorFlow [1] and trained by the Adam optimizer [195] with an initial learning rate of 10^{-3} . The dropout technique [300] is utilized in the fully-connected layer with a dropout probability of 0.1. Word embeddings are initialized using the publicly available fastText 300-dimensional pretrained embeddings from Facebook [36]. The batch size is set to 512, and the maximum number of training epochs is 1,000. The final model is determined by evaluating the mean average precision (MAP) on a validation dataset every 100 iterations.

12.2.5 Conversation Identification by Similarity Ranking (CISIR)

In the second stage of conversation disentanglement, i.e., part (2) in Figure 12.2, we aim to separate conversations based on the identified message pairs and their estimated similarity.

12.2.5.1 Graph-based Methods and Conversation Connectivity

It is intuitive to apply graph-based methods if pairwise relationships of messages are exploited [106]. Furthermore, methods based on single-pass clustering [333] can be also be treated as graph-based methods. However, graph-based methods have a risky drawback: A single false positive connection between two messages can be propagated to several messages from

Algorithm 12.1: The algorithm of conversation disentanglement by similarity ranking (CISIR).

```

1 CISIR ( $\mathbf{M}, \mathbf{D}, r, h$ );
   Input : Message set  $\mathbf{M}$ , the set of selected message pairs  $\mathbf{D}$ , the threshold of
           similarity ranks  $r$  and the threshold of similarity scores  $h$ .
   Output: A set of conversations  $\mathbf{C}$ 
2 Let  $G = (\mathbf{M}, \emptyset)$  be an undirected message graph
3 for  $m \in M$  do
4    $\mathbf{D}_m = \{(m_i, m_j, \hat{y}) \mid m_i = m \vee m_j = m\}$ 
5   Rank entries in  $\mathbf{D}_m$  by  $\hat{y}$  in a descending order
6   for  $k = 1$  to  $\min(r, |\mathbf{D}_m|)$  do
7     Let  $(m_i, m_j, \hat{y})$  be the  $k$ -th entry in ranked  $\mathbf{D}_m$ 
8     if  $\hat{y} < h$  then
9       break
10    Add an edge  $(m_i, m_j)$  into  $G$ 
11  $\mathbf{C} = \text{ConnectedComponents}(G)$ 
12 return  $\mathbf{C}$ 

```

different conversations. As shown in Figure 12.3, a certain percentage of message pairs are in different conversations, which can lead to numerous false positive connections.

False alarms may be reduced by raising the threshold that determines whether two messages are connected [333]. However, a high threshold can make disentangled conversations fragmented and the best threshold for each pair could vary.

12.2.5.2 The CISIR Algorithm

Instead of setting a high threshold, we propose the algorithm of Conversation Identification by SImilarity Ranking (CISIR). CISIR focuses on the top messages ranked by similarity scores. Based on Assumption 12.1, for each message, there exists at least one or more other messages in the same conversation posted closely in time. With this redundancy, a few pairs with stronger confidence, i.e., the top-ranked pairs, can be enough to extend a correct connectivity to earlier or later messages, while the low-ranked pairs can be ignored to reduce the risk of error propagation.

Given a set of selected message pairs with estimated similarity scores $\mathbf{D} = \{(m_i, m_j, \hat{y})\}$,

Algorithm 12.1 shows the procedure of CISIR with two parameters r and h , where r is a high threshold of similarity ranks and h is a lower threshold of similarity scores. Note that CISIR filters out pairs with low scores because a message can have more than r same-conversation pairs posted in its T -hour time window. For each message, CISIR ranks all of its associated pairs by the estimated similarity and only retrieves the top- r pairs whose similarity scores are greater than h . These retrieved high-confidence pairs are treated as the edges in a message graph G . Finally, CISIR divides G into connected components, and the messages in each connected component are treated as a conversation. In the experiments, we use grid search to set r and h as 5 and 0.5, respectively.

12.2.5.3 Improvement of Time Complexity

The efficiency of Algorithm 12.1 can be further improved. The top- r qualified pairs for each message can be pre-processed by a scan of \mathbf{D} with $|M|$ min-heaps which always contain at most $r + 1$ elements. When r is a small constant number, it only takes $O(|D|) = O(k \cdot |M|)$ for pre-processing, where k is the maximum number of messages posted in a T -hour time window. With pre-processed top pairs, CISIR can do graph construction and find connected components in $O(k|M|)$, which compares favorably to conventional methods in $O(|M|^2)$.

12.3 Experiments

In this section, we conduct extensive experiments on four publicly available datasets to evaluate SHCNN and CISIR in two stages.

12.3.1 Datasets and Experimental Settings

12.3.1.1 Datasets

Three datasets from Reddit and one dataset of IRC are used as the experimental datasets.

- **Reddit Datasets**¹⁰ The Reddit dataset is comprised of all posts and corresponding comments in all sub-reddits (i.e., forums in Reddit.com) from June 2016 to May 2017. Comments under a post can be treated as messages in one conversational thread. Here we manually merge all comments in a sub-reddit to construct a synthetic dataset of interleaved conversations. Note that although it is called a “synthetic dataset,” all messages are written by real users. Three sub-reddits with different popularity levels as shown in Table 12.1 are selected to build three datasets: gadgets, iPhone and politics.
- **IRC Dataset.** An annotated IRC dataset used in [106] is also included in our experiments. The IRC dataset consists of about 6 hours of messages in interleaved conversations. Even though the IRC dataset is significantly smaller and shorter than the Reddit datasets, it consists of natural, interleaved conversations with ground truth annotations, including thread id.

12.3.1.2 Experimental Settings

Humans may not participate in a large number of simultaneous conversations. e.g., an average of 1.79 for eight people [13], but there could be hundreds of concurrent posts in a subreddit. Hence, we adjusted the datasets to be more similar to real conversations. Specifically we removed some conversations so that every dataset has at most ten conversations at any point in time. Short messages with less than five words are also removed because even for humans they are frequently ambiguous. Too short conversations with less than ten messages are also discarded as outliers [277]. Training and validation data are randomly chosen from only 10% of the selected message pairs, respectively, because in real situations obtaining labels could be very costly. The remaining 80% of pairs are regarded as testing data. As a result, Table 12.1 shows the statistics of the four datasets after pre-processing.

¹⁰The organized Reddit dataset is publicly available in <https://files.pushshift.io/reddit/>.

Dataset	Reddit			IRC
	gadgets	iPhone	politics	
Conversations	287	617	3,671	39
Messages	8,518	12,433	105,663	497
Speakers	5,185	5,231	25,289	71
Train/Valid Pairs	3,445	5,556	244,492	5,995
Test Pairs	27,565	44,450	1,955,943	47,966

Table 12.1: Statistics of four datasets after pre-processing.

Table 12.2: Performance of pairwise similarity estimation in four datasets. Our approach is denoted as SHCNN. The performance with only low-level or high-level representations are denoted as SHCNN (L) and SHCNN (H). All improvements of SHCNN against the best baseline are significant at the 1% level of significance in a paired t -test.

Dataset	Reddit Datasets									IRC Dataset		
	gadgets			iPhone			politics			P@1	MRR	MAP
Metric	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP			
TimeDiff	0.6916	0.8237	0.8170	0.6085	0.7651	0.7495	0.4412	0.6362	0.5644	0.3262	0.5180	0.4384
Speaker	0.5643	0.7046	0.7425	0.5364	0.6595	0.6590	0.4021	0.4620	0.3914	0.4356	0.6263	0.6891
Text-Sim	0.7913	0.8746	0.8440	0.7347	0.8318	0.7872	0.5245	0.6672	0.5326	0.3712	0.5269	0.3108
Elsner	0.7758	0.8651	0.8321	0.6809	0.7935	0.7471	0.4643	0.6132	0.4884	0.1094	0.1886	0.2063
DeepQA	0.8011	0.8755	0.8511	0.7156	0.8112	0.7766	0.5593	0.6759	0.5685	0.7811	0.8182	0.8050
ABCNN	0.8374	0.8511	0.8502	0.8112	0.8520	0.8118	0.7419	0.6221	0.6644	0.7008	0.4142	0.5858
SHCNN	0.8834	0.9281	0.9005	0.8375	0.8944	0.8497	0.7696	0.8392	0.6967	0.9785	0.9838	0.9819
SHCNN (L)	0.8470	0.9080	0.8702	0.8066	0.8792	0.8275	0.7225	0.8070	0.6438	0.9807	0.9834	0.9750
SHCNN (H)	0.8490	0.9105	0.8704	0.8158	0.8851	0.8313	0.7228	0.8110	0.6283	0.9635	0.9728	0.8632

12.3.2 Pairwise Similarity Estimation

Message pair similarity estimation is treated as a ranking task and evaluated with three ranking evaluation metrics: precision at 1 (P@1), mean average precision (MAP) and mean reciprocal rank (MRR) [70]. We compare the performance with six baseline methods, including the difference of posted time (*TimeDiff*), sameness of speakers (*Speaker*), cosine similarity of text (*Text-Sim*), the approach proposed by Elsner and Charniak [106] (Elsner), *DeepQA* [289] and *ABCNN* [357]. Note that DeepQA and ABCNN are neural network-based models for question-answering. The approach of Mehri and Carenini [233] was not compared in our experiments because the RNN requires additional message sequences; moreover, its performance was only mildly better than Elsner, which performed poorly on IRC in Table 12.2.

Table 12.2 shows the performance of similarity estimation. Among all methods, neural network approaches [289, 357] perform better than other methods in most cases, indicating

$z(m_i)$	Message
T16	<i>“Arлие: Wow, maybe we just missed it when we were driving around”</i>
T18	<i>“Arлие: i’ve been very close to that situation myself”</i>

Figure 12.6: An example message pair in two different conversations from IRC shows how SHCNN discriminates between messages on different topics. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 0.67% of being in the same conversation for this pair while DeepQA with single-layer CNNs predicts 69.81%.

that message content representation has considerable impact on estimating pairwise similarity. SHCNN outperforms most of the baselines even if only low-level (L) or high-level (H) representations are exploited. When SHCNN captures both low- and high-level semantics, it significantly outperforms all baselines across the four datasets. For example, ABCNN can outperform SHCNN using only either low- or high-level representations in the politics dataset; however, SHCNN turns the tables after using both representations. An interesting observation is that ABCNN is the best baseline in every dataset except for IRC; this may be because the IRC data is too small to train complicated attention structures. On the contrary, our SHCNN can precisely capture semantics even with few parameters and limited data.

To shed deeper insights of how SHCNN surpasses other methods, we exhibit the prediction results of the IRC data and demonstrate the capability of SHCNN to simultaneously preserve local and more global information. Figure 12.6 presents an example to show how SHCNN is better than other methods in capturing more high-level topical information. Even though the main sentences of two messages are clearly on different topics, the baseline method DeepQA [289] still predicts a high similarity. This could be attributed to the context of author mention [333] and a bias on the local information, i.e., the exact same term “Arлие”, in the Siamese network used in DeepQA. On the contrary, SHCNN can capture more global information that differentiates the topics and correctly predicts a very low score. Figure 12.7 illustrates another example of how SHCNN outperforms other methods in preserving the similarity of local information. Both of the messages in the example have some segments related to software engineering. A baseline method ABCNN [357] with multiple-layer CNNs, however, still predicts a low score. This might be because both sentences are long so that the

$z(m_i)$	Message
T16	“Very well, I seem to be trying to show Arlie how its done and am coding a webserver. ”
T16	“Arlie: Good enough doesnt cut it! Is the 'faster' method a big change in design? Could I implement later without wanting to kill myself?”

Figure 12.7: An example message pair in a conversation from IRC shows how SHCNN captures similarity in local information. The leftmost column is the conversation IDs of the corresponding messages. SHCNN predicts 70.41% for this pair while ABCNN with multiple-layer CNNs predicts 36.50%.

local information is diluted after processing by multiple CNN layers. Differently, SHCNN is able to seize local information, correctly predicting a high score.

12.3.3 Conversation Identification

For conversation identification, three clustering metrics are adopted for evaluation: normalized mutual information (NMI), adjusted rand index (ARI) and F_1 score (F1). Six methods are implemented as the baselines for conversation disentanglement, including *Doc2Vec* [209], blocks of 10 messages (*Block-10*), messages of respective speakers (*Speaker*) [108], context-based message expansion (*CBME*) [333] and a graph-theoretical model with chat- and content-specific features [106] (*GTM*). The embedding-based clustering method, i.e., *Doc2Vec*, applies affinity propagation [120] to cluster messages embedded using *Doc2Vec* without being given the number of clusters, with the idea that messages in the same conversation would form a cluster. Note that message pairs in the training and validation data are not utilized in prediction for a fair comparison to all methods.

Table 12.3 shows the performance of conversation disentanglement. Note that “Oracle” represents the optimal performance for CISIR when all message pairs in identical conversations in \mathbf{D} are correctly retrieved. Because pairs in \mathbf{D} may not have enough coverage to connect all messages in a conversation, the optimal performance could be lower than 1.0. CISIR performs better than all baseline methods for all datasets, and achieves excellent performance in IRC, due in part to the high-performing similarity estimates from the first stage. Among the baseline methods, GTM performs relatively well on all datasets except for IRC.

Table 12.3: Performance of conversation disentanglement in four datasets. Our approach is denoted as CISIR. “Oracle” indicates the optimal performance if CISIR correctly retrieves all message pairs in identical conversations. All improvements of CISIR against the best baseline are significant at the 1% level of significance in a paired t -test.

Dataset	Reddit Datasets									IRC Dataset		
	gadgets			iPhone			politics					
Metric	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI	F1	NMI	ARI	F1
Doc2Vec	0.1757	0.0008	0.0589	0.2318	0.0002	0.0718	0.2672	0.0001	0.0506	0.2046	0.0048	0.1711
Block-10	0.7745	0.1840	0.3411	0.8203	0.2349	0.4251	0.8338	0.1724	0.3451	0.4821	0.0819	0.2087
Speaker	0.7647	0.0440	0.2094	0.7861	0.1001	0.3339	0.7480	0.0637	0.2207	0.7394	0.4572	0.6310
CBME	0.6913	0.0212	0.1465	0.7280	0.0339	0.1966	0.7883	0.0165	0.1382	0.2818	0.0324	0.1970
GTM	0.7942	0.1787	0.2986	0.8198	0.0536	0.2566	0.8496	0.3076	0.4292	0.0226	0.0001	0.2064
CISIR	0.8254	0.4287	0.4939	0.8552	0.4236	0.5187	0.8825	0.3561	0.4950	0.9330	0.9543	0.8798
Oracle	0.8608	0.4852	0.5560	0.9003	0.5448	0.6358	0.9651	0.8286	0.8863	0.9838	0.9850	0.9819

This is because messages are more frequently posted in the IRC dataset, thereby increasing the number of incorrect pairs in the constructed graph. Examining the graph constructed by GTM, there are only two connected components, indicating that many conversations were incorrectly combined; in contrast, CISIR may be exempt from error propagation because it only relies on top-ranked pairs. Doc2Vec is trained to predict words in a document in an unsupervised manner. Its lowest performance in the experiments may point out a need for supervised learning in the specific task of conversation disentanglement to tackle the variation in semantic patterns. Time and author contextual cues do help conversation disentanglement as seen in the results of Block-10 and Speaker. Both of these contexts are integrated into our model.

