University of California

Los Angeles

Feature Representation in Mining and Language Processing

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Thuy Vu

2017

Abstract of the Dissertation

Feature Representation in Mining and Language Processing

by

Thuy Vu
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2017
Professor Douglas S. Parker, Chair

Feature representation has been one of the most important factors for the success of machine learning algorithms. Since 2006, deep learning has been widely considered for various problems in different disciplines and, most of the time, has reset state-of-the-art results — thanks to its excellent ability to learn highly abstract representations of data. I focus on extracting additional structural features in network analysis and natural language processing (NLP) — via learning novel vector-based representations, usually known as embeddings.

For network analysis, I propose to learn representations for nodes, node embeddings, for social network applications. The embeddings are computed using attributes and links of nodes in the network. Experimental studies on community detection and mining tasks suggest that node embeddings can further reveal deeper structure of the network.

For NLP, I address the learning of representations at three levels: words, word relations, and linguistic expressions. First, I propose to extend the standard word embedding training process into two phases, treating context as second order in nature. This strategy can effectively compute embeddings for polysemous concepts of words, adding an extra conceptual layer for standard word embeddings. Second, I introduce the representations of "semantic binders" for words. These representations are learned using categorial grammar and are shown to effectively handle disambiguation, especially when meaning of a word largely depends on a specific context. Finally, I present a three-layer framework to learn representation for linguistic expressions — for solving the semantic compositionality problem, using recurrent neural networks driven by categorial-based combinatory rules. This strategy specifically addresses the limitations of recurrent neural network approaches in deciding how — and when — to include individual information in the compositional embedding. The framework is flexible and can be integrated with the proposed representations. I study the efficiency of the proposed representations in different NLP applications: word analogies, subject-verb-object agreement, paraphrasing, and sentiment analysis.

The dissertation of Thuy Vu is approved.

University of California, Los Angeles

2017

to my family

Table of Contents

List of Figures

List of Tables

Acknowledgments

Thank you to Professor D. Stott Parker, my advisor and mentor since my day-0 at UCLA. I am fortunate to be under his unique supervision. In every discussion, he patiently helped me explore all sorts of questions and possible perspectives of a research. There has been *hundreds* of these discussions over the past 67 months. One after another, I have become an independent researcher — and a better teacher, helping 581 students in 14 quarters teaching at UCLA. Stott is indeed my academic father.

I am indebted to my doctoral committee. I thank Professor Wei Wang for her insightful comments in my dissertation. I learned a lot about data mining in Wei's *Big Data Analytics* class. Thank you to Professor Edward L. Keenan for the generous guidance. Ed allowed me to sit in his *Mathematical Structures in Language* class, recommended relevant readings, and especially gave excellent feedbacks about my research. I have learned to enjoy and appreciate linguistics even more. I thank Professor Eleazar Eskin for the advice on my research and career development. I truly appreciated his introducing me to Bioinformatics, the topic I have been and will be reading more upon graduation.

Besides doing research, half of my UCLA time was for teaching. I thank Professor Paul Eggert for allowing me to assist him in most quarters. I learn from and admire Paul for his technical expertise and teaching principles. I also thank the CS department staff, especially Steve Arbuckle, for the helps on varied matters.

I would like to thank my friends and my colleagues at I^{2}R (Singapore), Zynx Health, IoES (UCLA), IPAM (UCLA), and the CS department (UCLA) for their friendship and collaboration. I thank Nguyen Do (BMAP) especially for his enormous support on providing the computing facility.

Finally, I thank my life partner, Tram Tong, for the immense support and strongholds. I thank my parents and siblings, for their unconditional trust, support, and prayer. This dissertation is practically impossible without them.

VITA

2001–2005	Bachelor of Science in Computer Science, University of Science, Vietnam National University at Ho Chi Minh City (VNU-HCM), Vietnam
2005–2006	Research Assistant, University of Science, VNU-HCM, Vietnam
2006–2011	Research Engineer, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
2011–2017	Teaching Assistant, Computer Science Department, UCLA
Summer 2012/13	Data Science Intern, Zynx Health Incorporated
2014–2015	Research Assistant, Institute of the Environment and Sustainability, UCLA
Summer 2016	Academic Mentor, Institute for Pure and Applied Mathematics, UCLA

Publications

Thuy Vu and D. Stott Parker. 2017. *Mining Community Structure with Node Embeddings.* Book Chapter in From Social Data Mining and Analysis to Prediction and Community Detection.

Thuy Vu and D. Stott Parker. 2017. *Extracting Urban Microclimates from Electricity Bills.* In AAAI 2017, Special Track on Computational Sustainability, San Francisco.

Thuy Vu and D. Stott Parker. 2016. *K-Embeddings: Learning Conceptual Embeddings for Words using Context.* In NAACL 2016, San Diego.

Thuy Vu and D. Stott Parker. 2015. *Node Embeddings in Social Network Analysis.* In ASONAM 2015, Paris.

Thuy Vu and Victor Perez. 2013. *Interest Mining from User Tweets* In CIKM 2013 Poster, San Francisco.

CHAPTER 1

Introduction

1.1 Introduction

It has been more than half a century since John McCarthy coined the term *artificial intelligence* (AI), defining it as "the science and engineering of making intelligent machines" [MMR55]. Impressive progress has been made in AI, but many challenges remain, and we have yet to satisfy this definition with current manmade intelligence [Tur50]. *Machine learning* (ML), a field closely related to AI, studies models that teach machines to learn *experience* from data for decision- and prediction-making. ML differentiates itself from AI on the objective of learning to solve problems as taught. Despite the differences, both AI and ML generally share one motivation — to make/teach machines to be intelligent.

Modeling *intelligence* is hard and, to some extent, equivalent to modeling every thing in the universe — as well as their interaction, communication, and composition. Therefore, *knowledge* representation — which includes representation of entities, their relations, and the mechanisms for reasoning, planning, and learning — is pivotal to AI and ML research [PMG97, RN03].

It is suggested that the performance of an algorithm generally depends on the abstraction level of the representation of the objects it handles — computability-wise. For example, it is usually straightforward for us, as human beings, to compare a "rose" and a "daisy". However, how to *teach* a machine to do this is not intuitively obvious. In practice, much attention in AI and ML has gone into engineering varied methods to unveil different abstractive and discriminative aspects of the data. In the computer science literature, *representation learning*, or also known as *feature learning*, studies algorithms that generate representations from raw data to be used in machine learning tasks.

The recent significant surge of interest in *deep learning* in the AI/ML literature is largely due to its ability to capture more abstract and discriminative aspects in learning representations. Deep learning (DL) includes a set of different learning architectures that learn representations for different kinds of data, from speech and signal data, to image and video, and to natural language.

In this thesis, I am motivated to study the adoption of deep learning and its possible extensions in two important problems: network analysis in data mining (DM), and language representation in natural language processing (NLP). Since 2009, DL has been widely and increasingly considered for almost all problems in DM and NLP. However, it appears that current proposed work in the literature tends to solve problems *computationally*, although it arguably overlooks certain factors that could contradict claimed insights and novel proposed solutions.

1.2 Problems and Contributions

I summarize the addressed problems and the thesis contributions in this section. I propose to extend applications of deep learning in two typical data-centric areas: data mining (DM) and natural language processing (NLP).

First, I study how the distributional hypothesis in learning representations can be beneficial for DM tasks, especially on social network studies. In particular, I propose to learn representations for nodes in networks using a strategy similar to the training of word embeddings. The impact of the methods is studied for community detection and network mining.

Second, I investigate different deep learning architectures, and apply these in the exploration of representations for more linguistic problems in NLP. Particularly, I address the learning of representations at three levels: words and conceptual words, word relations, and linguistic expressions.

1.2.1 Embeddings in a Network

We develop *node embeddings*, a distributed representation of nodes, for large-scale social network applications. We compute embeddings for nodes based on their attributes and links. We show that node embeddings can effectively reflect community structure in networks and thus, can be useful for a wide range of community related applications. We consider node embeddings in two different community related mining tasks: network mining and community detection.

First, we propose a generic integration of node embeddings for network processing in community detection algorithms. Our strategy aims to re-adjust input networks by adding and trimming links, using embedding-based node distances. We empirically show that the strategy can remove up to 32.16% of the links from the DBLP (computer science literature) citation network, yet improve performance for different algorithms by different evaluation metrics for community detections.

Second, we show that these embeddings can support many community-based mining tasks in social networks — including analyses of community homogeneity, distance, and detection of *community connectors* researchers (inter-community outliers, actors who connect communities) — thanks to the convenient yet efficient computation provided by node embeddings for structural comparisons. Our experimental results include many interesting insights about DBLP. For example, prior to 2013 the best way for research in *Natural Language & Speech* to gain "best-paper" recognition was to emphasize aspects related to *Machine Learning & Pattern Recognition*.

This chapter is based on the papers [VP15, VP16b]. Details are presented in Chapter 3.

1.2.2 Conceptual K-Embeddings for Words

We describe a technique for adding contextual distinctions to word embeddings by extending the usual embedding process — into two phases. The first phase resembles existing methods, but also constructs K classifications of concepts. The second phase uses these classifications in developing refined K embeddings for words (also known as: word K-embeddings).

The technique is iterative, scalable, and can be combined with other methods (including Word2Vec) in achieving still more expressive representations.

Experimental results show consistently large coverage gains on a Semantic-Syntactic Word Relationship test set for different K settings. For example, an overall gain of 20% is recorded at $K = 5$. In addition, we demonstrate that an iterative process can further tune the embeddings and gain an extra 1% ($K = 10$ in 3 iterations) on the same benchmark. The examples also show that polysemous concepts are meaningfully embedded in our K different conceptual embeddings for words.

This chapter is based on the paper [VP16a]. Details are presented in Chapter 4.

1.2.3 Embeddings in Semantic Compositionality

We address the compositionality problem from a linguistic standpoint. Specifically, we focus on two sub-problems: 1) modeling the *functional interactions* between words and phrases as they are combined for larger linguistic expressions, and 2) representing linguistic expressions using their constituent representations.

First, we propose a categorial-based semantic binder, namely SEBI, for short phrase compositionality in natural language. SEBI focuses on learning the representation of functional interactions between words and phrases as they are combined/composed for meaning representation. We observe that the Categorial Grammar formalism is highly effective for semantic computation on word sequences (e.g. phrases or sentences) and can be applied in the same vector space as regular word embeddings. Experimental results demonstrate that our strategy is highly competitive on multiple semantic compositionality tasks, even when applied in a straightforward way through single-dimensional vector calculations. This gives a way to extend current single-word and short-phrase embedding methods to represent compositional semantics, with many possible benefits for natural language processing. Details are presented in Chapter 5.

Second, we address the compositionality of linguistic expressions of arbitrary length. In general, we propose to represent an expression as a composite of embeddings via a three-level compositionality framework, relying on advances in recurrent neural networks and in linguistic formalisms. We argue that representation using a composite of component embeddings, if properly structured, can offer a solid added layer of information and produce a better representation. In particular, our strategy addresses the limitation of recurrent neural network approaches by suggesting, driven by categorial-based combinatory rules, how — and when — to include component information in the compositional embedding. In addition, we also show that our framework is flexible and can be incorporated into the proposed representations. Experimental results support our hypothesis — adding a gain in information corresponding to compositionality. Details are presented in Chapter 6.

CHAPTER 2

Background

How information should be described or represented is arguably the fundamental focus of many disciplinaries. It is important because expressive representation of information is crucial for information understanding, handling, and processing. In this chapter I explore the literature in linguistics and computer science on representations, and then explain how the following technical chapters benefit from this study.