12.4 Conclusion

In this chapter, we propose a novel framework for disentangling conversations, including similarity estimation for message pairs and conversation identification. In contrast to previous work, we assume that we do not need to select all message pairs in the first stage, thereby reducing computational time without sacrificing performance too much. To estimate conversation-level similarity, a Siamese Hierarchical Convolutional Neural Network, SHCNN, is proposed to minimize the estimation error as well as preserve both the low- and high-level semantics of messages. In the second stage, we developed the Conversation Identi-

fication by Similarity Ranking, CISIR, algorithm, which exploits the assumption made in the first stage and identifies individual, entangled conversations with high-ranked message pairs. Extensive experiments conducted on four publicly available datasets show that SHCNN and CISIR outperform several existing approaches in both similarity estimation and conversation identification.

CHAPTER 13

Enhancing Air Quality Prediction with Online Community Behaviors in Social Media

In this chapter, we focus on modeling community behaviors jointly achieved by multiple people with compositional and explainable behavior modeling proposed in Chapter 1. As an example, we study joint online behaviors in social media with a task of air quality prediction. As the blackbox of representation learning, we use a convolutional neural network to encode each tweet into a tweet representation. An attention function with a learnable context vector can conduct compositional operations to fuse the representations of tweets posted by users in a certain location within the time period.

13.1 Introduction

In recent centuries, industrialization has considerably changed human society by providing a stimulus to economic growth and improved life quality. However, the advancement is accompanied by the increase in air pollutant emissions and risks to public health. As a consequence, predicting real-time air quality information (AQI), such as the concentration of $PM_{2.5}$, has attracted more and more attention. Air quality prediction may help the government and society to better protect their citizens from potentially harmful effects of poor air quality.

To forecast AQI, one of the most conventional approaches is to exploit historical air quality and treat the task as a time series prediction problem [125, 377]. However, the air quality information can be too sophisticated to be predicted by only past AQI without any

additional knowledge. For example, other environmental factors like humidity and temperature can affect the air quality when real-world events like wildfires may also play a role. To learn the additional information, most of the relevant studies collect data from additional sensors like images [180] and ground sensors [377]. Nevertheless, these sensors are expensive in not only installation but also maintenance. As a result, exploiting sensors for air quality prediction may be too costly for most of the cities.

To learn additional knowledge without physical sensors, one of the most effective approaches is to leverage the wisdom of the crowd on the internet. For example, 81% of the adults in the USA spend on average two hours on social media and collectively publish 170 million tweets¹¹ every day on their feelings and observations [346]. In other words, social media users can be considered as “social sensors” to perceive environmental changes and real-world events. Although social sensing has been applied to detect or predict several real-world events, such as influenza surveillance [2, 99, 286] and earthquakes [283, 284], none of them focuses on predicting the air quality information. Note that although Jiang et al. [179] and Wang et al. [336] exploit social media to infer AQIs at current or past time, they cannot predict the future air quality. Moreover, the AQIs in these previous studies usually have considerable fluctuations, under which circumstance users tend to publish related posts, which makes the inference task much more manageable than general cases. In general cases, air quality changes gradually most time, which may be not sufficiently documented in social media. For instance, in California, more than 80% of the changes in air quality conditions are between good and moderate.

In this work, we aim to leverage social media for air quality prediction. Our approach consists of three stages, including (1) tweet filtering, (2) feature extraction, and (3) air quality prediction. In the first stage, all of the incoming tweets are filtered by geographical locations and keywords extracted from statistical and topical modeling. After filtering the tweets, a convolutional neural network is applied to extract the individual feature vector for each tweet with a max-over-time pooling layer. A max-over-tweet layer is then proposed to aggregate

¹¹For simplicity, the posts published on social media are called *tweets* in this work.

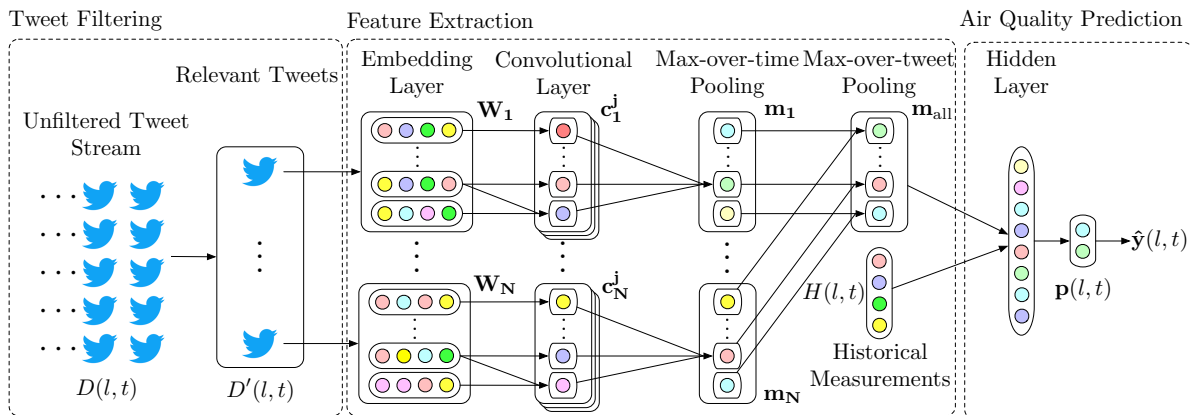


Figure 13.1: The framework of the proposed approach.

the feature vectors of all tweets as the social media features for predicting air quality using a fully-connected hidden layer to combine with historical measurements. Finally, experiments conducted on 7-month large-scale Twitter datasets show that our approach significantly outperforms all comparative baselines.

13.2 Air Quality Prediction with Social Media and NLP

Following the previous studies [377], we model the problem as a multi-class classification task. According to the Environmental Protection Agency ¹² (EPA) in USA, AQIs can be categorized into six classes as shown in Figure 13.1. Note that more than 99% of daily AQIs in the USA are similar and falling in the first two classes so that the classification task is more laborious than predicting numerical AQIs. Given a location l and a time t , the corpus $D(l, t)$ is defined as the N tweets published by any user located at the location l at time t . $a(l, t)$ denotes the AQI value in the location l at time t while the historical measurements $H(l, t) = a(l, t), a(l, t - 1), \dots, a(l, t - T + 1)$ provide AQIs at T time points. Given the corpus $D(l, t)$ and the historical measurements $H(l, t)$ at location l at time t , our goal is to predict the corresponding class y of the AQI at the next time point $t + 1$.

Framework Overview. Figure 13.1 illustrates the proposed three-stage framework. In the

¹²EPA: <https://www.epa.gov/>

AQI	Level of Concern
0-50	Good
51-100	Moderate
101-150	Unhealthy for Sensitive Groups
151-200	Unhealthy
201-300	Very Unhealthy
301-500	Hazardous

Table 13.1: Categorization of AQI from EPA.

first stage, the incoming tweets are filtered to remove irrelevant information. In the second stage, representative features are extracted from filtered tweets and historical measurements. In the last stage, we predict the category of air quality with a hidden layer and a softmax function.

13.2.1 Stage 1: Tweet Filtering

In most of the cities, the majority of tweets should be irrelevant to air quality because users are less likely to discuss air quality situations unless there is a dramatic change. Hence, we need to filter tweets before using them for air quality prediction. Following the previous work [235], we use three groups of keywords for filtering tweets, including **(1) environment-related terms** like *smog* released by EPA, **(2) health-related terms** like *choke* provided by the National Library of Medicine¹³, and **(3) significant terms** including the most significant 128 words correlated to high AQIs in χ^2 statistics [288].

The incoming tweets are filtered by the aforementioned keywords in the three groups. The tweets containing at least one of these keywords are likely to be relevant to the topics about air quality. We denote the corpus of relevant tweets as $D'(l, t)$. The features extracted from relevant tweets are expected to be more robust.

¹³<https://www.nlm.nih.gov/medical-terms.html>

13.2.2 Stage 2: Feature Extraction

To extract features from text data, the effectiveness of convolutional neural networks (CNNs) has been demonstrated in many studies [194]. In this work, CNNs with max-over-time pooling are applied to derive the representation for every tweet. We then propose *max-over-tweet pooling* to aggregate tweet representations across all relevant tweets as the corpus representation. Finally, the features can be acquired by concatenating the corpus representation and the historical measurements for prediction.

Tweet Representation. A tweet \mathbf{w}_i can be represented by a matrix $\mathbf{W}_i \in R^{d \times |\mathbf{w}_i|}$, where d is the dimension of word embeddings; and $|\mathbf{w}_i|$ is the number of words in the tweet. As shown in Figure 13.1, a CNN with $d \times k$ kernels extracts the n -gram semantics of k contiguous words. Note that the row dimension of kernels is identical to the word embedding dimension to jointly consider the overall embedding vector. The convolution with the j -th kernel produces a numerical vector \mathbf{c}_i^j , which is then aggregated by max-over-time pooling [75, 194]. As a result, the representation of a tweet \mathbf{m}_i can be derived by chaining the pooled results of all kernels.

Corpus Representation. Since relevant tweets in the corpus can be myriad and not fixed, we need to aggregate various representations into an ultimate representation for the whole corpus. Here we propose max-over-tweet pooling to derive the corpus representation. The layer of max-over-tweet pooling reads all tweet representations and aggregates them by deriving the maximum value for each representation dimension. More precisely, a dimension of the representation can be treated as the sensor about a particular topic while the max-over-tweet pooling layer attempts to find the maximum sensor value among the sensor values of all relevant tweets. Finally, the max-over-tweet pooling layer can derive the corpus representation \mathbf{m}_{all} by considering all tweet representations.

After determining the corpus representation \mathbf{m}_{all} , the final features $\mathbf{x}(l, t)$ for air quality prediction can be constructed by concatenating \mathbf{m}_{all} and the historical measurements $H(l, t)$. As a consequence, the final features incorporate the knowledge of existing observations and the crowd power on social media.

13.2.3 Stage 3: Air Quality Prediction

To address the air quality prediction, we apply a fully-connected hidden layer to estimate the logits of all classes. More precisely, the logits $\mathbf{z}(l, t)$ can be computed as $\mathbf{z}(l, t) = F(\mathbf{x}(l, t))$, where $F(\cdot)$ is a fully-connected hidden layer with L hidden units; the dimension of $\mathbf{z}(l, t)$ is identical to the number of classes in air quality categorization. Then the probabilistic score for each class can be obtained with a softmax function [131] when the prediction can be finally determined as the class with the highest score. Finally, the whole system can be computed and trained in an end-to-end manner and optimized by the cross-entropy loss [131].

13.3 Experiments

13.3.1 Experimental Settings

Data Collection. For social media data, we exploit the Twitter developer API¹⁴ to crawl 1% of general English tweets published in the USA with location tags from November 17, 2015, to June 12, 2016. Each of the crawled tweets is associated with the corresponding county and state. EPA releases daily AQIs for every county in the USA publicly, which serve as the historical measurements and the gold standard.

Experimental Datasets. We conduct experiments to predict daily air quality conditions for locations fine-grained to the county level. More specifically, each of the samples can be represented by a tuple (l, t) , where l is a county in the USA; t is a date with crawled tweets. For each tuple, the historical measures are the AQIs in the previous seven days as seven numerical features. Five experimental datasets are then constructed with the data of the five most polluted states according to the annual report from America Health Ranking¹⁵, including California (CA), Idaho (ID), Illinois (IL), Indiana (IN), and Ohio (OH). The overall datasets are further partitioned by time into a 30-week training dataset, two 5-week datasets

¹⁴<https://developer.twitter.com/en/docs.html>

¹⁵<https://www.americashealthrankings.org>

Dataset	CA	ID	IN	IL	OH
Overall tweets	85.3M	1.2M	9.2M	23.2M	31.7M
Relevant tweets	11.8M	0.07M	0.5M	1.0M	1.4M
Training tuples	7,435	1,175	2,990	1,804	3,647
Validation tuples	1,487	235	598	361	729
Testing tuples	1,483	235	599	361	730

Table 13.2: Statistics of five experimental datasets. The relevant tweets refer to the remaining tweets after the stage of tweet filtering.

for validation and testing. As a result, Table 13.2 shows the statistics of five experimental datasets. Note that more than 90% tweets are filtered as irrelevant tweets in the stage of tweet filtering. It also shows the necessity of filtering irrelevant tweets that can probably be noises for air quality prediction.

Implementation Details Our approach is implemented by Tensorflow [1] and trained by the Adam optimizer [196] with an initial learning rate 10^{-3} . After parameter tuning, λ is set to 10^{-3} while the number of hidden units in the hidden layer L is 128. The dimension of the word embeddings is 300. All of the activation functions in the model are set to exponential linear units (ELUs) [72]. For CNNs, 96 kernels with different sizes from 2 to 4 are applied to obtain a 96-dimensional representation for each relevant tweet in the corpus.

Baseline Methods. Because we are the first study using social media to predict air quality situation, there are much few available methods. Even though some studies [179] claim the capability of inferring ongoing AQIs with social media, they apply strong restrictions to derive features for highly polluted cities so that they are incapable of tackling most of the cases in our experiments. In the experiments, we compare with two baseline methods as follows: **(1) Prediction with only AQIs (PAQI):** To understand the base performance, PAQI predicts the air quality conditions with only historical measurements. The knowledge of social media is ignored for this baseline method. **(2) Bag-of-words Features (BOW):** To demonstrate the effectiveness of extracted features, we replace the extracted features with conventional bag-of-words features as a baseline method. Note that all baselines apply a neural network with a hidden layer for prediction.

Dataset	Method	Micro Average			Macro Average		
		Prec.	Rec.	F1	Prec.	Rec.	F1
ID	BOW	0.807	0.829	0.809	0.687	0.619	0.631
	PAQI	0.816	0.728	0.757	0.611	0.677	0.617
	Ours	0.863	0.811	0.828	0.691	0.776	0.714
IN	BOW	0.792	0.786	0.786	0.508	0.508	0.501
	PAQI	0.847	0.682	0.737	0.567	0.649	0.548
	Ours	0.855	0.849	0.852	0.640	0.652	0.645
IL	BOW	0.775	0.802	0.791	0.506	0.499	0.484
	PAQI	0.834	0.686	0.737	0.580	0.666	0.566
	Ours	0.844	0.847	0.845	0.646	0.638	0.640
OH	BOW	0.744	0.780	0.760	0.515	0.512	0.510
	PAQI	0.800	0.683	0.724	0.569	0.622	0.562
	Ours	0.813	0.813	0.815	0.629	0.627	0.627
CA	BOW	0.647	0.683	0.660	0.495	0.488	0.485
	PAQI	0.826	0.725	0.745	0.700	0.772	0.694
	Ours	0.830	0.786	0.798	0.728	0.786	0.742

Table 13.3: The overall classification performance of the baseline methods and our approach. All of the improvements of our approach (ours) over PAQI are significant with a paired t-test at a 99% significance level.