First, I present in Section 2.1 the background of representation in linguistics. Specifically, I study syntactic and semantic representations, and how meanings of language expressions are derived from these representations. In addition, I explain how the idea of bare grammar [KS03] is beneficial for research on semantic representations.

Second, I study information representation, also known as feature learning, in machine learning in Section 2.2. I briefly cover different computational techniques in learning representations using neural networks.

Finally, I conclude the chapter with suggestions on how problems in data mining and natural language processing can leverage on research in linguistics. In this dissertation, I will refer to features, representations, and embeddings as conceptually interchangeable.

2.1 Representation in Linguistics

Language is a unique communication vehicle that distinguishes humans from other species. It is considered a basis for measuring intelligence due to the large range of expressions that can be described in language. The scientific study of language is called linguistics.

Research in linguistics varies fundamentally due to the complexity of human language. From a practical standpoint, two basic questions are: 1) whether an expression is correctly formed with respect to a certain language (syntactically); and 2) whether the content of the expression is correctly formed with respect to a logic (semantically). Both focus on the problem of representing knowledge.

2.1.1 Language

Formally, a language is defined by a set of all valid strings. In most cases, this set is unlimited and defined over a limited alphabet Σ. Formally, if Σ is the alphabet, a language L over Σ consists of all valid strings s, $s \in L$ and $L \subseteq \Sigma^*$.

In human language, the alphabet is usually described a bit differently, with a lexicon — a set of words and their intrinsic structural variants. The scientific study of words and their lexical structure in linguistics is called morphology. Morphology considers multiple types of variations including affix (prefix and suffix) forms, inflection, morphological structure, compounds, reduplication, and ablaut (irregular verbs or plural nouns are forms of ablaut). For simplicity, we may assume these are all subsumed by a complete lexicon.

Given a linguistic expression, there are three levels of analyses:

- lexical analysis – to study words and their morphology
- syntactic analysis – to study the linguistic combinations of words in a constituent structure, considering necessary syntactic dependencies
- semantic analysis – to study how meaning of an expression is derived, considering previous derivations.

Semantic analysis is usually considered the ultimate goal in standard natural language processing, before speech and pragmatic analysis. The entire analytical process is usually described with a grammar.

2.1.2 Grammar

The size of a language L is infinite, and thus, L can only be described formally by a set of structural rules — grammar. In linguistics, grammar is fundamental to knowledge representation, and is considered the *mental representation* of linguistic knowledge. It is generally suggested that the choice of grammatical model affects the interpretability of the representation. There are different models/frameworks of grammar, such as:

- Relational Grammar
- Arc Pair Grammar
- Categorial Grammar
- Lexical Functional Grammar
- Head-Driven Phrase Structure Grammar
- Government Binding Theory/Minimalism.

As mentioned, different formalisms describe different linguistic aspects, with different competence toward interpretability. Yet, grammatical representation is often the sole input for semantic analysis. Therefore, it is important to emphasize the structural representation, defined by the grammar model, in semantic analysis.

2.1.3 Meaning Acquisition

Studies of interpreted meaning for linguistic expressions is called semantics. It is, however, nontrivial to *quantify* or even *qualify* the meaning of an expression linguistically, and/or mathematically, due to many factual aspects:

- individuals having different background knowledge could have different interpretations of the same linguistic expression;
- a linguistic expression can have different parses, and thus can be interpreted differently.

Morever, *meaning* of a word can be very different in different context settings, which makes the interpretation even less straightfoward. For example, the word “computer” in the following situations carries different meanings:

(a) this is a computer.

(b) this is an abstract computer.

(c) this is a new computer.

(d) this is a toy computer.

(e) this is a computer program.

Beside grammatical representation, interpretation of meaning can also consider other factors, including lexical representation, syntactic relations between words and phrases, scope, anaphora, and so on. While it is linguistically plausible to explore deep linguistic factors, I limit the focus in this dissertation to only on the first two of these factors; they are considered *invariant* in the literature of *bare grammar* [KS03].

2.1.4 Bare Grammar

In their monograph on linguistic invariants, Keenan et. al. discuss structural commonalities between different languages under a minimal generative framework called *bare grammar* [KS03], independent of other grammatical models.

The idea of bare grammar is based on the following observations.

- all grammatical models define a class of, usually infinite, expressions;
- complex expressions and their constituents of simplex (or atomic) expressions are distinguished.

In other words, *all* grammatical models admit two commonalities: 1) relations always have *structural constituents*; and 2) expressions of a language always have *grammatical categories.* Formally, a bare grammar G_B is a 4-tuple $(V, Cat, Lex, Rule)$, where:

- V is a list of items in the vocabulary;
- Cat is a list of all possible categories that a word $w \in V$ can be assigned to;
- Lex is a lexicon of the grammar, $Lex \subseteq V \times Cat$;
- $Rule$ is a set of transition functions $f : (V^* \times Cat)^+ \longrightarrow V^* \times Cat$.

Furthermore, $V^* \times Cat$ is the set of all possible *expressions* of the grammar G_B. A language L_{G_B} is said to be generated by G_B is the closure of Lex under $Rule$.

It has been showed in [KS03] that bare grammars obtain multiple properties:

1. Effability – bare grammar is capable of defining the same set of expressions that could be defined by other grammars;
2. Structural Similarity – bare grammar estimates the sameness of structure assuming a series of structure-preserving substitutions is known;
3. Invariants – bare grammar also has invariant properties, including invariant semantics.

The bare grammar formalism sheds light on how semantic information should be captured. Specifically, it suggests a proper way to represent linguistic expressions for computational tasks, including phrasal similarity, paraphrasing, and so on. In addition, the notion of Cat is in line with the idea of distinguishing both syntactical and semantical aspects of the representations. Analyses are studied to suggest multiple major linguistic invariants that can be achieved using the model.

2.2 Feature Representation in Machine Learning

In machine learning, *intelligence* is generally measured by the degree to which a system, or a model, can perform a task as well as a (presumably *intelligent*) human. Generally, the input of machine learning algorithms is data that was created or generated by humans, and the output is a model. The learning algorithms, however, are not able to *comprehend* raw data directly, but instead rely on the input being converted into certain latent *readable* representations in order to build a model. This step of creating the latent representation is called feature learning.

Apparently, feature learning is the most important first step, with large subsequent impact on the performance of the model. The representation is expected not only to carry descriptive information needed to reconstruct the raw data (as much as possible), but also to make it convenient for the algorithms to process.

Feature learning strategies have been studied a lot in the literature, ranging from manual or empirical selection to automatic extraction of features. In general, there are two standard strategies to learn features or representations of data. One is to extract different features separately and develop models or multiple models based on the features. A hybrid or ensemble strategy can be considered as a next consolidating step when there are multiple models to combine individual models in making decisions. The other is to assume the distributional properties of all the related features, and iteratively embed all information into a single vector space.

2.2.1 Learning Embeddings

Typically, features are stacked in a real-value vector, called an *embedding*. The number of values in the vector is also the dimension of the embedding. Formally, an embedding is a mathematical representation of a structure, or information contained within.

Good embeddings usually exhibit high reconstructability. In other words, a good embedding realizes a structure in some vector space in a way that guarantees some structural preservation. Embedding learning algorithms are evaluated by their ability to learn highly abstract yet computationally convenient representations. A general feature learning algorithm is a map from an entity $\mathcal{X}$ into $\mathbb{R}^d$, $f : \mathcal{X} \to \mathbb{R}^d$. In general, an embedding is described in terms of the following aspects:

1. **Smoothness** refers to the fidelity of a representation, such that:

   $$x_1, x_2 \in \mathcal{X} : x_1 \approx x_2 \Rightarrow f(x_1) \approx f(x_2)$$

   In this formulation, the spatial similarity relation is preserved by f.

2. **Compositionality** refers to the explanatory ability of an embedding, mainly concerning how an artifact's attributes are embedded in the representation. In addition, it also denotes the ability to combine embeddings.

3. **Memory** aims to preserve the trajectory of an embedding in the space $\mathbb{R}^d$. Recurrent neural networks are capable of representing this property.

4. **Simplicity** refers to the ability of embeddings to implement compositionality in a straightforward compositional way. In practice, linear combinations of embeddings are popular.

2.2.2 Deep Neural Networks

Kernel methods describe data in a top-down process using kernel functions. One advantage of this strategy is that users can decide the expressiveness of the kernel functions to build the covariance matrix, or Gram matrix. This is problematic, however, when the number of training examples grows large. In general, the training time is between $O(n^2)$ and $O(n^3)$, where n is the size of the training set.

Neural networks, on the other hand, have made significant progress, mainly thanks to distributed learning from large training data. At a high level, neural network models aim to simulate the brain function of mechanisms described in neuroscience studies. Generally, a neural network consists of a set of neurons and activation mechanisms. A typical neuron has inputs, outputs, and an activation function. The inputs and outputs are real-valued vectors in some d-dimensional vector space. Formally, let $\mathbf{x} \in \mathbb{R}^d$ be the input and $\mathbf{y}$ be the output of a neuron. $\mathbf{y}$ is computed as follows:

$$\mathbf{y} = f(\mathbf{w}^\top \mathbf{x} + b)$$

where $f(\cdot)$ is an activation function. Some popular activation functions (also known as sigmoid functions) are:

- logistic function $f_\sigma(x) = \frac{1}{1+e^{-x}}$, where $f_\sigma(x) \in [0, 1]$
- tanh function $f_{\tanh}(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$, where $f_{\tanh}(x) \in [-1, 1]$

A network of neurons defines a neural network. Let M be the set of input neurons, where $m = |M|$. Each neuron m_i possesses a unique set of coefficients $\{w_i, b_i\}$. The parameter vectors of a neural network are $W = \{w_i : i \in [1, m]\}, W \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$.

A neural network is trained though maximization of an objective function. The result of a training process is the network parameter W. In practice, five different architectures are studied for different kinds of tasks.

2.2.2.1 Auto-encoder

A standard auto-encoder network is a 3-layer feedforward, non-recurrent neural network, having two functions — encoder f_θ and decoder g_θ.

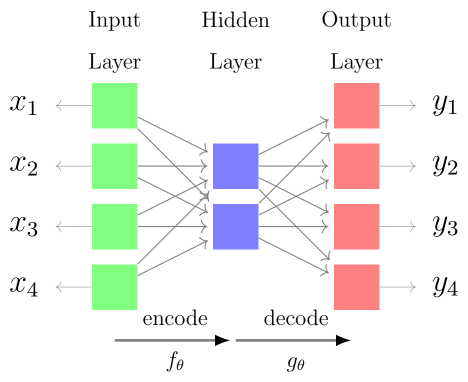


Figure 2.1: A sample of an auto-encoder network

$$f_\theta(\mathbf{x}) = s_f(W_f\mathbf{x} + b_f) = \mathbf{h} \tag{2.1}$$

$$g_\theta(\mathbf{h}) = s_g(W_g\mathbf{h} + b_g) = \mathbf{x}' \tag{2.2}$$

where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and $\mathbf{h} \in \mathbb{R}^p$; $f_\theta : \mathbb{R}^d \longrightarrow \mathbb{R}^p$ and $g_\theta : \mathbb{R}^p \longrightarrow \mathbb{R}^d$; $\theta = \{W_f, b_f, W_g, b_g\}$ is the network parameters. The hidden layer $\mathbf{h}$ is the feature representation or *code*. The training objective is to minimize the sum of reconstruction errors — the difference between $\mathbf{x}$ and $\mathbf{x}'$, $L(\mathbf{x}, \mathbf{x}')$

$$\text{argmin}_\theta \sum_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}, g_\theta(f_\theta(\mathbf{x}))). \tag{2.3}$$

An auto-encoder is used to find an efficient coding for the input when reducing its dimension. In other words, the size of p is much less than d in practice: $p \ll d$. Besides, there are other variants of auto-encoder, including the denoising auto-encoder (to neutralize corrupted inputs), sparse auto-encoder, and variational auto-encoder. In NLP, auto-encoders are used in order to extract features and train representations for words.