13.3.2 Experimental Results

For evaluation, micro- and macro-F1 scores are selected the evaluation metrics. Table 13.3 demonstrates the performance of the three methods. Micro-F1 scores are generally better than macro-F1 scores because the trivial cases like the class of good air quality are the majority of datasets with higher weights in micro-F1 scores. PAQI is better than BOW although BOW uses the knowledge of social media. It is because BOW features involve all irrelevant words so that the actual essential knowledge cannot be recognized. Our approach significantly outperforms all baseline methods in almost all metrics. More precisely, our approach improves the air quality prediction over PAQI from 6.92% to 17.71% in macro-F1 scores. The results demonstrate that social media and NLP can benefit air quality prediction.

In addition to the unbalanced datasets based on the categorization of EPA, we also conduct the experiments with relatively balanced datasets to show the robustness of our proposed approach. More specifically, the categorization is refined to four classes with finer

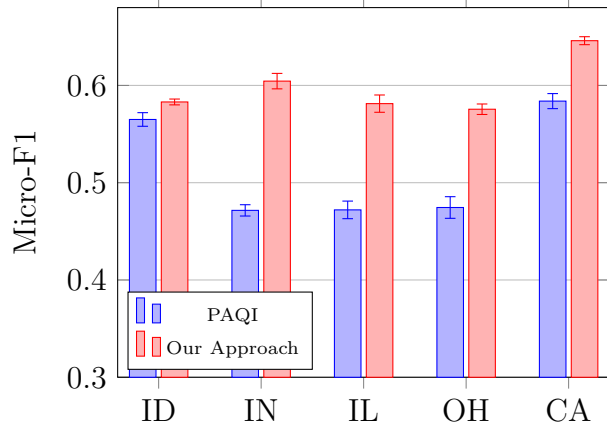


Figure 13.2: Micro F1 scores with four-class categorization. All of the improvements of our approach over the baseline method are significant with a paired t-test at a 99% significance level.

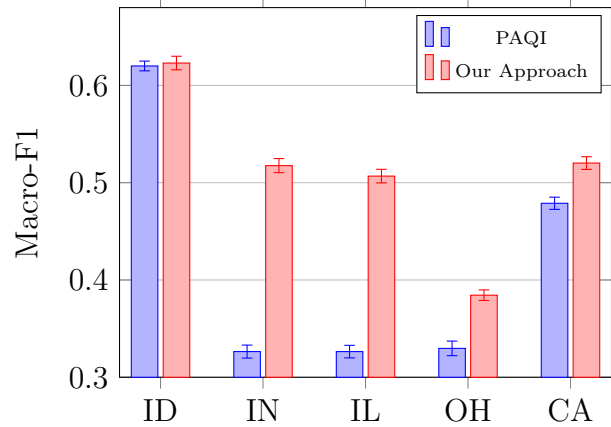


Figure 13.3: Macro F1 scores with four-class categorization. All of the improvements of our approach over the baseline method are significant with a paired t-test at a 99% significance level.

windows of AQIs, including: $[0, 25)$, $[25, 50)$, $[50, 75)$, and $[75, \infty)$. Figures 13.2 and 13.3 illustrate the Micro- and Macro-F1 scores of PAQI and our approach in the refined datasets. The experimental results show that the improvements are consistent with the experiments in unbalanced datasets of extreme air quality prediction. It also demonstrates the robustness of our proposed approach.

13.4 Conclusion and Discussions

In this chapter, we propose a novel framework for leveraging social media and NLP to air quality prediction. After filtering irrelevant tweets, a CNN derives a feature vector for each tweet with max-over-time pooling. We also propose the novel max-over-tweet pooling to aggregate the feature vectors of all tweets over numerous hidden topics. Finally, the corpus representation can be taken into account to predict air quality with historical measurements. The results of extensive experiments show that our proposed approach significantly outperforms two comparative baseline methods across both balanced and unbalanced datasets for different locations in the USA. This is because: (1) Most noisy and irrelevant tweets are effectively filtered in the stage of tweet filtering; (2) The convolutional neural network and

the proposed max-over-tweets are able to extract essential knowledge about air quality prediction from myriad tweets in social media; (3) There are some limitations on only using historical measurements, such as the capability of recognizing real-world events.

CHAPTER 14

Conclusion

We have introduced a three-layer universal framework to model complex human behaviors at different scales. In the first layer, we develop effective and efficient machine learning methods to derive robust representations for heterogeneous data. Precisely, we leverage the structural semantics in diverse data to not only encode more knowledge into representations but also reduce redundant computations for better efficiency. In the second layer, we learn representations of diverse resources in a universal latent space and estimate the importance of each resource with the aspect attention, thereby deriving satisfactory joint representations for heterogeneous data. In the third layer, we treat the representation learning methods as atomic blackboxes so that domain experts can conveniently conduct compositional and explainable operations to describe complex human behaviors at different scales for machine learning methods. The contributions of this dissertation in advancing human behavior modeling with heterogeneous data can be further summarized as follows:

- Our proposed end-to-end framework comprehensively addresses the research problems through all stages of human behavior modeling from harnessing heterogeneous data to tackling behaviors at different scales. We also provide various concrete examples with real-world applications and sufficient experiments for each layer.
- We propose various machine learning algorithms to capture structural semantics from heterogeneous data resources, such as texts (Chapter 3), sequences (Chapter 4), networks (Chapter 5), for deriving better machine-readable representations. The extensive experiments indicate that representations with structural semantics consistently lead to more satisfactory results than the outcomes of state-of-the-art approaches for downstream applications across different data resources.

- We present how to improve the efficiency of machine learning methods with the semantic structure for harnessing heterogeneous data, especially in the serving stage that is extremely important for practical applications. CANTOR (Chapter 6) finds the core-sets of user affinity groups to accelerate the inference of generating recommendations for latent factor models. TahcoRoll (Chapter 7) fits biological sequences in a binary structural automaton to speed up the process of signature profiling. These results demonstrate our contributions in addressing the research problems of high volume and high velocity in the era of big data.
- We integrate heterogeneous data with the universal latent space and the aspect attention. SHE-UI (Chapter 8) leverages the connections between entities with different types and derives universal representations with graph embedding algorithms for identifying users behind shared accounts. SPoD (Chapter 9) and HUG (Chapter 10) apply the aspect attention to model completely distinct resources in social media posts (i.e., texts, images, networks) to tackle the research problems of sponsor detection and user geolocation. The empirical results show that methods with universal and aspect-attentive attention significantly outperform conventional methods that independently consider different resources.
- We demonstrate the potential impacts of our proposed framework to model multi-scale human behaviors with concrete examples. For individual behaviors, RIN (Chapter 11) learns reformulation behaviors with homomorphic query embedding for query suggestion in web search. For interpersonal behaviors, SHCNN (Chapter 12) establishes the similarity graph of text messages by estimating the semantic similarity for disentangling multi-party conversations. For community behaviors, we apply deep learning methods to capture collective reflection of social media users to the environment for enhancing air quality prediction.

APPENDIX A

Appendix in QDS-Transformer

A.1 Experimental Details

In this section, we clarify the details about experimental datasets and experimental settings.

A.1.1 Experimental Datasets

TREC-19 DL Track Dataset. For ad-hoc retrieval, we adopt the TREC-19 DL track benchmark as the experimental dataset with training, dev, and test sets. Training and dev sets consist of large-scale human relevance assessments derived from the MS MARCO collection [25] with no negative labels and sparse positive labels for each query while relevance judgments in the test sets are annotated by NIST judges.

Few-shot Document Ranking Benchmarks. For few-shot learning, three retrieval benchmark datasets are utilized in our experiments, including Robust04, ClueWeb09-B, and ClueWeb12-B13. Robust04 provides 249 queries from TREC Robust track 2014 with relevance labels. ClueWeb09-B includes of 200 queries with relevance labels from TREC Web Track 2009-2012. ClueWeb12-B13 consists of 100 queries from TREC Web Track 2013-2014 with relevance labels.

Note that Table 3.1 in the paper summarizes the statistics of four experimental datasets. Datasets of all benchmarks are publicly available. The TREC-19 DL track provides all dataset on its official website¹⁶. The queries and relevance assessments of three few-shot

¹⁶<https://microsoft.github.io/TREC-2019-Deep-Learning/>

document ranking datasets can be found at the TREC website¹⁷ while document collocations are also publicly available on the corresponding sites¹⁸¹⁹²⁰.

A.1.2 Experimental Settings

Ad-hoc Retrieval. Experiments follow the protocol of the TREC-19 deep learning track. Each method is trained with the training set. The model parameters can be further fine-tuned with the dev set and the MRR@10 metric. The fine-tuned model is finally applied to the test set for evaluation. Following the official metrics, MRR@10 is used in dev set runs as labels are incomplete and shallow, while the test set is comprehensively evaluated using NDCG@10 and MAP@10.

Few-shot Document Ranking. All experimental settings for few-shot learning are consistent with the “MS MARCO Human Labels” setting in previous studies [365]. Each method first trains a neural ranker on MARCO training labels, which are identical as in the TREC DL track. The latent representations of trained models are then considered as features for a Coor-Ascent ranker for low-label datasets using five-fold cross-validation [83, 84] to rerank top-100 SDM retrieved results [239]. Standard metrics NDCG@20 and ERR@20 are used to compare the different approaches. The results are reported by taking the average of each test fold from the total 5 folds, wherein the rest 4 folds in each round are used as training and dev queries.

Hyperparameter Settings and Search. We adopt the pretrained model for sparse attention [31] and fix all of the hidden dimension numbers as 768 and the number of transformer layers as 12. BERT-based models use RoBERTa as pretrained models [222]. To hyperparameter tuning, we search the local attention window size w in {32, 64, 128, 256, 512, 1024} with the dev set and determine $w = 128$. Models are optimized by the Adam optimizer [196]

¹⁷<https://trec.nist.gov/>

¹⁸RB04: https://trec.nist.gov/data/qa/T8-QAdata/disks4_5.html

¹⁹CW09: <http://lemurproject.org/clueweb09/>

²⁰CW12: <https://lemurproject.org/clueweb12/>

with a learning rate 10^{-5} , $(\beta_1, \beta_2) = (0.9, 0.999)$, and a dropout rate 0.1. Under the hyperparameter settings, the parameter numbers of our implemented methods are shown in Table A.1 summarizing the sizes of parameters based on `model.parameters()` in PyTorch.

Method	#Params	Method	#Params
RoBERTa (FirstP)	124M	RoBERTa (MaxP)	124M
Sparse-Transformer	149M	Longformer-QA	149M
Transformer-XH	128M	QDS-Transformer	149M

Table A.1: Number of parameters for methods.

A.1.3 Evaluation Scripts

All evaluation measures are computed by the official scripts. For ad-hoc retrieval, we use `trec_eval`²¹ as the standard tool in the TREC community for evaluating ad-hoc retrieval runs. This is also the official setting of the TREC-19 deep learning track. For few-shot document ranking, we use graded relevance assessment script (`gdeval`)²² as the evaluation script measuring NDCG and ERR. Note that this setting is consistent with previous studies [83, 365].

A.2 Baseline Methods

In this section, we introduce each baseline method.

TREC Best Runs.

- **bm25tuned_prf** [352] fine-tunes the BM25 parameters with pseudo relevance feedback as the best BM25 based method in official runs.
- **srchvrs_run1** is marked as the best traditional ranking method among official runs [80].
- **TUW19-d3-re** [153] as the best method without using non-pretrained language models (non-PLM) in official runs utilizes a transformer to encode both of the query and the

²¹https://github.com/usnistgov/trec_eval

²²<https://trec.nist.gov/data/web/10/gdeval.pl>

document, thereby measuring interactions between terms and scoring the relevance.

- **bm25_expmarcomb** [7] combines sentence-level and document-level relevance scores with a pretrained BERT model.

Classical IR Methods.

- **SDM** [238] as a sequential dependence model conducts ranking based on the theory of probabilistic graphical models. We obtain ranking results of SDM from previous studies [83]. SDM is not only treated as a baseline method but also providing the candidate documents for reranking in the few-shot learning task.
- **Coor-Ascent** [239] is a linear feature-based model for ranking. It is also considered as the trainer in few-shot learning with representations from methods.

Neural IR Methods.

- **CO-PACRR** [163] utilizes CNNs to model query-document similarity matrices and provide a score using a max-pooling layer.
- **Conv-KNRM** [84] applies CNNs to independently encode the query and the document. The encoded representations are then integrated by a cross-matching layer, thereby deriving relevance scores.

Transformer-based Methods.

- **TK** [154] and **TKL** [155] apply transformers to independently model the query and document, thereby measuring term interactions at the embedding level.
- **RoBERTa (FirstP)** and **RoBERTa (MaxP)** [83] adapt long-form documents by considering the first paragraph and combining RoBERTa outputs with max-pooling over paragraphs. Note that each paragraph is also attached with query tokens before being fed into the model.
- **Transformer-XH** [372] encodes each sentence independently and considers their relations with an extra-hop attention layer. Note each sentence is also attached with query tokens as the model input.
- **Sparse-Transformer** [65] simply uses sparse local attention to tackle the efficiency issue of transformers.

- **Longformer-QA** [31] extends Sparse-Transformer by attaching two global attention tokens to the query and the document as their settings for question answering. Note that their global attention would not consider document structural information.

APPENDIX B

Appendix in MARU

B.1 The proof of Corollary 5.1

Proof. For the one-directional random walk, the expected number of visited tail nodes is $E_o = 1 + \sum_{i=0}^{2n} i \cdot p^i \cdot (1-p)$. For the bidirectional random walk, the expected number of visited tail nodes is $E_b = 1 + 2 \cdot \sum_{i=0}^n i \cdot p^i \cdot (1-p)$. Therefore, we have

$$\begin{aligned} \lim_{n \rightarrow \infty} E_b - E_o &= \lim_{n \rightarrow \infty} (1-p) \cdot \left(\sum_{i=1}^n i \cdot p^i - (i+n) \cdot p^{i+n} \right) \\ &= \lim_{n \rightarrow \infty} \frac{p \cdot (1-p^n) \cdot (2 \cdot n \cdot p^{n+1} - (2n+1) \cdot p^n + 1)}{1-p} \\ &= \frac{p}{1-p} > 0 \end{aligned}$$

□

APPENDIX C

Appendix in CANTOR

C.1 Proof of Theorem 6.1

Proof. Without loss of generality, we assume that vectors in \mathbb{A}_t , \mathbb{Q} , and s_t have unit norms. $\forall q \in \mathbb{Q}, i \in \mathbb{A}_t$, we have:

$$\begin{aligned} |p_i q^T - \mathcal{N}_{s_t}(p_i) q^T| &= |(p_i - \mathcal{N}_{s_t}(p_i)) q^T| \\ &\stackrel{(a)}{\leq} \sqrt{d} \|p_i - \mathcal{N}_{s_t}(p_i) q^T\|_2 \leq \sqrt{d} \|p_i - \mathcal{N}_{s_t}(p_i)\|_2 \leq \sqrt{d} \|p_i - \mathcal{N}_{s_t}(p_i)\|_2^2 \\ &= \sqrt{d} (\|p_i\|_2^2 + \|\mathcal{N}_{s_t}(p_i)\|_2^2 - 2\mathcal{N}_{s_t}(p_i) p_i^T) \stackrel{(b)}{\leq} \sqrt{d} [2 - 2\epsilon] = \delta, \end{aligned}$$

where we define $\delta = \sqrt{d} [2 - 2\epsilon]$. (a) follows from the fact that $\|\cdot\|_1 \leq \sqrt{d} \|\cdot\|_2$, where d is the dimension of the vector. (b) follows from the condition of theorem. \square

C.2 Proof of Theorem 6.2

Proof. Since s_t is a ϵ set cover of p_i s, there exist a δ such that s_t is a δ -user coreset of p_i s. Therefore, for any given query q and vector p_t sampled from $\mathbb{P}_{\mathbb{A}_t}$, we have

$$\begin{aligned} |\mathcal{N}_{s_t}(p_i) q^T - p_t q^T| &= |\mathcal{N}_{s_t}(p_i) q^T - p_i q^T + p_i q^T - p_t q^T| \\ &\leq |\mathcal{N}_{s_t}(p_i) q^T - p_i q^T| + |p_i q^T - p_t q^T| \leq \delta + |p_i q^T - p_t q^T| \end{aligned}$$

Since p_i and p_t follow the same distribution, p_i and p_t will have same expectation value and we have:

$$\begin{aligned}
\mathbb{E}[|\mathcal{N}_{s_t}(p_i)q^T - p_tq^T|] &\leq \mathbb{E}[\delta + |p_iq^T - p_tq^T|] \\
&= \delta + \mathbb{E}[|p_iq^T - p_tq^T|] \\
&\stackrel{(a)}{\leq} \delta + |\mathbb{E}[p_iq^T] - \mathbb{E}[p_tq^T]| \\
&= \delta,
\end{aligned}$$

where (a) follows the Jensen's inequality. Therefore, by Hoeffding's inequality, with probability at least $1 - \gamma$,

$$\frac{1}{k} \sum_{i=1}^k |\mathcal{N}_{s_t}(p_i)q^T - p_tq^T| \leq \delta + \sqrt{\frac{2 \log(1/\gamma)}{k}}.$$

By the fact that for any set S , $\min(S) \leq \text{mean}(S)$, we will have:

$$\begin{aligned}
\min_i (|\mathcal{N}_{s_t}(p_i)q^T - p_tq^T|) &\leq \frac{1}{k} \sum_{i=1}^k |\mathcal{N}_{s_t}(p_i)q^T - p_tq^T| \\
&\leq \delta + \sqrt{\frac{2 \log(1/\gamma)}{k}},
\end{aligned}$$

□

APPENDIX D

Appendix in TahcoRoll

D.1 Proof of Proposition 7.1

Proof. Given the prefix length i and the number of possible characters c , there are c^i possible prefixes in total. Assuming the characters are uniformly distributed, the probability that a particular prefix exists in n signatures is:

$$1 - \left(1 - \frac{1}{c^i}\right)^n.$$

Therefore, the expected number of collided prefixes is:

$$n - c^i \left(1 - \left(1 - \frac{1}{c^i}\right)^n\right),$$

and the expected number of prefixes without any collision is:

$$n - \left(n - c^i \left(1 - \left(1 - \frac{1}{c^i}\right)^n\right)\right) = c^i \left(1 - \left(1 - \frac{1}{c^i}\right)^n\right) = c^i \left(1 - \left(\frac{c^i - 1}{c^i}\right)^n\right).$$

However, the expected number above includes the cases that fail before reaching the i -th character. Hence, the expected number of these cases should be deducted from the above number. Finally, the expected number of signatures that fail to find their length- i prefixes along the trie during its insertion is:

$$c^i \left(1 - \left(\frac{c^i - 1}{c^i}\right)^n\right) - c^{i-1} \left(1 - \left(\frac{c^{i-1} - 1}{c^{i-1}}\right)^n\right).$$

□

D.2 Proof of Proposition 7.2

From Proposition 7.1, the expected number of node for prefix length i in n signatures is $c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right)$. Intuitively, summing all possible prefix lengths up to the length of signature m , the expected number of trie nodes is $\sum_{i=1}^m c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right)$. We provide the proper proof below.

Proof. Denote the expected number of signatures that fail to find their length- i prefixes on the trie during its insertion as $f(i)$. Given the length of signatures m , each signature that fails to find the length- i prefix along the trie during its insertion will result in the addition of $m - i + 1$ nodes. Based on Proposition 1, the expected number of nodes in the trie is:

$$\begin{aligned}
& \sum_{i=1}^m (m - i + 1) \cdot f(i) \\
&= \sum_{i=1}^m (m - i + 1) \left[c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right) - c^{i-1} \left(1 - \left(\frac{c^{i-1}-1}{c^{i-1}}\right)^n\right) \right] \\
&= \sum_{i=1}^m (m - i + 1) \left[c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right) \right] - \sum_{i=1}^{m-1} (m - i) \left[c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right) \right] \\
&= \sum_{i=1}^{m-1} [(m - i + 1) - (m - i)] \left[c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right) \right] + c^m \left(1 - \left(\frac{c^m-1}{c^m}\right)^n\right) \\
&= \sum_{i=1}^m c^i \left(1 - \left(\frac{c^i-1}{c^i}\right)^n\right).
\end{aligned}$$

□

D.3 Proof of Proposition 7.3

Proof. Suppose that the number of signatures to be added into a trie is extremely large. The expected number of nodes with c possible characters is:

$$\lim_{n \rightarrow \infty} \sum_{i=1}^m c^i \left(1 - \left(\frac{c^i - 1}{c^i} \right)^n \right) = \sum_{i=1}^m c^i = \frac{c(c^m - 1)}{c - 1} = \frac{c^{m+1} - c}{c - 1}.$$

For the plain AC, there are four possible characters, i.e., A, C, G and T. Hence, the expected number of its nodes N_A is:

$$N_A = \frac{4^{m+1} - 4}{4 - 1} = \frac{(2^2)^{m+1} - 4}{3} = \frac{2^{2m+2} - 4}{3} = \frac{4(2^{2m} - 1)}{3}.$$

For the thinned automaton, there are two possible characters, i.e., 0 and 1. Hence, the expected number of its nodes N_T is:

$$N_T = \frac{2^{m+1} - 2}{2 - 1} = 2(2^m - 1).$$

Finally, we compute the ratio of the expected number of two approaches as follows:

$$\frac{N_T}{N_A} = \frac{2(2^m - 1)}{\frac{4(2^{2m} - 1)}{3}} = \frac{3}{2} \cdot \frac{2^m - 1}{2^{2m} - 1} = \frac{3}{2} \cdot \frac{2^m - 1}{(2^m + 1)(2^m - 1)} = \frac{3}{2} \cdot \frac{1}{2^m + 1}.$$

□

BIBLIOGRAPHY

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *OSDI*, 2016, pp. 265–283. 53, 165, 199, 218, 233
- [2] H. Achrekar, A. Gandhe, R. Lazarus, S.-H. Yu, and B. Liu, “Predicting flu trends using twitter data,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011, pp. 702–707. 228
- [3] Activate, “Exploring the brand and influencer relationship in influencer marketing,” *State of Influencer Marketing Study*, 2018. 151, 152
- [4] M. Aharon, E. Hillel, A. Kagian, R. Lempel, H. Makabee, and R. Nissim, “Watch-it-next: A contextual tv recommendation system,” in *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, ser. ECML PKDD ’15, 2015, pp. 180–195. 18, 130
- [5] A. V. Aho and M. J. Corasick, “Efficient string matching: an aid to bibliographic search,” *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=360825.360855> 109, 112
- [6] J.-i. Akahani, K. Hiramatsu, and K. Kogure, “Coordinating heterogeneous information services based on approximate ontology translation,” in *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*. Citeseer, 2002, pp. 10–14. 20, 187, 188
- [7] Z. Akkalyoncu Yilmaz, S. Wang, and J. Lin, “H2oloo at trec 2019: Combining sentence and document evidence in the deep learning track,” in *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC 2019)*, 2019. 242
- [8] L. Akoglu, R. Chandy, and C. Faloutsos, “Opinion fraud detection in online reviews by network effects.” *ICWSM*, vol. 13, pp. 2–11, 2013. 19

- [9] J. Allan, “Introduction to topic detection and tracking,” *Topic detection and tracking*, pp. 1–16, 2002. 209
- [10] S. K. Ames, D. A. Hysom, S. N. Gardner, G. S. Lloyd, M. B. Gokhale *et al.*, “Scalable metagenomic taxonomy classification using a reference genome database.” *Bioinformatics*, vol. 29, no. 18, pp. 2253–60, 2013. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/23828782><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3753567><http://www.ncbi.nlm.nih.gov/pubmed/23828782>{%}5Cn<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3753567> 107
- [11] H. Amiri, P. Resnik, J. Boyd-Graber, and H. Daumé III, “Learning text pair similarity with context-sensitive autoencoders,” in *ACL’16*. ACL, 2016, pp. 1882–1892. 21
- [12] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012. 135
- [13] P. M. Aoki, M. H. Szymanski, L. Plurkowski, J. D. Thornton, A. Woodruff, and W. Yi, “Where’s the party in multi-party?: Analyzing the structure of small-group sociable talk,” in *CSCW’06*. ACM, 2006, pp. 393–402. 208, 221
- [14] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. 78, 143, 144, 145
- [15] R. Ashwal-Fluss, M. Meyer, N. R. Pamudurti, A. Ivanov, O. Bartok, M. Hanan, N. Evantal, S. Memczak, N. Rajewsky, and S. Kadener, “circrna biogenesis competes with pre-mrna splicing,” *Molecular cell*, vol. 56, no. 1, pp. 55–66, 2014. 41
- [16] K. F. Au, V. Sebastiano, P. T. Afshar, J. D. Durruthy, L. Lee *et al.*, “Characterization of the human esc transcriptome by hybrid sequencing,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 50, pp. E4821–E4830, 2013. 119

- [17] P. Audano and F. Vannberg, “KAnalyze: A fast versatile pipelined K-mer toolkit,” *Bioinformatics*, vol. 30, no. 14, pp. 2070–2072, 2014. [Online]. Available: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu152> 17
- [18] E. Aumayr, J. Chan, and C. Hayes, “Reconstruction of threaded conversations in online discussion forums.” in *ICWSM’11*, 2011, pp. 26–33. 21, 214
- [19] A. S. Authority, “The labelling of influencer advertising,” <https://www.asa.org.uk/uploads/assets/uploaded/e3158f76-ccf2-4e6e-8f51a710b3237c43.pdf>, 2019. 151, 153
- [20] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet, “Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces,” in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 257–264. 16, 100
- [21] L. Backstrom and J. Leskovec, “Supervised random walks: predicting and recommending links in social networks,” in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644. 85
- [22] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014. 188, 193, 195
- [23] —, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations, ICLR 2015*, 2015. 47, 49, 53
- [24] P. Bajaj and S. Shekhar, “Experience individualization on online tv platforms through persona-based account decomposition,” in *Proceedings of ACM International Conference on Multimedia*, ser. MM ’16, 2016, pp. 252–256. 18
- [25] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen *et al.*, “Ms marco: A human generated machine reading comprehension dataset,” *arXiv preprint arXiv:1611.09268*, 2016. 10, 25, 239

- [26] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, “Everyone’s an influencer: quantifying influence on twitter,” in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 65–74. [19](#)
- [27] G. Ballard, T. G. Kolda, A. Pinar, and C. Seshadhri, “Diamond sampling for approximate maximum all-pairs dot-product (mad) search,” in *2015 IEEE International Conference on Data Mining*. IEEE, 2015, pp. 11–20. [100](#)
- [28] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski *et al.*, “Spades: a new genome assembly algorithm and its applications to single-cell sequencing,” *Journal of computational biology*, vol. 19, no. 5, pp. 455–477, 2012. [108](#)
- [29] Z. Bar-Yossef and N. Kraus, “Context-sensitive query auto-completion,” in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 107–116. [20](#), [187](#), [198](#), [199](#), [201](#)
- [30] S. P. Barrett and J. Salzman, “Circular rnas: analysis, expression and potential functions,” *Development*, vol. 143, no. 11, pp. 1838–1847, 2016. [11](#), [41](#), [43](#), [56](#)
- [31] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” *arXiv:2004.05150*, 2020. [11](#), [27](#), [29](#), [30](#), [240](#), [243](#)
- [32] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013. [2](#), [210](#)
- [33] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, “An updated set of basic linear algebra subprograms (blas),” *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002. [15](#), [101](#)
- [34] P. Blunsom, E. Grefenstette, and N. Kalchbrenner, “A convolutional neural network for modelling sentences,” in *ACL’14*. ACL, 2014, pp. 655–665. [210](#)

- [35] J. Bobadilla, F. J. Ortega, A. Hernando, and A. Gutierrez, “Recommender systems survey,” *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013. 148
- [36] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *arXiv preprint arXiv:1607.04606*, 2016. 218
- [37] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna, “Query suggestions using query-flow graphs,” in *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 2009, pp. 56–63. 20, 187
- [38] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna, “From” dango” to” japanese cakes”: Query reformulation models and patterns,” in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT’09. IEEE/WIC/ACM International Joint Conferences on*, vol. 1. IEEE, 2009, pp. 183–190. 20, 187
- [39] C. E. Bonferroni, “Il calcolo delle assicurazioni su gruppi di teste,” *Studi in onore del professore salvatore ortu carboni*, pp. 13–60, 1935. 127
- [40] P. Bork, L. J. Jensen, C. Von Mering, A. K. Ramani, I. Lee, and E. M. Marcotte, “Protein interaction networks from yeast to human,” *Current opinion in structural biology*, vol. 14, no. 3, pp. 292–299, 2004. 96
- [41] M. Boss and C. Arenz, “A fast and easy method for specific detection of circular rna by rolling-circle amplification,” *ChemBioChem*, 2019. 41
- [42] L. Bottou, “Stochastic gradient learning in neural networks,” *Proceedings of Neuro-Nimes*, vol. 91, no. 8, 1991. 136
- [43] N. L. Bray, H. Pimentel, P. Melsted, and L. Pachter, “Near-optimal probabilistic RNA-seq quantification,” *Nature biotechnology*, vol. 34, no. 5, pp. 525–527, 2016. [Online]. Available: <http://arxiv.org/abs/1505.02710><http://www.nature.com/doi/10.1038/nbt.3519> 107

- [44] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” 2020. [36](#)
- [45] C. J. Burges, R. Ragno, and Q. V. Le, “Learning to rank with nonsmooth cost functions,” in *Advances in neural information processing systems (NIPS)*, 2007, pp. 193–200. [166](#), [167](#)
- [46] C. J. Burges, “From ranknet to lambdarank to lambdamart: An overview,” *Learning*, vol. 11, no. 23-581, p. 81, 2010. [200](#)
- [47] H. Cai, V. W. Zheng, and K. Chang, “A comprehensive survey of graph embedding: problems, techniques and applications,” *TKDE*, 2018. [62](#)
- [48] J. P. Callan, “Passage-level evidence in document retrieval,” in *SIGIR’94*. Springer, 1994, pp. 302–310. [22](#), [28](#)
- [49] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li, “Context-aware query suggestion by mining click-through and session data,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 875–883. [20](#), [186](#), [187](#)
- [50] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li, “Towards context-aware search by learning a very large variable length hidden markov model from search logs,” in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 191–200. [20](#), [187](#)
- [51] Q. Cao, X. Yang, J. Yu, and C. Palow, “Uncovering large groups of active malicious accounts in online social networks,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 477–488. [19](#)