2.2.2.2 Convolutional Neural Network

A convolutional neural network (CNN) is a standard neural network extended across space using shared weights. It is a multilayer neural network, and is formed by stacking layers of different types, usually in a pipeline setting.

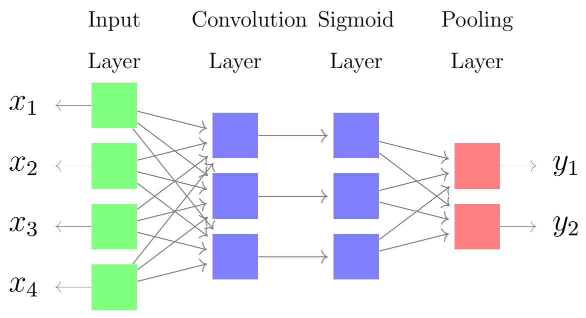


Figure 2.2: A sample of convolution neural network

A CNN is useful for connecting different tasks together through representation sharing. In the NLP litearature, a CNN is used to model systems that share representations across different tasks. Collobert et al. [CWB11] develop a unified framework for different tasks in NLP, including part-of-speech tagging, chunking, and named entity recognition.

2.2.2.3 Recurrent Neural Network

A recurrent neural network (RtNN) is a regular neural network with loops — where the output at time $t - 1$ will be the input at time t.

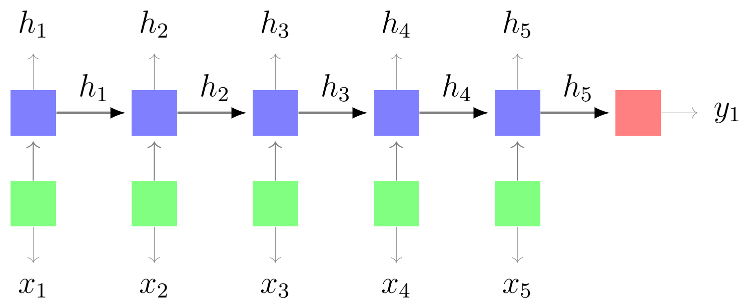


Figure 2.3: A sample of recurrent neural network

Recurrent neural networks are capable of maintaining an internal state when processing input sequences of arbitrary length. Therefore, the RtNN is useful for NLP tasks that require deep modeling of languages. The standard RtNN has a problem of a vanishing gradient if the input sequence is too long. Long-short term memory (LSTM) and the Gated Recurrent Unit (GRU) are the most popular models that use *gates* to control the problem. In Chapter 6, we propose a generic improvement for both of these models.

2.2.2.4 Recursive Neural Network

The Recursive neural network (RsNN) is also popular in natural language processing since it can efficiently learn structure through traversing the network in topological order. In general, weights are shared among neurons in the network in a recursive manner.

The RsNN is capable of processing a network-structured input. It is usually considered as a generalization of RtNN using the observation that sequential structure of RtNN is a special network structure. We think, however, this is only true to the extent of the structure propagation. In particular, although a sequence can be seen as a special case of a tree structure, the network architecture in RtNN is not a special case of those in RsNN. In our studies, directly specializing computations of RsNN for RtNN does not prove to be

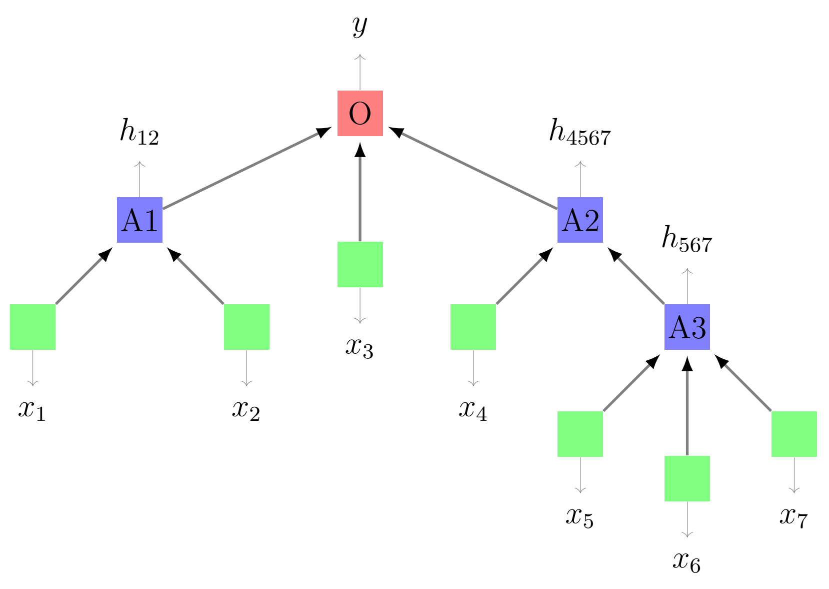


Figure 2.4: A sample of recursive neural network

effective; while imposing tree-structure on top of RtNN models consistently improves learning performance.

CHAPTER 3

Node Embeddings in Network Analysis

We develop *node embeddings*, a distributed representation of nodes, for large-scale social network applications. We compute embeddings for nodes based on their attributes and links. We show that node embeddings can effectively reflect community structure in networks and thus, be useful for a wide range of community related applications. We consider node embeddings in two different community related mining tasks.

First, we propose a generic integration of node embeddings for network processing in community detection algorithms. Our strategy aims to re-adjust input networks by adding and trimming links, using embedding-based node distances. We empirically show that the strategy can remove up to 32.16% of the links from the DBLP (computer science literature) citation network, yet improve performance for different algorithms by different evaluation metrics for community detection.

Second, we show that these embeddings can support many community-based mining tasks in social networks — including analyses of community homogeneity, distance, and detection of *community connectors* (inter-community outliers, actors who connect communities) — thanks to the convenient yet efficient computation provided by node embeddings for structural comparisons. Our experimental results include many interesting insights about DBLP. For example, prior to 2013 the best way for research in *Natural Language & Speech* to gain "best-paper" recognition was to emphasize aspects related to *Machine Learning & Pattern Recognition.*

3.1 Introduction

Mining of network data has gained great importance in recent years due to the significant developments in online social networking services and the wide applicability in different academic disciplines [FG, Geh09]. After decades of development on better mining algorithms [Han05], scalability has become the decisive factor in recent data science efforts. For example, network approximation has been explored as one solution for computational infeasibility in large network analysis problems, including detecting communities using betweenness centrality in networks of several thousands of nodes (or more) [RK14] and influence propagation in large social networks [GBL10].

Recently, the community in artificial intelligence (AI) and machine learning (ML) has made great progress in training representations using large artificial neural networks in a reasonable amount of time [HOT06]. This has enabled many applications and also spawned new research in machine learning areas involving *deep learning* using neural networks. Distributed representation learning using recurrent neural networks is one successful beneficiary of this research. Techniques in distributed representation learning have almost instantly set many new state-of-the-art records. More impressively, recent advances have shortened neural network training times for large data to reasonable levels. One recent example is the reduction of learning time for word representations training on 6 billion words to one day with roughly 300 cores [MCC13a], a decrease from 2500 cores in 2011 [CWB11] (more than a factor of 8) without significant compromise in performance.

Adoption of recurrent neural networks for distributed representation learning began early and is still gaining attention in many fields in computer science, including computer vision, speech recognition, and natural language processing (NLP). In NLP, deep learning has been adopted rapidly and in diverse ways to find better distributed representation for words, also known as word embeddings. Word embeddings have been successfully integrated in many NLP problems, in areas such as part-of-speech tagging and machine translation [CWB11], dependency parsing [LG14a] [CZZ15], and even sentiment analysis [SPW13] with impressive results. Deep representation learning is also studied in speech recognition [HCC14] and image processing [LSL14].

To the best of our understanding, few attempts have been made to apply neural networks for distributed representation learning in social network analysis and mining. Previously, we have presented applications of node embeddings in community-based network analysis [VP15]. In this paper, we extend our previous work in analyzing applications of node embeddings to the community detection problems. Our contribution is three-fold:

1. We introduce a simple yet generic method to learn embeddings for nodes in a network. Specifically, we adapt Skip-gram context handling algorithm for words in natural language processing to learn embeddings for nodes based on both their connectivity in the network and their own attributes.

2. We propose to use the learned embeddings to re-examine links in social networks. In particular, we employ embedding-based similarity metrics between nodes to re-adjust networks – adding new potential links and discarding distance-based irrelevant links. We experimentally show that newly re-adjusted networks can be beneficial in standard link-based community detection algorithms. We hypothesize that many algorithms on networks can do a better job when nodes are better spaced and links are more relevant.

3. We also show how node embeddings can be easily considered for mining tasks that require computing of similarity in networks, including community homogeneity, distance, and identification of *community connectors* (nodes with the important property of connecting communities). By treating research disciplines as communities, many interesting findings about the computer science literature have emerged from this work.

In addition, we construct ground truth for community detection in the computer science literature, DBLP, including best paper annotation in different conferences over the years for our experimental evaluations. We position our research with respect to distributed representation learning and community-based mining in social networks in Section 3.2. We then present our proposed node embeddings training in Section 3.3. We show how link re-adjustment and community-based mining are beneficial from the learned embeddings in social networks in Section 3.4 and 3.5 respectively. Our empirical studies are presented in Section 3.6, followed by conclusions in Section 3.7.

3.2 Background

3.2.1 Representation Learning

This paper is related to recent studies in learning distributed representations, especially in natural language processing (NLP). Distributed representations of concepts, known as embeddings, are usually encoded by high dimensional real-valued vectors. In general, the embeddings are learned through optimization of an objective function automatically on large volumes of unannotated data.

In NLP, neural-based language models represent language by embeddings of words in a high dimensional vector space. The learned word embeddings are then introduced for certain tasks that require word-based comparisons. Mikolov et. al. proposed continuous bag-of-words (CBOW) and Skip-gram, the current state of the art models [MCC13a]. Both are implemented and available from the `word2vec` project page: `code.google.com/p/word2vec/`. The main difference between these two models is in whether the objective is to learn representations to predict words (CBOW) or contexts (Skip-gram).

Our node embedding learning model is adapted from `word2vec`. We consider the Skip-gram for its sensitivity in dealing with low connection nodes, which is crucial when dealing with large social networks. Our work is also related to other research in learning neural-based language models [CWB11], neural word embeddings [LG14a, LG14b], word embeddings for speech recognition [BH14], and feature embeddings for dependency parsing [CZZ15].

3.2.2 Social Network Analysis and Mining

In recent years, the development of online social services has brought significant attention and rapid advances to studies in social network analysis and mining. Among different social network applications, community detection remains important because structural information is fundamental for large scale network analysis and mining. Community detection in networks has been extensively studied in the literature. Algorithms for this problem can be classified in three approaches with respect to the resources they rely on.

The first and most popular approach is to maximize distinctions in density among communities using network connectivity. For example, the algorithm proposed in [NG04] maximizes modularity score for a community setting of the network using edge-betweenness centrality. Even though this is a reliable method for best performance — aiming to maximize the evaluation metrics directly — it does not scale for networks with thousands of nodes due to high computation cost in repeated recomputation of edge betweenness after each iteration.

The second approach considers node-based features to determine communities. One exemplary work for this approach is [THP08], which developed the SNAP algorithm for grouping nodes using their attributes and relations in node connections. Node relations are among the identifying features for nodes, but are not relevant for the density measures considered in the previous approach. It is, however, worth noting that this approach performs best with sparse networks, where the network's degree distribution is largely uniform and the density is flat. It may not be effective for a regular social network when degree distributions follow a power law [CSN09].

The third approach is a hybrid of the previous two. Existing works on this approach include [RFP12, SYH09, YJC09, ZCY09]. In this paper, we suggest a strategy to bridge methods from the aforementioned two approaches. Thus, our proposed method should be classified in this category. Indeed, our node embeddings are basically learned from node descriptions, including attributes and relations with neighbors. And algorithms in the first approach can perform density-based community detection with similarity scores derived from the learned embeddings.

3.3 Distributed Representation in Networks

We present our proposed adaptation of the Skip-gram model for node embeddings training in this section. In general, we consider node attributes and links as *context* in a tuple (*node*, *context*) that we use to describe nodes of a network.

Formally, let $G(V, E, \mathcal{A})$ be a network with set of nodes V and set of edges E. Edges in E can be of arbitrary annotated type in a heterogeneous information network [SH12]. In addition, every node $v_i \in V$ is associated with descriptive attributes $\alpha_i \in \mathcal{A}, \alpha_i = \{a_1; a_2; \cdots\ a_{|\alpha_i|}\}$. Let D be the training input describing V, denoted as $\{(v_i, c_i)\}_{i=1}^{|V|}$, such that $c_i = (\alpha_i, \{v_j : (v_i, v_j) \in E\})$. The objective is as follows:

$$\max_{\mathbf{x}_v, \mathbf{x}_c} \left(\sum_{(v,c) \in D} \log \sigma(\mathbf{x}_c \cdot \mathbf{x}_v) + \sum_{(v,c) \in D'} \log \sigma(-\mathbf{x}_c \cdot \mathbf{x}_v) \right)$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $\mathbf{x}_v \in \mathbb{R}^d$ is embedding of node v. D' is generated from D through a negative sampling process. Specifically, $(v_i, c_i) \in D'$ means there exist indices x and y such that $(v_i, c_x | x \neq i) \in D$, $(v_y, c_i | y \neq i) \in D$, and $(v_i, c_i) \notin D$.

The training process maximizes the objective via a stochastic-gradient process. The output of the training includes embedding $\mathbf{x}_{v_i}$ for each v_i and embedding $\mathbf{x}_{c_j}$ for each context c_j. These embeddings for v_i and c_j should be proximally close if $(v_i, c_j) \in D$. As a result, two nodes sharing the same context should also be in close proximity to each other. The proximity between nodes can be estimated using Euclidean distance between their embeddings.

3.4 Embedding-based Community Detection (EBCD)

Two factors that define community are the connectivity density and the topical similarity of the vertices in the community. In the literature, many algorithms focus on the former through maximizing the community density by grouping vertices to maximize some graphical metrics, usually modularity. However, this might not yield satisfactory results in some applications when vertex similarity is more important. In addition, density-based algorithms usually suffer from computational infeasibility as the network size increases [RK14].

The immediate utility of node embeddings is the vector-based comparison of nodes in the network. This comparison has been proven effective in the NLP literature as we introduced earlier. Moreover, training of the embeddings can scale well and can be performed in an online setting. This benefits in handling online data in a social network. Finally, a wide range of descriptive features can be embedded in the same vector space, which makes comparisons feasible for networks of different types.

We consider three different ways to incorporate this similarity in community detection algorithms.

3.4.1 Node Clustering Algorithms

We consider nodes to be located independently in a d-dimensional space and assume no connections between nodes. We basically remove all edges and apply a vector quantization-based clustering algorithm to group vertices into K different groups depending on their proximity. Many clustering algorithms can fit this setting. For example, K-means and its variations including K-medians and K-medoids can be a good fit, depending on the nature of data. We consider K-means our base clustering algorithm in this paper. The basic idea of this algorithm is to partition a set of n observed data entities into K different clusters using proximity scores.

Even though the algorithm itself has some merit and the embeddings are proven representations, it is unusual for this method to yield impressive results. The biggest problem is its lack of consideration for network connectivity.

3.4.2 Community Detection with Weighted Network

We consider incorporating the distances between nodes directly into the network — as network weights. The literature shows that most community detection algorithms can handle weighted networks [FL09]. If an algorithm does not, however, it is still possible to consider weights by converting the network into unweighted multigraph format [New04]. This conversion allows multiple edges between two nodes.

Since the distance between embeddings means dissimilarity ($t \geq 0$), it is usually necessary to convert it into a similarity ($s \geq 0$) network before passing it into certain algorithms. There are many ways to convert between two metrics. Some standard conversion methods include:

- $s = 1 - t$, this conversion is only applicable when $t \in [0; 1]$
- $s = t^{-1}$, similarly, this conversion is straightforward and requires $t > 0$
- $s = (1 + t)^{-1}$, this conversion is similar to $s = t^{-1}$ without conditioning on t
- $s = \mathrm{e}^{\frac{-\beta \cdot t}{std(t)}}$, this is known as the heat kernel and can be simplified as follows:
- $s = \mathrm{e}^{-t^2}$

Adding weights into the network does encourage the algorithms to group closer nodes into the same community. However, this modification might not result in big differences in outcome since the algorithm itself still considers dense connection its main objective function. In other words, if the connectivity structure of the network remains unchanged, little difference in outcome can be expected.

3.4.3 $(\alpha; \beta)$ LInk Re-Adjustment in Networks (LIRA)

This strategy aims at resolving both of the above-mentioned weaknesses through combination. The direct clustering strategy has its own merits in combining nodes through their similarities. However, simply embedding distances in the network shows little effect when the network is sparse. Therefore, we propose to use grouping results from node clustering algorithms to re-adjust connectivity in the network. Specifically, we consider two operations on network edges:

- *disjoin* two nodes, removing the in-between edge, if they belong to two distant groups given by a clustering algorithm. The threshold $\alpha\%$ means that nodes belonging to the top $\alpha\%$ most-distant communities should be disjoined. Distance between communities is computed using the method in section 3.5.2.
- *join* two nodes if they belong to clusters that are immediately adjacent to each other. Each node is joined with the top $\beta\%$ such nodes.

3.4.4 Embedding-based Community Detection

Algorithm 1 presents our proposed integration of the network pre-processed by LIRA with a standard community detection algorithm, namely Embedding-based Community Detection (EBCD). This integration is generic and makes it easy for users to consider their favorite algorithms for clustering (as in line 2) and community detection (as in line 5).

Although LIRA networks are used in EBCD in this paper, we believe EBCD is also suited for other kinds of network analysis applications.

3.5 Mining in Social Networks

We present three applications of node embeddings in this section. First, we analyze homogeneity of communities in a network to study their concentration. Second, we propose to compute distance between communities. Last, we identify connectors of communities in a network. All computations for these applications rely on node embeddings to derive node similarities.

3.5.1 Community Homogeneity

Homogeneity for a community represents the degree of closeness of its members. Indeed, distances between members in a community reflects not only their topological structure but also their attributed dissimilarity. We define the homogeneity $h(\cdot)$ of a community c by the variance of all mutual distances:

$$h(c) = \left[\sum_{i=1}^{|E_c|} p_i \cdot (d_i - \mu_c)^2 \right]^{1/2}$$

Here $\mu_c = \sum_{i=1}^{|E_c|} p_i \cdot d_i$ is the mean of all distances in the community.

Algorithm 1 Embedding-based Community Detection

```
1:  procedure EBCD($G(V, E), \mathbf{e}_V, \alpha, \beta$)
2:      $K \leftarrow$ CLUSTERING($\mathbf{e}_V$)                       ▷ e.g. K-means
3:      $(D, J) \leftarrow$ LIRA($G, K, \alpha, \beta$)
4:      $G' \leftarrow G(V, E - D + J)$
5:      $C \leftarrow$ ComDet($G'$)             ▷ a generic community detection algorithm
6:      return $C$                                      ▷ community assignments
7:  end procedure
8:
9:  procedure LIRA($G(V, E), K, \alpha, \beta$)
10:     $D \leftarrow \emptyset$
11:     for all $e \in E$ do
12:         $(v_f, v_g) \leftarrow e$                           ▷ get both ends of $e$
13:         $c_{v_f} \leftarrow K(v_f)$                          ▷ get cluster of $v_f$
14:         $c_{v_g} \leftarrow K(v_g)$                          ▷ get cluster of $v_g$
15:         if $c_{v_f \text{ or } v_g} \in$ the $\alpha\%$ most distant to $c_{v_g \text{ or } v_f}$ then
16:             $D \leftarrow D + e$                          ▷ include $e$ into $D$
17:         else if $c_{v_f \text{ or } v_g} \in$ the $\beta\%$ most related to $c_{v_g \text{ or } v_f}$ then
18:             $J \leftarrow J + e$                          ▷ include $e$ into $J$
19:         end if
20:     end for
21:     return $(D, J)$
22: end procedure
```

3.5.2 Community Distance

There are many ways to define distance between two groups of entities. Distances in this paper consider all possible connections between two communities. Specifically, our distance between communities is the average total distance of all members. Formally, the distance $d(\cdot,\cdot)$ between two communities $\mathbf{C}_i$ and $\mathbf{C}_j$ is defined as:

$$d(\mathbf{C}_i;\mathbf{C}_j) = \frac{\sum_{(u;v)\in\mathbf{C}_i\times\mathbf{C}_j} d(u;v)}{|(u;v)\in\mathbf{C}_i\times\mathbf{C}_j|}$$

where u and v belong to community $\mathbf{C}_i$ and $\mathbf{C}_j$ respectively.

3.5.3 Community Connectors Identification

Finally, we define a *connector of a community* $\mathbf{C}_i$ *to community* $\mathbf{C}_j$ as a node $v \in \mathbf{C}_i$ that is closest (having smallest average distance) to all nodes in $\mathbf{C}_j$. This is a type of outlier that sits between communities. In other words, the set of connectors is:

$$c(\mathbf{C}_i,\mathbf{C}_j) = \{v \in \mathbf{C}_i : d(\{v\};\mathbf{C}_j) = min_{k\in\mathbf{C}_i} d(\{k\},\mathbf{C}_j)\}$$

Unlike influencers or leaders, community connectors are a type of inter-community outlier, and thus, not necessarily well-known. They usually play significant roles in a social network, especially in increasing communication among communities.

3.6 Experiments

We present our experimental studies for our node embeddings in this section.

3.6.1 Dataset Construction

We consider the DBLP citation network compiled by aminer.org in our experiments. We downloaded the version released in September 2013 (arnetminer.org/lab-datasets/citation/DBLP_citation_Sep_2013.rar). This dataset consists of 2,244,018 papers and 2,083,983 citation relationships for 299,565 papers (about 7 citations each). Each paper has title, authors, publishing venue, abstract, and citations.

We classify all papers and authors into fields in computer science. We consider 24 research fields compiled by Microsoft Academic Search (`academic.research.microsoft.com`). In each field, a list of related conferences, or publishing venues, is provided. We have a total of 2,695 different publishing venues in computer science classified into 24 different research fields.

It is not straightforward to align these research fields to `aminer.org`'s 8,881 publishing venues. We assign research fields for each of these venues using text classification on titles, and use these to build our community dataset. Specifically, we follow two steps:

1. We leverage clean alignments (exact matching) between two lists for training data for title classifiers. Altogether 798,293 papers in 2,123 different publishing venues have a cleanly-assigned field (∼81.8%). We use this as training set for string classification of titles — training 24 Naive Bayes classifiers for 24 fields using the Natural Language Toolkit (`nltk.org`). These classifiers share the same feature set of the 4,572 most frequent words, compiled by unifying the top 1,000 most frequent words in each field (with English stopwords removed). This vector size is large and representative for our classification tasks. Finally, each classifier is trained with the positive samples against the a random set of negative samples having a size of 5 times the number of positive samples to sharpen its discrimination against false classifications.