- [52] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, “Learning to rank: from pairwise approach to listwise approach,” in *Proceedings of the 24th International Conference on Machine Learning (ICML)*. ACM, 2007, pp. 129–136. 166, 167
- [53] O. Celma, *Music Recommendation and Discovery in the Long Tail*. Springer, 2010. 99, 141, 148
- [54] M. Chaabane, R. M. Williams, A. T. Stephens, and J. W. Park, “circdeep: deep learning approach for circular rna classification from other long non-coding rna,” *Bioinformatics*, vol. 36, no. 1, pp. 73–80, 2020. 12, 41, 42, 46, 51, 52, 54
- [55] H. Chae, J. Park, S.-W. Lee, K. P. Nephew, and S. Kim, “Comparative analysis using K-mer and K-flank patterns provides evidence for CpG island sequence evolution in mammalian genomes,” *Nucleic acids research*, vol. 41, no. 9, pp. 4783–91, 2013. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/23519616><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3643570> 107, 108, 127
- [56] Y. D. Challenge, “Yelp dataset challenge,” 2013. 74, 75
- [57] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, “Heterogeneous network embedding via deep architectures,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128. 13, 14
- [58] S. Chaudhuri and R. Kaushik, “Extending autocompletion to tolerate errors,” in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 707–718. 20, 187
- [59] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, “Harp: Hierarchical representation learning for networks,” in *AAAI*, 2018. 13
- [60] L. Chen, Y.-H. Zhang, G. Huang, X. Pan, S. Wang, T. Huang, and Y.-D. Cai, “Discriminating circrnas from other lncrnas using a hierarchical extreme learning machine

- (h-elm) algorithm with feature selection,” *MGG*, vol. 293, no. 1, pp. 137–149, 2018. 11, 41
- [61] P. Chen, S. Si, S. Kumar, Y. Li, and C.-J. Hsieh, “Learning to screen for fast softmax inference on large vocabulary neural networks,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=ByeMB3Act7> 16, 86, 87, 100
- [62] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, “{TVM}: An automated end-to-end optimizing compiler for deep learning,” in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 578–594. 24, 29, 31
- [63] T. Chen and Y. Sun, “Task-guided and path-augmented heterogeneous network embedding for author identification,” in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 295–304. 64
- [64] X. Chen, P. Han, T. Zhou, X. Guo, X. Song, and Y. Li, “circrnadb: a comprehensive database for human circular rnas with protein-coding annotations,” *Scientific reports*, vol. 6, no. 1, pp. 1–6, 2016. 51
- [65] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse transformers,” URL <https://openai.com/blog/sparse-transformers>, 2019. 11, 22, 23, 27, 29, 30, 242
- [66] W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, “Libmf: a library for parallel matrix factorization in shared-memory systems,” *JMLR*, vol. 17, no. 1, pp. 2971–2975, 2016. 100
- [67] C.-H. Cho, “Why do people avoid advertising on the internet?” *Journal of advertising*, vol. 33, no. 4, pp. 89–97, 2004. 161
- [68] H. Cho, J. Davis, X. Li, K. S. Smith, A. Battle *et al.*, “High-resolution transcriptome

analysis with long-read rna sequencing,” *PLoS One*, vol. 9, no. 9, p. e108095, 2014.

119

- [69] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014. 46, 193
- [70] D. M. Christopher, R. Prabhakar, and S. Hinrich, “Introduction to information retrieval,” *An Introduction To Information Retrieval*, vol. 151, p. 177, 2008. 217, 222
- [71] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does bert look at? an analysis of bert’s attention,” *arXiv*, 2019. 26, 36
- [72] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” in *ICLR*, 2016. 182, 233
- [73] —, “Fast and accurate deep network learning by exponential linear units (elus),” in *ICLR’16*, 2016. 217
- [74] J. D. Cohen, “Recursive hashing functions for n-grams,” *ACM Transactions on Information Systems (TOIS)*, vol. 15, no. 3, pp. 291–320, 1997. 116
- [75] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *JMLR*, vol. 12, pp. 2493–2537, 2011. 215, 231
- [76] F. T. Commission, “The ftc’s endorsement guides: What people are asking,” <https://www.ftc.gov/tips-advice/business-center/guidance/ftcs-endorsement-guides-what-people-are-asking>, 2017. 151, 153
- [77] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009. 66
- [78] G. Cormode and S. Muthukrishnan, “An improved data stream summary: The count-min sketch and its applications,” *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0196677403001913>
18

- [79] G. M. Correia, V. Niculae, and A. F. T. Martins, “Adaptively sparse transformers,” in *EMNLP*, 2019. 11, 27
- [80] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees, “Overview of the trec 2019 deep learning track,” *arXiv preprint arXiv:2003.07820*, 2020. 10, 23, 25, 26, 29, 241
- [81] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010, vol. 520. 148
- [82] F. Cunningham, M. R. Amode, D. Barrell, K. Beal, K. Billis, and Others, “Ensembl 2015,” *Nucleic Acids Research*, vol. 43, no. D1, pp. D662–D669, 2015. [Online]. Available: <https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gku1010>
119
- [83] Z. Dai and J. Callan, “Deeper text understanding for ir with contextual neural language modeling,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 985–988. 10, 26, 30, 240, 241, 242
- [84] Z. Dai, C. Xiong, J. Callan, and Z. Liu, “Convolutional neural networks for soft-matching n-grams in ad-hoc search,” in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, p. 126–134. 10, 30, 31, 240, 242
- [85] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” in *ACL*, 2019. 10
- [86] V. Dang and B. W. Croft, “Query reformulation using anchor text,” in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 41–50. 20, 187

- [87] C. A. Davis Jr, G. L. Pappa, D. R. R. de Oliveira, and F. de L. Arcanjo, “Inferring the location of twitter messages based on user relationships,” *Transactions in GIS*, 2011. 176
- [88] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury, “Learning to attend, copy, and generate for session-based query suggestion,” in *CIKM '17*. ACM, 2017, pp. 1747–1756. 20, 188, 190, 191, 198, 199, 200, 201, 203
- [89] H. Deng, J. Han, B. Zhao, Y. Yu, and C. X. Lin, “Probabilistic topic models with biased propagation on heterogeneous information networks,” in *KDD*. ACM, 2011, pp. 1271–1279. 69
- [90] S. Deorowicz, A. Debudaj-Grabysz, and S. Grabowski, “Disk-based k-mer counting on a PC.” *BMC bioinformatics*, vol. 14, no. 160, p. 160, 2013. [Online]. Available: <http://www.biomedcentral.com/1471-2105/14/160><http://www.ncbi.nlm.nih.gov/pubmed/23679007>{%}5Cn<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3680041>{%}5Cn<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3680041>{&}tool=pmcentrez{&}rendertyp 17, 109
- [91] S. Deorowicz, M. Kokot, S. Grabowski, and A. Debudaj-Grabysz, “KMC 2: Fast and resource-frugal k-mer counting,” *Bioinformatics*, vol. 31, no. 10, pp. 1569–1576, 2014. [Online]. Available: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv022> 17, 109
- [92] T. Derrien, R. Johnson, G. Bussotti, A. Tanzer, S. Djebali, H. Tilgner, G. Guernec, D. Martin, A. Merkel, D. G. Knowles *et al.*, “The gencode v7 catalog of human long noncoding rnas: analysis of their gene structure, evolution, and expression,” *Genome research*, vol. 22, no. 9, pp. 1775–1789, 2012. 40
- [93] M. Deshpande and G. Karypis, “Item-based top-n recommendation algorithms,” *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004. 15

- [94] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018. [153](#), [158](#), [166](#), [167](#), [169](#)
- [95] —, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186. [10](#), [22](#), [26](#)
- [96] Q. Ding, H.-F. Yu, and C.-J. Hsieh, “A fast sampling algorithm for maximum inner product search,” in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 3004–3012. [16](#)
- [97] T. H. Do, D. M. Nguyen, E. Tsiligianni, B. Cornelis, and N. Deligiannis, “Multiview deep learning for predicting twitter users’ location,” *arXiv preprint arXiv:1712.08091*, 2017. [176](#), [182](#), [183](#)
- [98] Y. Dong, N. V. Chawla, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144. [13](#), [14](#), [63](#), [64](#), [65](#), [66](#), [70](#), [71](#), [75](#), [80](#)
- [99] M. Dredze, “How social media will change public health,” *IEEE Intelligent Systems*, vol. 27, no. 4, pp. 81–84, 2012. [228](#)
- [100] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, “The yahoo! music dataset and kdd-cup’11,” in *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18*. JMLR. org, 2011, pp. 3–18. [86](#)
- [101] W. Du, P. Poupart, and W. Xu, “Discovering conversational dependencies between messages in dialogs.” in *AAAI’17*, 2017, pp. 4917–4918. [21](#), [209](#)
- [102] L. Duan, C. Aggarwal, S. Ma, R. Hu, and J. Huai, “Scaling up link prediction with

- ensembles,” in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 367–376. [15](#), [86](#), [87](#), [99](#), [100](#)
- [103] U. Dube, J. L. Del-Aguila, Z. Li, J. P. Budde, S. Jiang, S. Hsu, L. Ibanez, M. V. Fernandez, F. Farias, J. Norton *et al.*, “An atlas of cortical circular rna expression in alzheimer disease brains demonstrates clinical and pathological associations,” *Nature neuroscience*, vol. 22, no. 11, pp. 1903–1912, 2019. [40](#)
- [104] R. A. Dubin, M. A. Kazmi, and H. Ostrer, “Inverted repeats are necessary for circularization of the mouse testis sry transcript,” *Gene*, vol. 167, no. 1-2, pp. 245–248, 1995. [41](#)
- [105] J. Eisenstein, B. O’Connor, N. A. Smith, and E. P. Xing, “A latent variable model for geographic lexical variation,” in *EMNLP*, 2010. [181](#)
- [106] M. Elsner and E. Charniak, “You talking to me? a corpus and algorithm for conversation disentanglement.” in *ACL’08*. ACL, 2008, pp. 834–842. [21](#), [209](#), [211](#), [218](#), [221](#), [222](#), [224](#)
- [107] —, “Disentangling chat,” *Computational Linguistics*, vol. 36, no. 3, pp. 389–409, 2010. [209](#), [211](#), [217](#)
- [108] —, “Disentangling chat with local coherence models,” in *ACL-HLT’11*. ACL, 2011, pp. 1179–1189. [209](#), [211](#), [217](#), [224](#)
- [109] F. Errica, M. Podda, D. Bacciu, and A. Micheli, “A fair comparison of graph neural networks for graph classification,” *arXiv preprint arXiv:1912.09893*, 2019. [76](#)
- [110] N. J. Evans, J. Phua, J. Lim, and H. Jun, “Disclosing instagram influencer advertising: The effects of disclosure language on advertising recognition, attitudes, and behavioral intent,” *Journal of Interactive Advertising*, vol. 17, no. 2, pp. 138–149, 2017. [19](#), [152](#), [164](#)

- [111] N. J. Evans, M. G. Hoy, and C. C. Childers, “Parenting “youtube natives”: The impact of pre-roll advertising and text disclosures on parental responses to sponsored child influencer videos,” *Journal of Advertising*, vol. 47, no. 4, pp. 326–346, 2018. 19
- [112] B. S. Everitt, *The Cambridge dictionary of statistics*. Cambridge University Press, 2006. 138
- [113] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *arXiv preprint arXiv:2101.03961*, 2021. 4
- [114] A. Fiannaca, M. La Rosa, L. La Paglia, R. Rizzo, and A. Urso, “nrc: non-coding rna classifier based on structural features,” *BioData mining*, vol. 10, no. 1, p. 27, 2017. 12, 41, 52
- [115] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani, “Using association rules to discover search engines related queries,” in *Web Congress '03*. IEEE, 2003, pp. 66–71. 20
- [116] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani, “Concept-based interactive query expansion,” in *CIKM '05*. ACM, 2005, pp. 696–703. 20
- [117] T. O. for Economic Co-operation and Development, “Good practice guide on on-line advertising,” [http://www.oecd.org/officialdocuments/publicdisplaydocumentpdf/?cote=DSTI/CP\(2018\)16/FINAL&docLanguage=En](http://www.oecd.org/officialdocuments/publicdisplaydocumentpdf/?cote=DSTI/CP(2018)16/FINAL&docLanguage=En), 2019. 151, 153
- [118] A. Frankish, M. Diekhans, A.-M. Ferreira, R. Johnson, I. Jungreis, J. Loveland, J. M. Mudge, C. Sisu, J. Wright, J. Armstrong *et al.*, “Gencode reference annotation for the human and mouse genomes,” *Nucleic acids research*, vol. 47, no. D1, pp. D766–D773, 2019. 52
- [119] A. C. Frazee, A. E. Jaffe, B. Langmead, and J. T. Leek, “Polyester: simulating RNA-seq datasets with differential transcript expression.” *Bioinformatics*, vol. 31, no. 17, pp. 2778–84, 2015. [Online]. Available: <http://biorxiv.org/content/early/>

2014/12/12/006015.abstract<http://www.ncbi.nlm.nih.gov/pubmed/25926345><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4635655> 119

- [120] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *science*, vol. 315, no. 5814, pp. 972–976, 2007. 131, 133, 139, 143, 144, 145, 224
- [121] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001. 166, 167
- [122] T.-y. Fu, W.-C. Lee, and Z. Lei, “Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning,” in *CIKM*. ACM, 2017, pp. 1797–1806. 13, 15, 64, 70, 75, 80
- [123] Y. Gao, J. Wang, and F. Zhao, “Ciri: an efficient and unbiased algorithm for de novo circular rna identification,” *Genome biology*, vol. 16, no. 1, p. 4, 2015. 11, 53
- [124] Y. Gao, J. Zhang, and F. Zhao, “Circular rna identification based on multiple seed matching,” *Briefings in bioinformatics*, vol. 19, no. 5, pp. 803–810, 2018. 11
- [125] D. D. Genc, C. Yesilyurt, and G. Tuncel, “Air pollution forecasting in ankara, turkey using air pollution index and its relation to assimilative capacity of the atmosphere,” *Environmental monitoring and assessment*, vol. 166, no. 1-4, pp. 11–27, 2010. 227
- [126] C. Gentile, S. Li, and G. Zappella, “Online clustering of bandits,” in *International Conference on Machine Learning*, 2014, pp. 757–765. 100
- [127] P. Glažar, P. Papavasileiou, and N. Rajewsky, “cirbase: a database for circular rnas,” *Rna*, vol. 20, no. 11, pp. 1666–1670, 2014. 52
- [128] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *AISTATS*, 2011, pp. 315–323. 50, 53
- [129] G. H. Gonnet and R. A. Baeza-Yates, “An analysis of the karp-rabin string matching algorithm,” *Information Processing Letters*, vol. 34, no. 5, pp. 271–274, 1990. 117

- [130] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *NIPS*, 2014, pp. 2672–2680. [71](#)
- [131] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. [72](#), [218](#), [232](#)
- [132] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. M. Blei, “Bayesian nonparametric poisson factorization for recommendation systems.” in *AISTATS*, 2014, pp. 275–283. [138](#)
- [133] P. K. Gopalan, L. Charlin, and D. Blei, “Content-based recommendations with poisson factorization,” in *NIPS*, 2014, pp. 3176–3184. [138](#)
- [134] E. Grave, A. Joulin, M. Cissé, D. Grangier, and H. Jégou, “Efficient softmax approximation for gpus,” in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1302–1310. [91](#)
- [135] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16, 2016, pp. 855–864. [13](#), [62](#), [63](#), [64](#), [65](#), [71](#), [75](#), [80](#), [135](#), [136](#), [144](#), [193](#)
- [136] J. Guo, X. Cheng, G. Xu, and X. Zhu, “Intent-aware query similarity,” in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 259–268. [20](#)
- [137] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, “A deep relevance matching model for ad-hoc retrieval,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016. [10](#)
- [138] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and

- X. Cheng, “A deep look into neural ranking models for information retrieval,” 2019. 10
- [139] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NIPS*, 2017. 76
- [140] W. A. Hamilton, O. Garretson, and A. Kerne, “Streaming on twitch: fostering participatory communities of play within live mixed media,” in *CHI’14*. ACM, 2014, pp. 1315–1324. 208
- [141] B. Han, P. Cook, and T. Baldwin, “Geolocation prediction in social media data by finding location indicative words,” in *COLING*, 2012. 176, 181
- [142] J. Han and C. Moraga, “The influence of the sigmoid function parameters on the speed of backpropagation learning,” in *ANNIIP*. Springer, 1995, pp. 195–201. 50, 195, 217
- [143] S. Han, Y. Liang, Q. Ma, Y. Xu, Y. Zhang, W. Du, C. Wang, and Y. Li, “Lncfinder: an integrated platform for long non-coding rna identification utilizing sequence intrinsic composition, structural information and physicochemical property,” *Briefings in bioinformatics*, vol. 20, no. 6, pp. 2009–2027, 2019. 12
- [144] T. B. Hansen, T. I. Jensen, B. H. Clausen, J. B. Bramsen, B. Finsen, C. K. Damgaard, and J. Kjems, “Natural rna circles function as efficient microRNA sponges,” *Nature*, vol. 495, no. 7441, pp. 384–388, 2013. 41
- [145] T. B. Hansen, M. T. Venø, C. K. Damgaard, and J. Kjems, “Comparison of circular rna prediction tools,” *Nucleic acids research*, vol. 44, no. 6, pp. e58–e58, 2016. 11
- [146] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, “An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge,” in *ACL*, 2017, pp. 221–231. 48
- [147] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *Acm*

- transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, p. 19, 2016. 74, 75, 99
- [148] Q. He, D. Jiang, Z. Liao, S. C. Hoi, K. Chang, E.-P. Lim, and H. Li, “Web query recommendation via sequential query prediction,” in *Data Engineering, 2009. ICDE’09. IEEE 25th International Conference on*. IEEE, 2009, pp. 1443–1454. 20, 187, 199, 201
- [149] B. Hidasi and D. Tikk, “General factorization framework for context-aware recommendations,” *Data Mining and Knowledge Discovery (DMKD)*, vol. 30, no. 2, pp. 342–371, Mar. 2016. 148
- [150] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” 2016. 130
- [151] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006. 50, 197, 210
- [152] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, 1997. 46, 193
- [153] S. Hofstätter, M. Zlabinger, and A. Hanbury, “Tu wien@ trec deep learning’19—simple contextualization for re-ranking,” *arXiv preprint arXiv:1912.01385*, 2019. 241
- [154] —, “Interpretable & time-budget-constrained contextualization for re-ranking,” *arXiv preprint arXiv:2002.01854*, 2020. 30, 242
- [155] S. Hofstätter, H. Zamani, B. Mitra, N. Craswell, and A. Hanbury, “Local self-attention over long text for efficient document retrieval,” in *SIGIR*, 2020. 10, 30, 242
- [156] J. Holt and L. McMillan, “Merging of multi-string BWTs with applications.” *Bioinformatics*, vol. 30, no. 24, pp. 3524–31, 2014. [Online]. Available: <http://bioinformatics.oxfordjournals.org/content/30/24/3524.full> 18, 109

- [157] B. Hooi, N. Shah, A. Beutel, S. Günnemann, L. Akoglu, M. Kumar, D. Makhija, and C. Faloutsos, “Birdnest: Bayesian inference for ratings-fraud detection,” in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 495–503. [19](#)
- [158] B. Hu, Y. Fang, and C. Shi, “Adversarial learning on heterogeneous information networks,” in *KDD*. ACM, 2019. [76](#), [80](#)
- [159] B. Huang and K. M. Carley, “A hierarchical location prediction neural network for twitter user geolocation,” in *EMNLP*, 2019. [182](#), [183](#)
- [160] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang, “Relevant term suggestion in interactive web search based on contextual information in query session logs,” *Journal of the Association for Information Science and Technology*, vol. 54, no. 7, pp. 638–649, 2003. [20](#), [187](#)
- [161] J. Huang and E. N. Efthimiadis, “Analyzing and evaluating query reformulation strategies in web search logs,” in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 77–86. [20](#), [21](#), [187](#)
- [162] K. Hui, A. Yates, K. Berberich, and G. de Melo, “Pacrr: A position-aware neural ir model for relevance matching,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1049–1058. [10](#)
- [163] K. Hui, A. Yates, K. Berberich, and G. De Melo, “Co-pacrr: A context-aware neural ir model for ad-hoc retrieval,” in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 279–287. [30](#), [242](#)
- [164] R. Hussein, D. Yang, and P. Cudré-Mauroux, “Are meta-paths necessary?: Revisiting heterogeneous graph embeddings,” in *CIKM*. ACM, 2018, pp. 437–446. [13](#), [15](#), [63](#), [64](#), [65](#), [70](#), [71](#), [75](#), [80](#)
- [165] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International journal of forecasting*, vol. 22, no. 4, pp. 679–688, 2006. [143](#)

- [166] Y. Ibrahim, “Instagramming life: Banal imaging and the poetics of the everyday,” *Journal of Media Practice*, vol. 16, no. 1, pp. 42–54, 2015. [4](#)
- [167] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613. [16](#), [86](#)
- [168] A. Ivanov, S. Memczak, E. Wyler, F. Torti, H. T. Porath, M. R. Orejuela, M. Piechotta, E. Y. Levanon, M. Landthaler, C. Dieterich *et al.*, “Analysis of intron sequences reveals hallmarks of circular rna biogenesis in animals,” *Cell reports*, vol. 10, no. 2, pp. 170–177, 2015. [41](#)
- [169] W. R. Jeck and N. E. Sharpless, “Detecting and characterizing circular rnas,” *Nature biotechnology*, vol. 32, no. 5, p. 453, 2014. [59](#)
- [170] W. R. Jeck, J. A. Sorrentino, K. Wang, M. K. Slevin, C. E. Burd, J. Liu, W. F. Marzluff, and N. E. Sharpless, “Circular rnas are abundant, conserved, and associated with alu repeats,” *Rna*, vol. 19, no. 2, pp. 141–157, 2013. [41](#), [43](#), [56](#)
- [171] D. Jiang, K. W.-T. Leung, and W. Ng, “Context-aware search personalization with concept preference,” in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 563–572. [21](#), [187](#)
- [172] J.-Y. Jiang and P.-J. Cheng, “Classifying user search intents for query auto-completion,” in *ICTIR ’16*. ACM, 2016, pp. 49–58. [20](#), [187](#), [191](#), [192](#), [198](#)
- [173] J.-Y. Jiang and W. Wang, “Rin: Reformulation inference network for context-aware query suggestion,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2018, pp. 197–206. [6](#)
- [174] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng, “Learning user reformulation behavior for query auto-completion,” in *SIGIR ’14*. ACM, 2014, pp. 445–454. [20](#), [21](#), [187](#), [188](#), [191](#), [198](#), [200](#), [201](#)

- [175] J.-Y. Jiang, J. Liu, C.-Y. Lin, and P.-J. Cheng, “Improving ranking consistency for web search by leveraging a knowledge base and search logs,” in *CIKM '15*. ACM, 2015, pp. 1441–1450. [192](#)
- [176] J.-Y. Jiang, P.-J. Cheng, and W. Wang, “Open source repository recommendation in social coding,” in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017. [163](#)
- [177] J.-Y. Jiang, C.-T. Li, Y. Chen, and W. Wang, “Identifying users behind shared accounts in online streaming services,” in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 65–74. [5](#)
- [178] J.-Y. Jiang, M. Zhang, C. Li, M. Bendersky, N. Golbandi, and M. Najork, “Semantic text matching for long-form documents,” in *The World Wide Web Conference*, 2019, pp. 795–806. [22](#)
- [179] W. Jiang, Y. Wang, M.-H. Tsou, and X. Fu, “Using social media to detect outdoor air pollution and monitor air quality index (aqi): a geo-targeted spatiotemporal analysis framework with sina weibo (chinese twitter),” *PloS one*, vol. 10, no. 10, p. e0141185, 2015. [228](#), [233](#)
- [180] Y. Jiang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang, “Maqs: a personalized mobile sensing system for indoor air quality monitoring,” in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 271–280. [228](#)
- [181] N. Jindal, B. Liu, and E.-P. Lim, “Finding unusual review patterns using unexpected rules,” in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 1549–1552. [19](#)
- [182] R. Jones, B. Rey, O. Madani, and W. Greiner, “Generating query substitutions,” in *WWW '06*. ACM, 2006, pp. 387–396. [187](#)

- [183] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *ICML '15*, 2015, pp. 2342–2350. 46, 193
- [184] C. J.-T. Ju, R. Li, Z. Wu, J.-Y. Jiang, Z. Yang *et al.*, “Fleximer: Accurate Quantification of RNA-Seq via Variable-Length k-mers,” in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics - ACM-BCB '17*. New York, New York, USA: ACM Press, 2017, pp. 263–272. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3107411.3107444> 108, 111, 127
- [185] C. J.-T. Ju, J.-Y. Jiang, R. Li, Z. Li, and W. Wang, “Tahcoroll: An efficient approach for signature profiling in genomic data through variable-length k-mers,” *bioRxiv*, 2017. 46
- [186] D. Jurgens, “That’s what friends are for: Inferring location in online social media platforms based on social relationships,” in *ICWSM*, 2013. 176
- [187] Z. Kang, C. Peng, and Q. Cheng, “Top-n recommender system via matrix completion,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 15
- [188] R. M. Karp and M. O. Rabin, “Efficient randomized pattern-matching algorithms,” *IBM Journal of Research and Development*, vol. 31, no. 2, pp. 249–260, 1987. [Online]. Available: <https://pdfs.semanticscholar.org/c47d/151f09c567013761632c89e237431c6291a2.pdf><http://dl.acm.org/citation.cfm?id=1012156.1012171> 116
- [189] O. Kaššák, M. Kompan, and M. Bieliková, “Personalized hybrid recommendation for group of users: Top-n multimedia recommender,” *Information Processing & Management*, vol. 52, no. 3, pp. 459–477, 2016. 15
- [190] L. Kelly, G. Kerr, and J. Drennan, “Avoidance of advertising in social networking sites: The teenage perspective,” *Journal of interactive advertising*, vol. 10, no. 2, pp. 16–27, 2010. 161

- [191] S. Kim, J. Han, S. Yoo, and M. Gerla, “How are social influencers connected in instagram?” in *International Conference on Social Informatics (SocInfo)*. Springer, 2017. 161
- [192] S. Kim, J.-Y. Jiang, M. Nakada, J. Han, and W. Wang, “Multimodal post attentive profiling for influencer marketing,” in *Proceedings of The Web Conference 2020*, 2020, pp. 2878–2884. 19, 161, 163
- [193] S. Kim, J.-Y. Jiang, and W. Wang, “Discovering undisclosed paid partnership on social media via aspect-attentive sponsored post learning,” in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 319–327. 5
- [194] Y. Kim, “Convolutional neural networks for sentence classification,” in *EMNLP’14*. ACL, 2014. 210, 214, 215, 231
- [195] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR’16*, 2015. 218
- [196] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014. 31, 199, 233, 240
- [197] S. Kinsella, V. Murdock, and N. O’Hare, “‘i’m eating a sandwich in glasgow’ modeling locations with tweets,” in *SMUC*, 2011. 175
- [198] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016. 153, 156, 166, 167, 169
- [199] —, “Variational graph auto-encoders,” *arXiv preprint arXiv:1611.07308*, 2016. 76
- [200] —, “Semi-supervised classification with graph convolutional networks,” in *ICLR*, 2017. 182
- [201] N. Kitaev, Łukasz Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” *arXiv preprint arXiv:2001.04451*, 2020. 11

- [202] M. Kokot, M. Długosz, and S. Deorowicz, “KMC 3: counting and manipulating k-mer statistics,” *Bioinformatics*, vol. 3, no. May, pp. 1–3, 2017. [Online]. Available: <http://arxiv.org/abs/1701.08022><https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btx304> 17, 109
- [203] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, no. 8, pp. 30–37, 2009. 15, 86
- [204] J. Kunegis, “Konect: the koblenz network collection,” in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 1343–1350. 99
- [205] S. Kurtz, A. Narechania, J. C. Stein, and D. Ware, “A new method to compute K-mer frequencies and its application to annotate large repetitive plant genomes.” *BMC genomics*, vol. 9, no. 9, p. 517, 2008. [Online]. Available: <http://www.zbh.uni-hamburg.de/Tallymer>.<http://www.biomedcentral.com/1471-2164/9/517> 18, 109
- [206] S. Lai, L. Xu, K. Liu, and J. Zhao, “Recurrent convolutional neural networks for text classification.” in *AAAI’15*, vol. 333, 2015, pp. 2267–2273. 210
- [207] N. Landwehr, “Modeling interleaved hidden processes,” in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 520–527. 19
- [208] E. Lasda and R. Parker, “Circular rnas: diversity of form and function,” *Rna*, vol. 20, no. 12, pp. 1829–1842, 2014. 41
- [209] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *ICML ’14*, 2014, pp. 1188–1196. 192, 210, 224
- [210] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 41, 217
- [211] C.-J. Lee, R.-C. Chen, S.-H. Kao, and P.-J. Cheng, “A term dependency-based approach for query terms ranking,” in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1267–1276. 21