2. We assign fields to a venue by identifying the most popular field assigned to its papers though plurality voting. In addition, fields for 1,431,652 unclassified papers are assigned via their venues. Similarly, voting is also used to assign authors to fields as members. An author can be a member of a single field, while still being an author in multiple fields. There are 1,260,485 authors assigned memberships in 24 research fields in our dataset.

In addition, we also tag best papers in this dataset. We use the list of best paper awards since 1996 consolidated by Jeff Huang (`jeffhuang.com/best_paper_awards.html`). Table 3.1 displays the statistics for our annotated community dataset. This dataset is useful for data mining studies on Computer Science research.

3.6.2 Citation-based Author Embeddings in DBLP

We study the citation networks of authors in different fields in DBLP. For each paper, we extract two lists of authors: the paper authors (citers) and the authors of citing papers (citees). All possible connections between two author lists are extracted and passed to the representation learning in order to learn citation-based author embeddings. Embeddings for each author are the outcome of the learning process. More importantly, the proximity between authors denotes the similarity of their citing behavior. Presumably researchers working in a close community should exhibit high similarity (low distance) in citing behavior.

Each citation-based author embedding is a vector of 200 real numbers. We also filter out authors having a total of fewer than 500 citations in all their papers.

One immediate result of the learned embeddings is the ability to query *most similar* embeddings using Euclidean distance. In particular, our embeddings reflect citations from researchers to researchers, so *most similar* should mean high overlap in citing behavior. This shows high efficiency of the embeddings since the size is limited to 200 instead of the total number of researchers in the naive strategy. In addition, the embeddings can also capture other different aspects of citation activities. This provides us a tool for similarity queries. We denote $\mathbf{e}(.)$ the embedding representation of an actor. Table 3.2 shows a query example of the five researchers most similar to Christos Faloutsos, who has reportedly made 5,239 citations to 5,060 different researchers as of 2013. Christos Faloutsos is also one of the leading scientists in the "Data Mining" field. The first half of Table 3.2 shows that his citations largely overlap with many other leading researchers in "Databases". This is not a surprise since these two fields are historically related. In the second half of Table 3.2, we display Christos Faloutsos's top five most similar researchers after *subtracting* the embedding of Philip S. Yu, another famous researcher in "Data Mining". Interestingly, this reveals that

Christos Faloutsos shares significant interests with researchers in Computer Vision, especially in content-based retrieval.

In the following subsections, we continue to present experimental results using the resulting embeddings for other mining tasks.

3.6.3 Community Detection Results

In this section, we present the evaluation of our proposed strategy for community detection, EBCD. We consider K-means for clustering and five different algorithms for community detection: Fast Greedy [New03], Walk Trap [PL05], Leading Eigenvector [New06], InfoMap [RB08], and Multi-level [BGL08].

Our evaluation metrics include:

- Variation Inf. – variation of information; lower is better.
- Normalized MI – normalized mutual information; higher is better.
- Split-Join – node-based edit distance between two settings; lower is better.
- Adjusted Rand – Rand considers grouping chance; higher is better.
- Modularity – division strength of a network into modules; higher is better.

Table 3.3 and Table 3.4 present our results. We consider standard outputs of these community detection algorithms on original unweighted networks as baselines, presented in the first column *Baseline* in Table 3.3. The next column, *Weighted*, show the results on networks with edges weighted by the embedding distance of two end nodes. Underlined results denote better scores between *Baseline* and *Weighted*. As seen, results on weighted networks are mostly better than the baseline's, winning 21/25 comparisons.

The following two columns present the results of EBCD in two different parameter settings of (α, β) for LIRA. The second setting ($\alpha = 0.7$ and $\beta = 0.2$) is harsher than the first one ($\alpha = 0.2$ and $\beta = 0.2$) since a large amount of edges are filtered out. Specifically, of the

724,301 links in the original network, 257,559 ($\sim$35.56%) and 109,854 ($\sim$15.17%) links are filtered out with $\alpha = 0.7$ and $\alpha = 0.2$ respectively. After that, another 24,986 ($\sim$3.4%) and 27,650 ($\sim$3.8%) links are introduced with $\alpha = 0.2$ for these two settings.

As seen, results of EBCD outperform *Baseline* and *Weighted* in 16/25 comparisons, which implies that applying LIRA on networks as preprocessing for community detections can not only simplify the network structure thus resulting in faster computation, but also improve the overall performance when many distance-based irrelevant links are filtered out. Among five algorithms are considered, Walk Trap, Leading Eigenvector, and InfoMap usually yield best performance, which is in accordance with previous benchmarks [FL09, OLC11]. Finally, none of EBCD results could produce better performance in modularity comparisons. In other word, high modularity is only recorded when algorithms consider the full network, in either the original or our weighted networks. There are multiple reasons for this outcome; however, we think this is reasonable considering the modularity score of 0.381 of the ground truth, shown in Table 3.4.

In Table 3.4, we compare community detection results of InfoMap with K-means, by ignoring all link information (as presented in Section 3.4.1), and the ground truth. The results support our hypothesis that even though we cannot discard all links in networks for community detections, we can apply adjustments on links for better results. In addition, the results in Table 3.3 and Table 3.4 indicate that simply maximizing network modularity might not be the best objective function for real social networks.

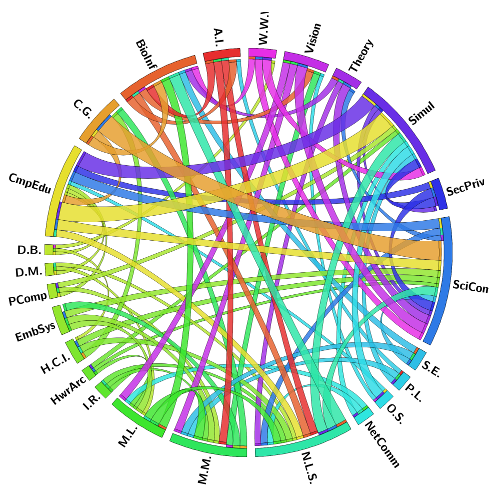
3.6.4 Mining in Community Data Results

Table 3.5 shows the *homogeneity* score for inbound citations to a research community (IC), and outbound citations to communities outside (OOC). The IC can be interpreted as the inner diversity in a community. We record comparable high homogeneity OOC scores for most communities. This shows that mutual communication among the fields is highly diverse. The result also indicates that *Machine Learning & Pattern Recognition* community is very selective in citing papers from outside. At the other extreme, it is interesting that, though sitting at different extremes in IC score ranking, *Scientific Computing* and *Hardware & Architecture* are the top two fields in citing outsiders. This might be explained by their research results being usually inspired and applied outside the community. The IC scores also yield many interesting insights. For example, the *Data Mining* community has many different and independent problems, and thus community communication is less converged; while its sister field *Natural Language & Speech* community (N.L.S.) is very connected and focused on languages.

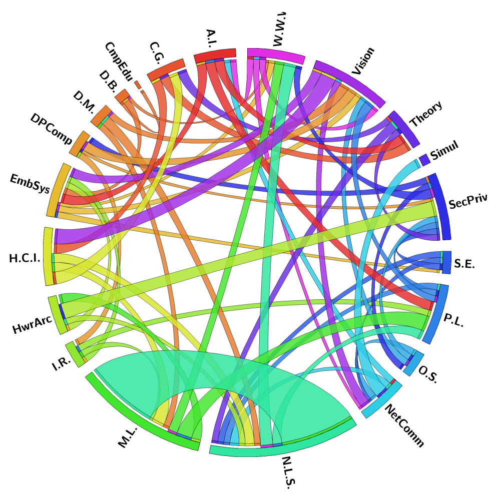
Second, we analyze the distance between communities in order to understand inter-field communication, and the OOC homogeneity score. For each community, we visualize its top three citing buddies (see Figure 3.1a). The visualization indeed provides many insights. For example, we can see that N.L.S. people frequently communicate with *Bio-Informatics* colleagues, while favorite buddies of *Human-Computer Interface* are in *Computer Graphics* and N.L.S. Each community provides insights of this kind.

In addition, we compare regular citations and impactful citations. By 'impactful' we mean pivotal works selected as best papers in conferences. In N.L.S., to our surprise, the impactful citations were to *Machine Learning*, *World Wide Web*, and *Programming Languages*. This could suggest 'impactful' research trends to follow. Another interesting impactful citation is from *Hardware & Architecture* to *Security & Privacy*, suggesting increasing interest in security. Also, *Vision* research might consider greater focus on *Human-Computer Interaction* for real-life impact.

Finally, we present some examples for community connectors in Table 3.6, for the *Data Mining* community. These authors could be considered outliers to the extent that they work between the two communities, or that their work is related to multiple research fields. For example, *David A. Cieslak* is a central researcher in *Data Mining* but has interests in a number of other fields, with researchers in different disciplines. Another example is *Jianhui Chen*. He is an active researcher in *Data Mining* with 30 publications (6 from KDD). At the same time, he is also working on dimensionality reduction and structural regularization, with papers in CVPR 2007, ICML 2009, and NIPS 2011. These works are commonly



(a) Regular citation in 24 fields in DBLP



(b) Best-paper citation in 24 fields in DBLP

Figure 3.1: Citation Analysis in DBLP

cited by the *Computer Vision* community. Interestingly, *Shunkai Fu* is connector for both *Human-Computer Interaction* and *Machine Learning & Pattern Recognition*. *Shunkai Fu* is also a founder of many mobile startups. In fact, these community connectors share similar motivations with impactful works above — trying to connect different research disciplines.

3.7 Conclusion

This paper reports on our development of *node embeddings*, an adaptation of representation learning, for use with nodes in social networks. Representation learning methods have been the source of impressive results in a variety of fields, but not social network analysis to our knowledge. We set out to investigate the successfulness of this approach here.

Essentially node embeddings represent a network in a d-dimensional vector space, in which each node is located independently. Links in networks are re-adjusted via our proposed EBCD using proximal measures between clusters of link's nodes. Gains recorded in community detection tasks on the resulting network indicate that our strategy can effectively re-adjust links using node embeddings in large network for more informative analysis and mining performance.

In addition, this paper also reports on large-scale experiments with a dataset covering the computer science literature. We implemented node embeddings for the DBLP citation network prior to September 2013, a network with over 2 million papers and over 2 million citations on about 300,000 papers, using 200-dimensional vectors in the representation of each node. The experimental results proved that node embedding is definitely useful, not only in social network analysis but also in general network analysis and mining.