- [212] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, “Stacked cross attention for image-text matching,” in *ECCV*, 2018, pp. 201–216. 48
- [213] D. Lemire and O. Kaser, “Recursive n-gram hashing is pairwise independent, at best,” *Computer Speech & Language*, vol. 24, no. 4, pp. 698–710, 2010. 117
- [214] S. Li, A. Karatzoglou, and C. Gentile, “Collaborative filtering bandits,” in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 539–548. 100
- [215] S. Li, W. Chen, S. Li, and K.-S. Leung, “Improved algorithm on online clustering of bandits,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 2923–2929. 100
- [216] Y. Li and X. Yan, “MSPKmerCounter: A Fast and Memory Efficient Approach for K-mer Counting,” *arXiv preprint*, vol. 1505.06550, pp. 1–7, 2015. [Online]. Available: <http://arxiv.org/abs/1505.06550> 17, 109
- [217] Z. Li, C. Huang, C. Bao, L. Chen, M. Lin, X. Wang, G. Zhong, B. Yu, W. Hu, L. Dai *et al.*, “Exon-intron circular rnas regulate transcription in the nucleus,” *Nature structural & molecular biology*, vol. 22, no. 3, p. 256, 2015. 41
- [218] Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao, and H. Li, “Mining concept sequences from large-scale search logs for context-aware query suggestion,” *TIST*, vol. 3, no. 1, p. 17, 2011. 20, 192
- [219] D. Liben-Nowell and J. Kleinberg, “The link-prediction problem for social networks,” *JASIST*, vol. 58, no. 7, pp. 1019–1031, 2007. 62
- [220] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *IEEE Internet computing*, no. 1, pp. 76–80, 2003. 85
- [221] R. Liu, T. Wu, and B. Mozafari, “A bandit approach to maximum inner product search,” *CoRR*, vol. abs/1812.06360, 2019. 16

- [222] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019. 240
- [223] C. Lou and S. Yuan, “Influencer marketing: how message value and credibility affect consumer trust of branded content on social media,” *Journal of Interactive Advertising*, vol. 19, no. 1, pp. 58–73, 2019. 19, 152
- [224] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *ICML’13*, vol. 30, 2013. 217
- [225] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” *Information Systems*, vol. 45, pp. 61–68, 2014. 16, 96
- [226] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE transactions on pattern analysis and machine intelligence*, 2018. 16, 87, 96, 97
- [227] A. A. Mamun, S. Pal, and S. Rajasekaran, “KCMBT: A k-mer Counter based on Multiple Burst Trees,” *Bioinformatics*, vol. 32, no. 18, pp. 2783–2790, 2016. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/27283950> 18, 109
- [228] G. Marçais and C. Kingsford, “A fast, lock-free approach for efficient parallel counting of occurrences of k-mers,” *Bioinformatics*, vol. 27, no. 6, pp. 764–770, 2011. [Online]. Available: <http://www.cbcb.umd.edu/> 17, 108
- [229] MarketingHub, “The state of influencer marketing 2019 : Benchmark report,” *Benchmark Report*, 2019. 151, 154, 163
- [230] E. Mayfield, D. Adamson, and C. P. Rosé, “Hierarchical conversation structure prediction in multi-party chat,” in *SIGDIAL’12*. ACL, 2012, pp. 60–69. 21, 209, 211, 213

- [231] M. Mazloom, R. Rietveld, S. Rudinac, M. Worring, and W. Van Dolen, “Multimodal popularity prediction of brand-related social media posts,” in *Proceedings of the 24th ACM international conference on Multimedia (MM)*. ACM, 2016, pp. 197–201. 158
- [232] M. L. McHugh, “Interrater reliability: the kappa statistic,” *Biochemia medica: Biochemia medica*, vol. 22, no. 3, pp. 276–282, 2012. 169
- [233] S. Mehri and G. Carenini, “Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks,” in *IJCNLP’17*, vol. 1, 2017, pp. 615–623. 21, 210, 222
- [234] Q. Mei, D. Zhou, and K. Church, “Query suggestion using hitting time,” in *CIKM ’08*. ACM, 2008, pp. 469–478. 20, 187, 192
- [235] S. Mei, H. Li, J. Fan, X. Zhu, and C. R. Dyer, “Inferring air pollution by sniffing social media,” in *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. IEEE, 2014, pp. 534–539. 230
- [236] P. Melsted and J. K. Pritchard, “Efficient counting of k-mers in DNA sequences using a bloom filter,” *BMC Bioinformatics*, vol. 12, no. 333, 2011. [Online]. Available: <http://pritch.bsd.uchicago.edu/bfcounter.html> 17, 108
- [237] S. Memczak, M. Jens, A. Elefsinioti, F. Torti, J. Krueger, A. Rybak, L. Maier, S. D. Mackowiak, L. H. Gregersen, M. Munschauer *et al.*, “Circular rnas are a large class of animal rnas with regulatory potency,” *Nature*, vol. 495, no. 7441, pp. 333–338, 2013. 11, 41
- [238] D. Metzler and W. B. Croft, “A markov random field model for term dependencies,” in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 472–479. 22, 29, 242
- [239] —, “Linear feature-based models for information retrieval,” *Information Retrieval*, vol. 10, no. 3, pp. 257–274, 2007. 240, 242

- [240] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *Proceedings of Workshop at International Conference on Learning Representations*, 2013. 131, 135
- [241] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119. 13, 63, 64, 71, 75, 136, 144, 145, 192, 210
- [242] X. Min, W. Zeng, N. Chen, T. Chen, and R. Jiang, “Chromatin accessibility prediction via convolutional long short-term memory networks with k-mer embedding,” *Bioinformatics*, vol. 33, no. 14, pp. i92–i101, 2017. 46
- [243] B. Mitra, “Exploring session context using distributed representations of queries and reformulations,” in *SIGIR ’15*. ACM, 2015, pp. 3–12. 205
- [244] S. Moro, P. Rita, and B. Vala, “Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach,” *Journal of Business Research*, vol. 69, no. 9, 2016. 161, 163
- [245] J. Mueller and A. Thyagarajan, “Siamese recurrent architectures for learning sentence similarity.” in *AAAI’16*, 2016, pp. 2786–2792. 21, 210, 217
- [246] M. Naehrig, K. Lauter, and V. Vaikuntanathan, “Can homomorphic encryption be practical?” in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 113–124. 191
- [247] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *ICML ’10*, 2010, pp. 807–814. 194
- [248] G. Navarro and M. Raffinot, *Flexible pattern matching*. Cambridge University Press, 2008. 109, 112

- [249] B. Neyshabur and N. Srebro, “On symmetric and asymmetric lshs for inner product search,” in *ICML*, 2015. 16, 86, 100
- [250] R. Nogueira and K. Cho, “Passage re-ranking with bert,” *arXiv preprint arXiv:1901.04085*, 2019. 10, 22, 24
- [251] R. Nogueira, W. Yang, K. Cho, and J. Lin, “Multi-stage document ranking with bert,” *arXiv preprint arXiv:1910.14424*, 2019. 10, 22, 26
- [252] E. Ntoutsi, K. Stefanidis, K. Nørnvåg, and H.-P. Kriegel, “Fast group recommendations by applying user clustering,” in *International Conference on Conceptual Modeling*. Springer, 2012, pp. 126–140. 15
- [253] Y. Ohshima and Y. Gotoh, “Signals for the selection of a splice site in pre-mrna: computer analysis of splice junction sequences and like sequences,” *J. Mol. Biol.*, vol. 195, no. 2, pp. 247–259, 1987. 59
- [254] T. Opsahl and P. Panzarasa, “Clustering in weighted networks,” *Social networks*, vol. 31, no. 2, pp. 155–163, 2009. 62
- [255] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *KDD*. ACM, 2016, pp. 1105–1114. 13
- [256] M. Ouedraogo, C. Bettembourg, A. Breteau, O. Sallou, C. Diot, O. Demeure, and F. Lecerf, “The duplicated genes database: identification and functional annotation of co-localised duplicated genes across genomes,” *PloS one*, vol. 7, no. 11, p. e50653, 2012. 52
- [257] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu, “Learning to suggest: a machine learning framework for ranking query suggestions,” in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 25–34. 20

- [258] X. Pan and K. Xiong, “Predcircrna: computational classification of circular rna from other long non-coding rna using hybrid features,” *Mol. Omics*, vol. 11, no. 8, pp. 2219–2226, 2015. 11, 41, 51, 52
- [259] P. Pandey, M. A. Bender, R. Johnson, and R. Patro, “Squeakr: an exact and approximate k-mer counting system,” *Bioinformatics*, p. btx636, 2017. [Online]. Available: [+http://dx.doi.org/10.1093/bioinformatics/btx636](http://dx.doi.org/10.1093/bioinformatics/btx636) 17, 108
- [260] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035. 31
- [261] R. Patro, S. M. Mount, and C. Kingsford, “Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms.” *Nature biotechnology*, vol. 32, no. 5, pp. 462–4, 2014. [Online]. Available: <http://dx.doi.org/10.1038/nbt.2862> 107
- [262] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *JMLR*, vol. 12, no. Oct, pp. 2825–2830, 2011. 54
- [263] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.” in *Proceedings of Empirical Methods in Natural Language Processing*, ser. EMNLP ’14, vol. 14, 2014, pp. 1532–1543. 131, 135, 182, 192
- [264] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD ’14, 2014, pp. 701–710. xxv, 13, 62, 63, 64, 71, 75, 80, 136, 144, 145
- [265] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 2227–2237. 4

- [266] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proc. of NAACL*, 2018. 166, 167
- [267] C. P. Ponting, P. L. Oliver, and W. Reik, “Evolution and functions of long noncoding rnas,” *Cell*, vol. 136, no. 4, pp. 629–641, 2009. 40
- [268] S. Poria, E. Cambria, and A. Gelbukh, “Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis,” in *EMNLP’15. ACL*, 2015, pp. 2539–2544. 210
- [269] D. M. Powers, “Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation,” 2011. 76, 143
- [270] S. Qu, Z. Liu, X. Yang, J. Zhou, H. Yu, R. Zhang, and H. Li, “The emerging functions and roles of circular rnas in cancer,” *Cancer letters*, vol. 414, pp. 301–309, 2018. 40
- [271] A. Rahimi, T. Cohn, and T. Baldwin, “A neural model for user geolocation and lexical dialectology,” in *ACL*, 2017. 175, 182, 183
- [272] —, “Semi-supervised user geolocation via graph convolutional networks,” *arXiv preprint arXiv:1804.08049*, 2018. 176, 182, 183
- [273] A. Rahman, I. Hallgrímsdóttir, M. B. Eisen, and L. Pachter, “Association Mapping From Sequencing Reads Using K-mers,” *bioRxiv*, pp. 1–14, 2017. [Online]. Available: <http://dx.doi.org/10.1101/141267><http://biorxiv.org/content/early/2017/05/23/141267?rss=1> 107, 108
- [274] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A lock-free approach to parallelizing stochastic gradient descent,” in *NIPS*, 2011, pp. 693–701. 72

- [275] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” in *ICLR*, 2018. 53
- [276] G. Ren, X. Ni, M. Malik, and Q. Ke, “Conversational query understanding using sequence to sequence modeling,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 1715–1724. 21
- [277] Z. Ren, J. Ma, S. Wang, and Y. Liu, “Summarizing web forum threads based on a latent topic propagation process,” in *CIKM’11*. ACM, 2011, pp. 879–884. 221
- [278] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461. 15, 148, 149
- [279] G. Rizk, D. Lavenier, and R. Chikhi, “DSK: K-mer counting with very low memory usage,” *Bioinformatics*, vol. 29, no. 5, pp. 652–653, 2013. [Online]. Available: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt020> 109
- [280] S. Roller, M. Speriosu, S. Rallapalli, B. Wing, and J. Baldridge, “Supervised text-based geolocation using language models on an adaptive grid,” in *EMNLP*, 2012. 181
- [281] D. R.-J. G.-J. Rydning, “The digitization of the world from edge to core,” *Framingham: International Data Corporation*, 2018. 4
- [282] A. Sacan and I. H. Toroslu, “Approximate similarity search in genomic sequence databases using landmark-guided embedding,” in *SISAP*. IEEE, 2008, pp. 43–50. 60
- [283] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860. 228

- [284] —, “Tweet analysis for real-time event detection and earthquake reporting system development,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 919–931, 2013. 228
- [285] S. L. Salzberg, M. Pertea, J. A. Fahrner, and N. Sobreira, “DI-AMUND: direct comparison of genomes to detect mutations.” *Human mutation*, vol. 35, no. 3, pp. 283–8, 2014. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/24375697><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4031744><http://doi.wiley.com/10.1002/humu.22503> 107, 108
- [286] M. Santillana, A. T. Nguyen, M. Dredze, M. J. Paul, E. O. Nsoesie, and J. S. Brownstein, “Combining search, social media, and traditional data sources to improve influenza surveillance,” *PLoS computational biology*, vol. 11, no. 10, p. e1004513, 2015. 228
- [287] R. L. Santos, C. Macdonald, and I. Ounis, “Learning to rank query suggestions for adhoc and diversity search,” *Information Retrieval*, vol. 16, no. 4, pp. 429–451, 2013. 20
- [288] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 39. 230
- [289] A. Severyn and A. Moschitti, “Learning to rank short text pairs with convolutional deep neural networks,” in *SIGIR’15*. ACM, 2015, pp. 373–382. 21, 210, 214, 216, 217, 222, 223
- [290] D. Shen, Q. Yang, J.-T. Sun, and Z. Chen, “Thread detection in dynamic text message streams,” in *SIGIR’06*. ACM, 2006, pp. 35–42. 21, 209
- [291] C. Shi, B. Hu, X. Zhao, and P. Yu, “Heterogeneous information network embedding for recommendation,” *TKDE*, 2018. 64

- [292] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, “Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering,” in *RecSys*. ACM, 2012, pp. 139–146. [149](#)
- [293] K. Shim, M. Lee, I. Choi, Y. Boo, and W. Sung, “Svd-softmax: Fast softmax approximation on large vocabulary neural networks,” in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5463–5473. [91](#), [100](#)
- [294] M. Shokouhi, “Learning to personalize query auto-completion,” in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 103–112. [20](#), [187](#), [198](#), [199](#), [201](#)
- [295] A. Shrivastava and P. Li, “Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips),” in *Advances in Neural Information Processing Systems*, 2014, pp. 2321–2329. [16](#), [86](#), [100](#)
- [296] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, “Zero-shot learning through cross-modal transfer,” in *NIPS*, 2013, pp. 935–943. [51](#)
- [297] Y. Song, D. Zhou, and L.-w. He, “Query suggestion by constructing term-transition graphs,” in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 353–362. [187](#)
- [298] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie, “A hierarchical recurrent encoder-decoder for generative context-aware query suggestion,” in *CIKM '15*. ACM, 2015, pp. 553–562. [20](#), [188](#), [190](#), [198](#), [199](#), [200](#), [201](#), [203](#)
- [299] R. F. Sproull, “Refinements to nearest-neighbor searching ink-dimensional trees,” *Algorithmica*, vol. 6, no. 1-6, pp. 579–589, 1991. [16](#), [86](#)
- [300] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014. [218](#)

- [301] A. Strehl and J. Ghosh, “Cluster ensembles—a knowledge reuse framework for combining multiple partitions,” *Journal of machine learning research*, vol. 3, no. Dec, pp. 583–617, 2002. [143](#)
- [302] C. Stubb and J. Colliander, ““this is not sponsored content”—the effects of impartiality disclosure and e-commerce landing pages on consumer responses to social media influencer posts,” *Computers in Human Behavior*, 2019. [19](#), [152](#)
- [303] C. Suenkel, D. Cavalli, S. Massalini, F. Calegari, and N. Rajewsky, “A highly conserved circular rna is required to keep neural cells in a progenitor state in the mammalian brain,” *Cell Reports*, vol. 30, no. 7, pp. 2170–2179, 2020. [43](#), [56](#)
- [304] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, “Adaptive attention span in transformers,” in *ACL*, 2019. [11](#), [27](#)
- [305] Y. Sun and J. Han, “Mining heterogeneous information networks: principles and methodologies,” *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 1–159, 2012. [66](#)
- [306] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” *VLDB*, vol. 4, no. 11, pp. 992–1003, 2011. [13](#), [63](#), [74](#)
- [307] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, “Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks,” *TKDD*, vol. 7, no. 3, p. 11, 2013. [66](#)
- [308] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS ’14*, 2014, pp. 3104–3112. [188](#), [195](#)
- [309] L. Szabo, R. Morey, N. J. Palpant, P. L. Wang, N. Afari, C. Jiang, M. M. Parast, C. E. Murry, L. C. Laurent, and J. Salzman, “Statistically based splicing detection reveals neural enrichment and tissue-specific induction of circular rna during human fetal development,” *Genome biology*, vol. 16, no. 1, p. 126, 2015. [11](#)

- [310] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 2818–2826. [153](#), [158](#)
- [311] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, “Learning sentiment-specific word embedding for twitter sentiment classification.” in *ACL’14*, 2014, pp. 1555–1565. [210](#)
- [312] J. Tang, M. Qu, and Q. Mei, “Pte: Predictive text embedding through large-scale heterogeneous text networks,” in *KDD*. ACM, 2015, pp. 1165–1174. [13](#), [14](#)
- [313] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW ’15, 2015, pp. 1067–1077. [13](#), [14](#), [75](#), [80](#), [144](#), [145](#), [166](#), [167](#)
- [314] J. Tang, S. Chang, C. Aggarwal, and H. Liu, “Negative link prediction in social media,” in *Proceedings of the eighth ACM international conference on web search and data mining*. ACM, 2015, pp. 87–96. [85](#)
- [315] L. Tang and H. Liu, “Relational learning via latent social dimensions,” in *KDD*. ACM, 2009, pp. 817–826. [62](#)
- [316] I. Tenney, D. Das, and E. Pavlick, “Bert rediscovers the classical nlp pipeline,” *arXiv preprint arXiv:1905.05950*, 2019. [24](#), [27](#)
- [317] L. F. Thomas and P. Satrom, “Circular rnas are depleted of polymorphisms at microRNA binding sites,” *Bioinformatics*, vol. 30, no. 16, pp. 2243–2246, 2014. [41](#)
- [318] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf, “Accelerated dp based search for statistical translation,” in *Fifth European Conference on Speech Communication and Technology*, 1997. [199](#)

- [319] B. Twardowski, “Modelling contextual information in session-aware recommender systems with neural networks,” in *Proceedings of the 10th ACM International Conference on Recommender Systems*, ser. RecSys ’16, 2016, pp. 273–276. 130
- [320] C. F. Van Loan, “Matrix computations (johns hopkins studies in mathematical sciences),” 1996. 197
- [321] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 25, 48, 71, 153, 155
- [322] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *ICLR*, 2018. 178
- [323] K. Verstrepen and B. Goethals, “Top-n recommendation for shared accounts,” in *Proceedings of the 9th ACM Conference on Recommender Systems*, ser. RecSys ’15, 2015, pp. 59–66. 18, 130, 141
- [324] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *JMLR*, vol. 11, no. Oct, pp. 2837–2854, 2010. 78
- [325] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *KDD*, 2016. 13, 14
- [326] D. Wang, T. Zeng, Z. Lin, L. Yan, F. Wang, L. Tang, L. Wang, D. Tang, P. Chen, and M. Yang, “Long non-coding rna snhg5 regulates chemotherapy resistance through the mir-32/dnajb9 axis in acute myeloid leukemia,” *Biomed. Pha.*, vol. 123, p. 109802, 2020. 40
- [327] F. Wang, C.-T. Lu, Y. Qu, and S. Y. Philip, “Collective geographical embedding for geolocating social network users,” in *PAKDD*, 2017. 176

- [328] H. Wang, C. Wang, C. Zhai, and J. Han, “Learning online discussion structures by conditional random fields,” in *SIGIR’11*. ACM, 2011, pp. 435–444. 21
- [329] J. Wang and L. Wang, “Prediction of back-splicing sites reveals sequence compositional features of human circular rnas,” in *ICCABS*. IEEE, 2017, pp. 1–6. 41, 53
- [330] —, “Deep learning of the back-splicing code for circular rna formation,” *Bioinformatics*, vol. 35, no. 24, pp. 5235–5242, 2019. 12, 41, 42, 52
- [331] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich, X. He, P. Mieczkowski, S. A. Grimm, C. M. Perou *et al.*, “Mapsplice: accurate mapping of rna-seq reads for splice junction discovery,” *Nucleic acids research*, vol. 38, no. 18, pp. e178–e178, 2010. 11
- [332] L. Wang, M. Lui, S. N. Kim, J. Nivre, and T. Baldwin, “Predicting thread discourse structure over technical web forums,” in *EMNLP’11*. ACL, 2011, pp. 13–25. 21, 214
- [333] L. Wang and D. W. Oard, “Context-based message expansion for disentanglement of interleaved text conversations,” in *NAACL’09*. ACL, 2009, pp. 200–208. 21, 209, 213, 217, 218, 219, 223, 224
- [334] S. Wang and J. Jiang, “A compare-aggregate model for matching text sequences,” in *ICLR’17*, 2017. 21, 216
- [335] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, “Signed network embedding in social media,” in *SDM*. SIAM, 2017, pp. 327–335. 13
- [336] Y.-d. Wang, X.-k. Fu, W. Jiang, T. Wang, M.-H. Tsou, and X.-y. Ye, “Inferring urban air quality based on social media,” *Computers, Environment and Urban Systems*, vol. 66, pp. 110–116, 2017. 228
- [337] Z. Wang, Y. Yang, L. He, and J. Gu, “User identification within a shared account: Improving ip-tv recommender performance,” in *Proceedings of the 18th East European*

- Conference on Advances in Databases and Information Systems*, ser. ADBIS '14, 2014, pp. 219–233. 18, 130
- [338] M. Weimer, A. Karatzoglou, and A. Smola, “Improving maximum margin matrix factorization,” *Machine Learning*, vol. 72, no. 3, pp. 263–276, 2008. 148, 149
- [339] J. Weng and B.-S. Lee, “Event detection in twitter,” in *ICWSM*, 2011. 175
- [340] C. C. Werry, “Internet relay chat,” *Computer-mediated communication: Linguistic, social and cross-cultural perspectives*, pp. 47–63, 1996. 208
- [341] B. P. Wing and J. Baldridge, “Simple supervised document geolocation with geodesic grids,” in *ACL*, 2011. 176
- [342] B. W. Wojdyski, N. J. Evans, and M. G. Hoy, “Measuring sponsorship transparency in the age of native advertising,” *Journal of Consumer Affairs*, vol. 52, no. 1, pp. 115–137, 2018. 19, 152, 172
- [343] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, “Transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019. 31
- [344] D. E. Wood and S. L. Salzberg, “Kraken: ultrafast metagenomic sequence classification using exact alignments,” *Genome Biology*, vol. 15, no. 3, p. R46, 2014. [Online]. Available: <https://doi.org/10.1186/gb-2014-15-3-r46> 107
- [345] B. Wu, C. Xiong, M. Sun, and Z. Liu, “Query suggestion with feedback memory network,” in *WWW '18. International World Wide Web Conferences Steering Committee*, 2018, pp. 1563–1571. 20, 188, 198
- [346] T. Wu, Z. Deng, Z. Feng, D. J. Gaskin, D. Zhang, and R. Wang, “The effect of doctor-consumer interaction on social media on consumers’ health behaviors: Cross-sectional study,” *Journal of medical Internet research*, vol. 20, no. 2, 2018. 228

- [347] W. Wu, H. Li, and J. Xu, “Learning query and document similarities from click-through bipartite graph with metadata,” in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 687–696. 20
- [348] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, “Listwise approach to learning to rank: theory and algorithm,” in *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, 2008. 162
- [349] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li *et al.*, “Soapdenovo-trans: de novo transcriptome assembly with short rna-seq reads,” *Bioinformatics*, vol. 30, no. 12, pp. 1660–1666, 2014. 108
- [350] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, “End-to-end neural ad-hoc ranking with kernel pooling,” in *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 55–64. 10, 11, 23, 31
- [351] M. Yan, C. Li, C. Wu, B. Bi, W. Wang, J. Xia, and L. Si, “Idst at trec 2019 deep learning track: Deep cascade ranking with generation-based document expansion and pre-trained language modeling,” *NIST TREC 2019 Deep Learning Track*, 2020. 10, 26, 30
- [352] P. Yang and J. Lin, “Reproducing and generalizing semantic term matching in axiomatic information retrieval,” in *European Conference on Information Retrieval*. Springer, 2019, pp. 369–381. 241
- [353] W. Yang, H. Zhang, and J. Lin, “Simple applications of bert for ad hoc document retrieval,” *arXiv preprint arXiv:1903.10972*, 2019. 10, 22, 26
- [354] X. Yang, S. Kim, and Y. Sun, “How do influencers mention brands in social media? sponsorship prediction of instagram posts,” in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 101–104. 19, 152, 161, 164, 171

- [355] Y. Yang, Q. Hu, L. He, M. Ni, and Z. Wang, “Adaptive temporal model for iptv recommendation,” in *Proceedings of the 16th International Conference on Web-Age Information Management*, ser. WAIM ’15, 2015, pp. 260–271. [18](#), [130](#)
- [356] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical attention networks for document classification,” in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489. [27](#), [36](#), [179](#)
- [357] W. Yin, H. Schütze, B. Xiang, and B. Zhou, “Abcnn: Attention-based convolutional neural network for modeling sentence pairs,” *TACL*, vol. 4, pp. 259–272, 2016. [21](#), [210](#), [214](#), [216](#), [217](#), [222](#), [223](#)
- [358] C.-Y. Yu and H.-C. Kuo, “The emerging roles and functions of circular rnas and their generation,” *Journal of biomedical science*, vol. 26, no. 1, p. 29, 2019. [41](#)
- [359] H.-F. Yu, C.-J. Hsieh, Q. Lei, and I. S. Dhillon, “A greedy approach for budgeted maximum inner product search,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5453–5462. [16](#), [86](#), [87](#), [100](#)
- [360] H. Zamani and W. B. Croft, “Estimating embedding vectors for queries,” in *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*. ACM, 2016, pp. 123–132. [198](#)
- [361] —, “Relevance-based word embedding,” in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 505–514. [198](#)
- [362] D. R. Zerbino and E. Birney, “Velvet: algorithms for de novo short read assembly using de bruijn graphs,” *Genome research*, vol. 18, no. 5, pp. 821–829, 2008. [108](#)
- [363] A. Zhang, N. Fawaz, S. Ioannidis, and A. Montanari, “Guess who rated this movie: Identifying users through subspace clustering,” in *Proceedings of the Twenty-Eighth*

- Conference on Uncertainty in Artificial Intelligence*, ser. UAI'12, 2012, pp. 944–953. 18, 19, 130, 144, 145
- [364] J. Zhang, J. Guo, X. Yu, X. Yu, W. Guo *et al.*, “Mining K-mers of Various Lengths in Biological Sequences,” in *International Society for Biomedical Research on Alcoholism*. Springer, Cham, 2017, pp. 186–195. [Online]. Available: http://link.springer.com/10.1007/978-3-319-59575-7_{-}17 108, 111, 127
- [365] K. Zhang, C. Xiong, Z. Liu, and Z. Liu, “Selective weak supervision for neural information retrieval,” in *Proceedings of The Web Conference 2020*, 2020, pp. 474–485. 23, 29, 30, 240, 241
- [366] M. Zhang, X. Liu, W. Wang, J. Gao, and Y. He, “Navigating with graph representations for fast and scalable decoding of neural language models,” in *NIPS*, 2018. 16, 87, 91, 96, 97, 100
- [367] Q. Zhang, J. Pell, R. Canino-Koning, A. C. Howe, and C. T. Brown, “These are not the k-mers you are looking for: efficient online k-mer counting using a probabilistic data structure,” *PloS one*, vol. 9, no. 7, p. e101271, 2014. 18, 108
- [368] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *NIPS'15*, 2015, pp. 649–657. 210
- [369] X.-O. Zhang, H.-B. Wang, Y. Zhang, X. Lu, L.-L. Chen, and L. Yang, “Complementary sequence-mediated exon circularization,” *Cell*, vol. 159, no. 1, pp. 134–147, 2014. 11
- [370] X.-O. Zhang, R. Dong, Y. Zhang, J.-L. Zhang, Z. Luo, J. Zhang, L.-L. Chen, and L. Yang, “Diverse alternative back-splicing and alternative splicing landscape of circular rnas,” *Genome research*, vol. 26, no. 9, pp. 1277–1287, 2016. 53
- [371] Z. Zhang and W. Wang, “RNA-Skim: a rapid method for RNA-Seq quantification at transcript level.” *Bioinformatics*, vol. 30, no. 12, pp. i283–i292, 2014. [Online]. Available: <http://bioinformatics.oxfordjournals.org/content/30/12/i283.full> 107

- [372] C. Zhao, C. Xiong, C. Rosset, X. Song, P. Bennett, and S. Tiwary, “Transformer-xh: Multi-evidence reasoning with extra hop attention,” in *International Conference on Learning Representations*, 2020. [Online]. Available: <https://openreview.net/forum?id=r1eLiCNYwS> 10, 26, 30, 242
- [373] Y. Zhao, J. Cao, and Y. Tan, “Passenger prediction in shared accounts for flight service recommendation,” in *Proceedings of the 10th Asia-Pacific Services Computing Conference on Advances in Services Computing*, ser. APSCC '16, 2016, pp. 159–172. 18, 130
- [374] Y. Zhao, Z. Liu, and M. Sun, “Representation learning for measuring entity relatedness with rich information,” in *IJCAI*, 2015. 14
- [375] C. Zheng, J.-Y. Jiang, Y. Zhou, S. D. Young, and W. Wang, “Social media user geolocation via hybrid attention,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1641–1644. 5
- [376] C. Zheng, Q. Zhang, G. Long, C. Zhang, S. D. Young, and W. Wang, “Measuring time-sensitive and topic-specific influence in social networks with lstm and self-attention,” *IEEE Access*, 2020. 178
- [377] Y. Zheng, X. Yi, M. Li, R. Li, Z. Shan, E. Chang, and T. Li, “Forecasting fine-grained air quality based on big data,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 2267–2276. 227, 228, 229
- [378] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018. 178