Table 3.1: Statistical Results in Computer Science

Computer Science Field	#Member	#Author	#Paper
Algorithms & Theory	134,219	984,720	433,250
Artificial Intelligence	42,623	272,911	100,167
Bioinformatics & Computational Biology	32,896	126,921	43,481
Computer Education	9,177	40,551	15,160
Computer Vision	21,539	153,613	68,446
Data Mining	4,294	37,101	12,909
Databases	25,610	162,345	59,139
Distributed & Parallel Computing	22,338	161,578	53,603
Graphics	11,809	64,061	21,493
Hardware & Architecture	34,752	188,350	61,125
Human-Computer Interaction	209,087	748,969	262,602
Information Retrieval	14,713	81,586	26,056
Machine Learning & Recognition	19,675	117,681	40,575
Multimedia	19,638	110,776	35,842
Natural Language & Speech	26,989	149,874	53,617
Networks & Communications	257,752	1,220,809	431,104
Operating Systems	1,261	9,906	3,121
Programming Languages	14,655	93,974	48,572
Real-Time & Embedded Systems	279,306	946,254	328,979
Scientific Computing	4,922	18,189	7,066
Security & Privacy	16,560	82,131	31,265
Simulation	3,635	14,348	4,995
Software Engineering	38,863	173,551	63,863
World Wide Web	14,172	68,034	23,515
Total	1,260,485	6,028,233	2,229,945

Table 3.2: Most Similar Researcher Query

Top	“$\mathbf{e}$(Christos Faloutsos)”	Field	Distance
1	Hanan Samet	Algorithms	0.958
2	Caetano Traina Jr.	Databases	0.954
3	Thomas Seidl	Databases	0.953
4	Hans-Peter Kriegel	Databases	0.952
5	Christian Böhm	Databases	0.952
	“$\mathbf{e}$(Christos Faloutsos)\$\mathbf{e}$(Philip S. Yu)”		
1	Hans-Peter Kriegel	Databases	0.353
2	Edwin R. Hancock	Vision	0.334
3	David A. Forsyth	Vision	0.327
4	Thomas S. Huang	Vision	0.326
5	Alberto Del Bimbo	Networks	0.323

Table 3.3: Community Detection Result

Algorithm	Metric	*Baseline*	*Weighted*	$(0.2; 0.2)$	$(0.7; 0.2)$
Fast Greedy, 2003 $O((E+V)\times V)$ [New03]	Variation Inf.	<u>2.611</u>	2.696	**2.552**	2.598
	Normalized MI	0.190	<u>0.239</u>	0.256	**0.267**
	Split-Join	<u>**5,018**</u>	5,394	4,788	5,061
	Adjusted Rand	0.086	<u>0.098</u>	0.123	**0.133**
	Modularity	0.287	<u>**0.366**</u>	0.312	0.345
Walk Trap, 2005 $O(E\times V^2)$ [PL05]	Variation Inf.	3.049	3.162	3.207	**3.045**
	Normalized MI	0.402	0.409	0.407	**0.415**
	Split-Join	<u>**5,337**</u>	5,466	5,398	5,393
	Adjusted Rand	0.210	**0.211**	0.204	**0.211**
	Modularity	<u>**0.297**</u>	0.268	0.250	0.249
Leading Eigenvector, 2006 $O((E+V)\times V)$ [New06]	Variation Inf.	2.934	<u>2.721</u>	**2.765**	2.740
	Normalized MI	0.214	<u>0.240</u>	0.202	**0.317**
	Split-Join	5,813	<u>5,575</u>	5,073	**4,908**
	Adjusted Rand	0.082	<u>0.094</u>	0.073	**0.115**
	Modularity	0.320	<u>**0.329**</u>	0.302	0.306
InfoMap, 2008 $O(E)$ [RB08]	Variation Inf.	2.633	<u>2.575</u>	2.513	**2.405**
	Normalized MI	0.380	<u>0.385</u>	0.401	**0.403**
	Split-Join	4,831	<u>4,744</u>	4,618	**4,598**
	Adjusted Rand	0.196	<u>0.202</u>	0.205	**0.207**
	Modularity	0.419	<u>**0.428**</u>	0.393	0.396
Multi-level, 2008 $O(V\times\log V)$ [BGL08]	Variation Inf.	2.704	<u>2.675</u>	**2.669**	2.671
	Normalized MI	0.316	<u>**0.348**</u>	0.333	0.333
	Split-Join	5,101	<u>5,014</u>	5,010	**5,009**
	Adjusted Rand	0.168	<u>0.182</u>	**0.193**	0.188
	Modularity	0.419	<u>**0.423**</u>	0.399	0.403

Table 3.4: Community Detection Result

Metric	Baseline	Weighted	K-means	EBCD$(0.7; 0.2)$	Truth
VI	2.633	2.575	4.844	**2.405**	0.000
NMI	0.380	0.385	0.081	**0.403**	1.000
Split-Join	4,831	4,744	8,225	**4,598**	0.000
Adjusted Rand	0.196	0.202	0.019	**0.207**	1.000
Modularity	0.419	**0.428**	0.146	0.396	0.381

Table 3.5: Homogeneity Analysis: In-Community (IC) and Out-Of-Community (OOC), sorted by IC ascendingly

Computer Science Field	IC	OOC
Scientific Computing	0.168	0.678
Machine Learning & Pattern Recognition	0.216	0.401
Natural Language & Speech	0.229	0.406
Bioinformatics & Computational Biology	0.234	0.472
Graphics	0.264	0.403
Artificial Intelligence	0.281	0.420
Computer Education	0.281	0.448
Multimedia	0.284	0.448
Computer Vision	0.294	0.423
Security & Privacy	0.308	0.435
World Wide Web	0.343	0.431
Human-Computer Interaction	0.363	0.482
Real-Time & Embedded Systems	0.377	0.501
Information Retrieval	0.393	0.455
Programming Languages	0.433	0.513
Software Engineering	0.434	0.501
Algorithms & Theory	0.443	0.492
Networks & Communications	0.450	0.502
Databases	0.456	0.549
Simulation	0.459	0.513
Distributed & Parallel Computing	0.466	0.540
Data Mining	0.468	0.540
Operating Systems	0.519	0.536
Hardware & Architecture	0.579	0.610

Table 3.6: Connectors in the Data Mining Community

From Data Mining	Author	Score
Algorithms & Theory	David A. Cieslak	0.363
Artificial Intelligence	D. Sculley	0.244
Bioinformatics & Computational Biology	Muneaki Ohshima	0.318
Computer Education	Guilong Liu	0.295
Computer Vision	Jianhui Chen	0.258
Databases	Xingzhi Sun	0.528
Distributed & Parallel Computing	David A. Cieslak	0.496
Graphics	Jianhui Chen	0.274
Hardware & Architecture	Yanbo J. Wang	0.532
Human-Computer Interaction	Shunkai Fu	0.345
Information Retrieval	Ronen Feldman	0.363
Machine Learning & Pattern Recognition	Shunkai Fu	0.245
Multimedia	David A. Cieslak	0.289
Natural Language & Speech	Ata Kaban	0.259
Networks & Communications	David A. Cieslak	0.363
Operating Systems	Yanbo J. Wang	0.629
Programming Languages	Yanbo J. Wang	0.486
Real-Time & Embedded Systems	David A. Cieslak	0.387
Scientific Computing	David A. Cieslak	0.321
Security & Privacy	David A. Cieslak	0.301
Simulation	David A. Cieslak	0.444
Software Engineering	David A. Cieslak	0.362
World Wide Web	Makoto Haraguchi	0.355

CHAPTER 4

Word K-Embeddings

We describe a technique for adding contextual distinctions to word embeddings by extending the usual embedding process — into two phases. The first phase resembles existing methods, but also constructs K classifications of concepts. The second phase uses these classifications in developing refined K embeddings for words, which are referred to as *word K-embeddings*. The technique is iterative, scalable, and can be combined with other methods (including Word2Vec) in achieving still more expressive representations.

Experimental results show consistently large performance gains on a Semantic-Syntactic Word Relationship test set for different K settings. For example, an overall gain of 20% is recorded at $K = 5$. In addition, we demonstrate that an iterative process can further tune the embeddings and gain an extra 1% ($K = 10$ in 3 iterations) on the same benchmark. The examples also show that polysemous concepts are meaningfully embedded in our K different conceptual embeddings for words.

4.1 Introduction

Neural-based word embeddings are vectorial representations of words in high dimensional real valued space. Successes with these representations have resulted in their being considered for an increasing range of natural language processing (NLP) tasks. Recent advances in word embeddings have shown great effects that are pushing forward state-of-the-art results in NLP [KCC08, TRB10, CWB11, YZD13, MCC13a, MSC13, MYZ13]. For example, [KCC08] was an early attempt to consider word embeddings in dependency parsing. The consideration achieves accuracy gains of 1.14% for English and 1% for Czech on the Penn Treebank and the Prague Dependency Treebank dataset. Chunking and named-entity recognition (NER) have also been beneficial and have achieved new state-of-the-art results, simply by considering word embeddings as additional unsupervised features [TRB10]. Not only a free (and efficient) framework for performance gains, word embeddings have also become a go-to technique for boosting training time thanks to recent advances in deep learning in NLP [CWB11, MCC13a, MSC13, MYZ13]. Embedding learning models for words are also being adapted for tasks in other research fields [RMR15, VP15]. The Continuous bag of words (CBOW) and Skip-gram [MCC13a] are currently considered as state-of-the-art in learning algorithms for word embeddings.

The ability of words to assume different roles (syntax) or meanings (semantics) presents a basic challenge to the notion of word embedding [EP08, RM10a, HSM12, TDB14, NSP14, CXH15]. External resources and features are introduced to address this challenge. In general, individuals with no linguistic background can generally resolve these differences without difficulty. For example, they can distinguish "bank" as referring to a riverside or a financial establishment without semantic or syntactic analysis.

Distinctions of role and meaning often follow from context. The idea of exploiting context in linguistics was introduced with a distributional hypothesis: "linguistic items with similar distributions have similar meanings" [Har54]. Firth soon afterwards emphasized this in a famous quote: "a word is characterized by the company it keeps" [Fir57].

We propose to exploit only context information to distinguish different concepts behind words in this chapter. The contribution of this chapter is to note that a *two-phase word embedding* training can be helpful in adding contextual information to existing embedding methods:

- we limit *context* to mean the surrounding words of a given word.
- we use learned context embeddings to efficiently cluster word contexts into K classifications of concepts, independent of the word embeddings.
- this approach can complement existing sophisticated, linguistically-based features, and

can be used with word embeddings to achieve gains in performance by considering contextual distinctions for words.

- two-phase word embedding may have other applications as well, conceivably permitting some 'non-linear' refinements of linear embeddings.

In the next section, we discuss related work. We then present our learning strategy for word K-embeddings, outlining how the value of K affects its power in increasing syntactic and semantic distinctions in Section 4.3. Following this, a large-scale experiment serves to validate the idea — from several different perspectives. Finally, we offer conclusions about how adding contextual distinctions to word embeddings (with our second phase of embedding) can gain power in distinguishing between different aspects of words.

4.2 Related Work

Recently a dynamic community evaluation has focused on the potential of word embeddings for NLP [AK14], with many interests converging in the topic. Currently, researchers are working toward finding ways to enrich learned embeddings both syntactically and semantically; recent efforts include [YZD13].

The strategy of using compound features is traditional in NLP, and its efficiency has been proven in many NLP tasks; for example, it outperforms embeddings with simple Brown cluster features in [TRB10]. [YZD13] suggest that compound features should be considered, and specifically show that compound-style features in clustered embedding can improve Chunking and NER performance.

Similarly, [QCN14] proposed a method for incorporating word proximity and ambiguity awareness into word embeddings. Word proximity features are computed using distances derived from the CBOW model on words annotated with POS tags. Their results show improvement on CBOW and Skip-gram models with altered word and proximity considerations in both semantic and syntactic benchmarks.

A more significant approach to improve word embeddings, and directly related to our work, is to consider multiple representations for a single word to address polysemy. Multiple efforts have been reported in the literature. First, [EP08] is an early attempt to discriminate different meanings of a word using syntax information in context. In [RM10a], the authors proposed "prototype" vectors that represent different semantic clusters for a word. The word clusters are computed through context words clustering using the von Mises-Fisher (vMF) distribution. In addition, [HSM12] adopted a similar strategy to cluster context words, and considered idf-weighting of important context words. Recently, [TDB14] introduced an EM-based training method for multi-word embeddings.

Our technique is distinct from these works in literature. Specifically, we investigate another direction — the extension of the word embedding process into a second phase — which allows context information to be consolidated with the embeddings. Rather than annotating words with features, our technique treats context as second-order in nature, suggesting an additional representation step. We annotate words with their most common contextual role, learned by performing clustering algorithms (e.g. K-means) on the contextual word embeddings. Our method relies on the learned contextual vector representations. We have proven this method empirically to be efficient in many applications.

In addition, this two-phase technique is not difficult to implement with Word2Vec-like frameworks, is scalable, and can be combined with existing state-of-the-art strategies like the compound-feature methods here.

4.3 Learning Word K-Embeddings

As presented, the use of multiple semantic representations for a word in resolving polysemy has a significant literature [EP08, RM10a, HSM12, TDB14, NSP14, CXH15]. Strategies often focus on discrimination using syntactic and semantic information.

Our strategy for word K-embeddings is different and can be described in two, possibly iterative, phases:

1. Annotating words with concepts (defined by their contextual clusters)

2. Training embeddings using the resulting annotated text.

4.3.1 Concept Annotation using Context Embeddings

We propose to annotate words with concepts given by learned context embeddings, which are an under-utilized output of word embedding training [MCC13a, LG14a]. Our strategy is based on the assumption that the context of a word is useful for discriminating its conceptual alternatives in polysemy. In general, our concept annotation for words is performed in two steps — clustering of context embeddings followed by annotation.

Specifically, we first employ a clustering algorithm to cluster the context embeddings. K-means is our algorithm of choice. The clustering algorithm will assign each context word to a distinct cluster. This result is then used to re-assign words in the training data to their contextual cluster.

Second, we annotate words in the training data with their most common contextual cluster (of their context words). We define *context words* to mean the surrounding words of a given word. Formally, a word is annotated with a concept given by the following function:

$$\max_{c \in \mathcal{C}} \sum_{(w_i, c_i) \in W} f(c_i, c)$$

Here W is the set of context words of the current word, and $f(c_i, c_j)$ is a boolean function whose output is 1 if the input parameters are equal:

$$f(c_i, c_j) = \begin{cases} 1, & \text{if } c_i = c_j \\ 0, & \text{otherwise.} \end{cases}$$

The cluster-annotated dataset is then passed into the next training phase.

4.3.2 Training Word K-Embeddings

The second phase is similar to existing word embedding training systems. The number of clusters K defines the maximum number of different representations for words. Table 4.1

presents the statistics for different selections of K using the dataset mentioned in the Section 4.4.

K	total embeddings	vocabulary size	ratio
1	1,965,139	1,965,139	1.00
5	2,807,016	1,443,061	1.95
10	2,740,351	1,474,704	1.86
15	3,229,945	1,374,055	2.35
20	3,236,882	1,410,521	2.29
25	3,382,722	1,383,162	2.45
30	3,404,150	1,418,027	2.40

Table 4.1: Total embeddings and vocabulary size for different K for Wikipedia dataset. Words with frequency lower than 5 are filtered during pre-processing.

Each value K in Table 4.1 is shown with the total number of embeddings and vocabulary size. Words in the vocabulary can have up to K different embeddings for different annotated concepts. As K increases, the size of the vocabulary decreases — yet remains largely stable for different values of K greater than 1. This is explained by the instances of words being spread among different concepts, resulting in a lower word count per concept. In our setting, concept-annotated words with fewer than 5 occurrences are discarded during training of word embeddings.

It is interesting to note that the total number of embeddings is broadly stable and less affected by K. For example, as we allow up to 10 different concepts for a word ($K = 10$), the total number of embeddings grows only slightly compared to the result for $K = 1$. The average number of embeddings for a word is 1.86 for $K = 10$. In other words, concept annotations do converge as we increase K.

In addition, Table 4.2 shows the distribution of embeddings across different conceptual groups. It is interesting to see that group#5 covers 94.38% of the 1,474,704 vocabulary words. This suggests the group is mostly equivalent to the regular word embeddings. Our experiments presented in the following section also confirm that performance of standard word embeddings in this group is comparable to that of regular embeddings.

concept group id	total embeddings	%
0	568,510	20.75
1	183,041	6.68
2	76,212	2.78
3	56,832	2.07
4	24,435	0.89
5	1,391,828	50.79
6	61,223	2.23
7	25,742	0.94
8	75,080	2.74
9	62,215	2.27
10	215,233	7.85
Total	2,740,351	100

Table 4.2: Embedding count per conceptual group for $K = 10$

4.3.3 Word K-Embedding Training Workflow

Figure 4.1 presents our proposed workflow to train context-based conceptual word K-embeddings. Our system allows each word to have at most K different embeddings, where each is a representation for a certain concept.

The input to the workflow is a large-scale text dataset. Initially, we compute context embeddings for words as presented previously. We can derive context embeddings directly from the training of almost any context-based word embeddings, where word embeddings are computed via their context words.

Subsequently, we cluster context embeddings into groups which reflect varied concepts

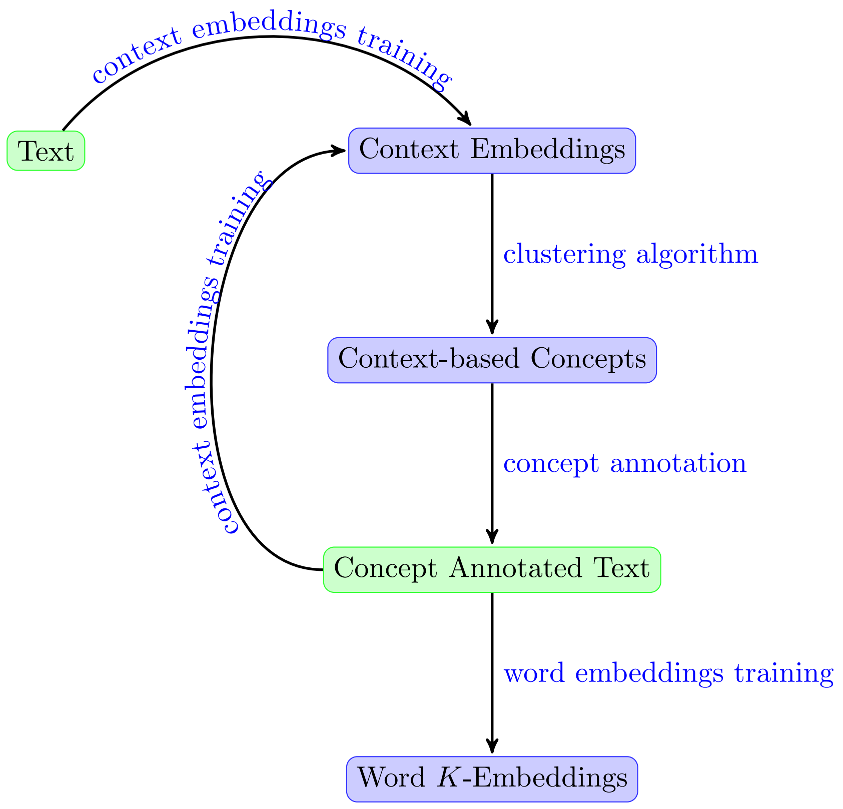


Figure 4.1: Training Word K-Embeddings

in some semantic vector space. Each context embedding is assigned to a cluster denoting its conceptual role as a context word. Any clustering algorithm for vectors can be applied for this task.

Embeddings of annotated context words are used to compute concepts of words in a sentence. Our hypothesis is that the concept of a word is defined by the concept of its surrounding words. We annotate concepts for all words in the training data.

Finally, the concept-annotated training data is passed into any standard algorithm for training word embeddings for the conceptual word K-embeddings.

4.4 Experiments

4.4.1 Settings

Our training data for word embeddings is Wikipedia for English, downloaded on November, 2014. It consists of 4,591,457 articles, with a total of 2,015,823,886 words. The dataset is pre-processed with sentence and word tokenization. We convert text to lower-case prior to training. We consider $|W|= 5$ for the size of the context window W presented in Section 4.3.1.

We used the Semantic-Syntactic Word Relationship test set [MCC13a] for our experimental studies. This dataset consists of 8,869 semantic and 10,675 syntactic queries. Each query is a tuple of four words (A, B, C, D) for the question "A is to B as C to what?". These queries can be either semantic or syntactic. D, to be predicted from the learned embeddings, is defined as the closest word to the vector $(A - B + C)$. We used Word2Vec for training and `scikit-learn` for clustering tasks.

We evaluate the accuracy of the prediction of D in these queries. A query is considered hit if there exists at least one correct match and all the words are in the same concept group. This is based on the assumption that if "A is to B as C is to D", either (A, B) and (C, D) OR (A, C) and (B, D) have to be in the same concept group.

4.4.2 Results

The embeddings learned in phases 1 and 2 can be compared, using different values for K in the K-means clustering. Word relationship performance results are shown in Table 4.3.

Our proposed technique in phase 2 achieves consistently high performance. For example, when $K = 5$, our absolute performance is 89% and 81% in semantic and syntactic relationship evaluations, gaining 24% and 16% from the standard CBOW model (phase 1). When $K = 25$, the performance yields the best combined result. As shown in Table 4.1, the total number of embeddings and vocabulary size differ by a small multiplicative factor as K increases.

In another comparison, Figure 4.2 plots our K-Embeddings results versus the results of a *relaxed* evaluation for CBOW, which considers the top K embeddings instead of the

Analogy Type	Total	CBOW	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 25$	$K = 30$
capital-country	506	85.18	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
capital-world	4,524	78.89	96.60	97.24	99.12	99.18	**99.29**	99.27
currency	866	20.01	36.72	31.18	40.65	41.22	**42.84**	44.80
city-in-state	2,467	44.75	90.76	89.26	95.42	97.16	97.28	**97.61**
family	506	85.38	96.25	99.01	97.23	98.42	97.83	**99.21**
semantic eval.	8,869	64.67	89.30	88.83	92.32	92.96	93.18	**93.53**
adj.-to-adverb	992	19.76	51.41	59.98	65.73	68.95	**70.06**	61.90
opposite	812	26.72	42.12	48.52	62.81	62.19	**68.84**	56.03
comparative	1,332	87.99	97.30	97.82	**99.25**	99.17	99.02	**99.25**
superlative	1,122	52.65	71.93	74.15	**81.02**	78.88	78.70	75.22
pres.participle	1,056	64.96	87.31	91.57	93.47	92.52	**96.78**	94.03
nationality	1,599	90.87	93.62	94.81	93.87	94.68	**95.37**	94.93
past-tense	1,560	65.51	66.28	94.42	94.49	95.45	**96.41**	93.72
plural	1,332	77.40	93.92	95.42	**97.90**	96.55	95.80	96.92
plural-verbs	870	66.78	83.33	94.60	94.71	95.63	**95.75**	93.22
syntactic eval.	10,675	65.17	81.72	85.94	88.83	88.93	**90.08**	87.21
combined eval.	19,544	64.94	85.16	87.25	90.42	90.76	**91.49**	90.08

Table 4.3: K-embeddings performance

best. Even though our evaluation is restricted to one-best for each of the K embeddings, the overall (combined) performance for different K settings is still consistently better than the top K embeddings of CBOW. Moreover, for a specific K setting, the total number of different embeddings considered in K-Embeddings is always less than that of the top K. For example, in our peak result ($K = 25$), the total number of embeddings considered in the evaluation set is only about 76.17% of the total embeddings with the top 25 of CBOW.

In addition, we also compare the performances of K-embeddings in multiple iterations under the same K setting in Table 4.4. It shows that the K-embeddings are improved after a certain number of iterations. In particular, for $K = 10$, we can achieve best performance

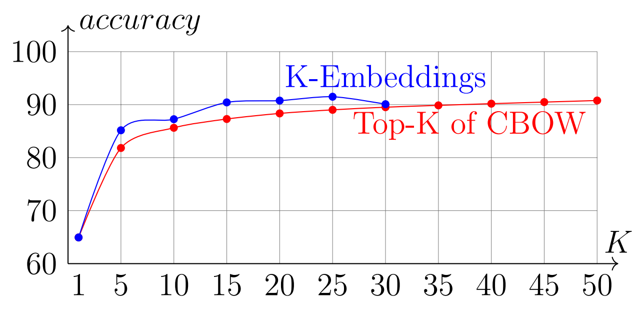


Figure 4.2: Word K-Embeddings and Top-K of CBOW accuracy comparison

after 3 to 4 iterations, gaining roughly 1%.

Type	iter_1	iter_2	iter_3	iter_4	iter_5
Semantic	88.8	89.6	**90.3**	90.0	89.0
Syntactic	85.9	84.8	**86.7**	**86.7**	85.8
Combined	87.3	87.0	**88.3**	88.2	87.3

Table 4.4: Performance of $K = 10$ in five iterations

Finally, it is also worth noting that the performance does not always increase linearly with the number of embeddings or vocabulary size. This suggests that as we achieve better performance in K-embeddings, we should also gain more compact conceptual embeddings.

4.4.3 Word Expressivity Analysis

Expressivity of word groups for "mercury" and "fan" are studied in Table 4.5, Table 4.6, and Table 4.7.

The first two rows shows most related words of "mercury" and "fan" without concepts annotation (baseline). The following rows present our K-embeddings result. This table illustrates the differences that arise in multiple representations of a word, and shows semantic distinctions among these representations.

For example, different representations for the word "mercury" indeed represent a spectrum of aspects for the word, ranging from related-cosmos, related chemical element, automobile, or even to music. The same can be seen for “fan” — where we find concepts related to fan as a follower/supporter, fan as in machinery, or Fan as a common Chinese surname. Indeed, we can find many different conceptual readings of these words. These not only reflect different polysemous meanings, but also their conceptual aspects in the real world. Observe that most related words are grouped into distinct concept groups, and thus yield high semantic distinctions. The result firmly suggests that context embeddings, like word embeddings, can capture efficiently high linguistic regularities.

4.5 Conclusion

In this chapter, we have presented a technique for adding contextual distinctions to word embeddings with a second phase of embedding training. This contextual information gains power in distinguishing among different aspects of words. Experimental results with embedding of the English variant of Wikipedia (over 2 billion words) shows significant improvements in both semantic- and syntactic- based word embedding performance. The result also presents a wide range of interesting concepts of words in expressivity analysis.

These results strongly support the idea of using context embeddings to exploit context information for problems in NLP. As we highlighted earlier, context embeddings are underutilized, even though word embeddings have been extensively exploited in multiple applications.

Furthermore, the contextual approach can complement existing sophisticated, linguistically-based features, and can be combined with other learning methods for embedding. These results are encouraging; they suggest that useful extensions of current methods are possible with two-phase embeddings.

#	Word	Most Similar Words
	Chinese	Taiwanese, Singaporean, Japanese, Taiwan, Han, Cantonese, Huang, Chen
	mercury	cadmium, barium, centaur, jupiter, venus, lanthanum, fluoride, nickel
0	drought	droughts, flooding, floods, famine, flood, wildfires, desiccation, inundation
	Disneyland	Disneysea, Fantasyland, Epcot, Tomorrowland, Imagineers, Frontierland
	fan	fans, fanbase, fan-base, supporter, fandom, supporters, followings, gamers, ultras
	Chinese_1	Japanese_1, Thai_1, sumac_1, rhubarb_1, viola_1, affinis_1, indica_1, European_1, aspera_1
	Chinese_3	Japanese_3, Bangkok_3, China_3, Language_3, Chinese_1, Oriental_3
1	Chinese_4	Taiwanese_4, Japanese_4, Vietnamese_4, Singaporean_4, Chung_4, Chan_4, Chinese-born_9
	Chinese_5	Taiwanese_5, Japanese_5, Qing_5, Han_5, Manchu_5, Thai_5, Burmese_5, Mongolian_5
	Chinese_8	Han_8, Ming_8, Yan_8, Lin_8, Qing_8, Tang_8, Yuan_8, Jin_8, Fa_8, Chen_8
	mercury_1	vanadium_1, iron_1, sulfur_1, polonium_1, thallium_1, antimony_1, lithium_1, iodine_1
	mercury_2	sodium_2, magnesium_2, molten_2, hydrogen_2, lamp_2, ammonia_2, helium_2, dust_2
1	mercury_3	tribune_3, dragon_3, curlew_3, keith_3, stanley_3, charlie_3, bonnet_3, wreck_3, barrett_3
	mercury_5	ammonia_5, magnesium_5, sulfur_5, arsenic_5, selenium_5, dust_5, radioactivity_5, iodine_5
	mercury_9	Neptune_9, Titan_9, Jupiter_9, meteor_9, cadmium_9, Sun_9, echo_9
	drought_0	winter_0, flood_0, earthquake_0, spring_0, snowfall_0, moist_0, floods_0, drought_9, rain_0
	drought_1	frost_1, rainfall_1, watering_1, pasture_1, dry_1, dryland_1, infestations_1, harvesting_1
1	drought_2	crisis_2, recession_2, vaporization_2, deforestation_2, flooding_2, tides_2, hunger_2
	drought_5	famine_5, flooding_5, floods_5, droughts_9, rainfall_5, deforestation_5, rain_5, epidemics_5
	drought_9	famine_9, floods_9, flooding_9, drought_2, floods_5, famines_9, famine_5
	disneyland_0	epcot_0, epcot_9, dreamworld_0, adventureland_0, safari_0, carousel_0, shangri-la_0
	disneyland_2	nightclub_2, disneyland_9, rides_2, paris_2, intamin_2, bowie_2, hyatt_2, epcot_0
1	disneyland_3	waterpark_3, westin_3, moana_3, casino_3, frazier_3, holiday_3, carpark_3, themed_3
	disneyland_5	disney_5, hotels_5, hotel_5, amusement_5, studios_5, epcot_0, restaurant_5, mall_5
	disneyland_9	epcot_9, disneysea_9, tomorrowland_9, efteling_9, legoland_9, fantasyland_9, monorail_9
	fan_1	inlet_1, crinoids_1, sect_1, wedge_1, beach_1, cuban_1, ball_1, valley_1, mound_1, flatfish_1
	fan_3	supporter_3, spell_3, left_3, resident_3, rufc_3, graduate_3, musician_3, fan_9, f.c._9, credited_3
1	fan_4	supporter_4, likes_4, legend_4, bust_4, member_4, disciple_4, parody_4, descendent_4, statue_4
	fan_5	fandom_5, fanbase_5, gamer_5, buzz_5, gamers_5, indie_5
	fan_8	Xiang_8, Yong_8, Xin_8, Yang_8, Cui_8, Guo_8, Wang_8, Xue_8, Lin_8, Zhang_8

Table 4.5: Word Expressivity Analysis: baseline vs. iteration #1

#	Word	Most Similar Words
	Chinese	Taiwanese, Singaporean, Japanese, Taiwan, Han, Cantonese, Huang, Chen
	mercury	cadmium, barium, centaur, jupiter, venus, lanthanum, fluoride, nickel
0	drought	droughts, flooding, floods, famine, flood, wildfires, desiccation, inundation
	Disneyland	Disneysea, Fantasyland, Epcot, Tomorrowland, Imagineers, Frontierland
	fan	fans, fanbase, fan-base, supporter, fandom, supporters, followings, gamers, ultras
	Chinese$_1$	Taiwanese$_1$, Japanese$_1$, Shanghai$_1$, Mongolian$_1$, Manchu$_1$, China$_1$, Taiwan$_1$
	Chinese$_2$	Sui$_2$, Luo$_2$, Xin$_2$, Chin$_2$, Tang$_2$, Ming$_2$, Mongolian$_2$, Ju$_2$, Han$_2$, Ye$_2$
3	Chinese$_6$	Japanese$_6$, Taiwanese$_6$, Thai$_6$, Han$_6$, Mongolian$_6$, Manchu$_6$, Lao$_6$, Qing$_6$, Burmese$_6$
	Chinese$_7$	kimchi$_7$, sumac$_7$, bicolor$_7$, army$_7$, currant$_7$, pistachio$_7$, guava$_7$, camellia$_7$, pickled$_7$
	Chinese$_8$	Pinyin$_8$, Japanese$_8$, Hangul$_8$, Thai$_8$, Taiwan$_8$, Mandarin$_8$, Tai$_8$
	mercury$_1$	polaris$_1$, cadmium$_1$, nimbus$_1$, neptune$_1$, lithium$_1$, liquid$_1$, triton$_1$, mars$_1$
	mercury$_3$	comet$_3$, neptune$_3$, argo$_3$, stax$_3$, org$_3$, echo$_3$, columbia$_3$, scepter$_3$
3	mercury$_6$	arsenic$_6$, lithium$_6$, oxygen$_6$, methane$_6$, dust$_6$, cadmium$_6$, sulfur$_6$, sulfate$_6$, dioxins$_6$
	mercury$_7$	cadmium$_7$, nickel$_7$, pollutants$_7$, impurities$_7$, ammonia$_7$, dioxins$_7$, sulfates$_7$, bismuth$_7$
	mercury$_9$	stakes$_9$, jason$_9$, sean$_9$, ted$_9$, jamie$_9$, journal$_9$, allenby$_9$, bolingbroke$_9$, jensen$_9$
	drought$_1$	floods$_1$, flooding$_1$, snowstorm$_1$, rainstorm$_1$, snowstorms$_1$, famine$_1$
	drought$_3$	flooding$_3$, floods$_3$, rain$_3$, excess$_3$, monsoon$_3$, epidemic$_3$, earthquake$_3$, economy$_3$
3	drought$_6$	flooding$_6$, floods$_6$, famine$_6$, famines$_6$, deforestation$_6$, rains$_6$, winter$_6$, rain$_6$
	drought$_7$	grandiflora$_7$, grasshopper$_7$, drongos$_7$, latifolia$_7$, floribunda$_7$, nutrient-poor$_7$
	drought$_8$	winter$_8$, tides$_8$, floods$_8$, rain$_8$, flooding$_8$, cold$_8$, disturbed$_8$, salinity$_8$
	disneyland$_1$	tomorrowland$_1$, epcot$_1$, disneysea$_1$, playland$_1$, amusement$_1$, resort$_1$
	disneyland$_3$	epcot$_3$, fantasyland$_3$, resort$_3$, drive-in$_3$, safari$_3$, dreamworld$_3$, seaworld$_3$, zoo$_3$
3	disneyland$_6$	disney$_6$, hotel$_6$, hotels$_6$, studio$_6$, mall$_6$, casino$_6$, pavilion$_6$, park$_6$
	disneyland$_8$	u-bahn$_8$, tram$_8$, victoria$_8$, jubilee$_8$, piccadilly$_8$, sightseeing$_8$, limited-stop$_8$, tour$_8$
	disneyland$_9$	casino$_9$, gaylord$_9$, ski$_9$, radisson$_9$, concourse$_9$, this$_9$, wembley$_9$, tanglewood$_9$
	fan$_2$	yong$_2$, ye$_2$, ching$_2$, hao$_2$, yi$_2$, chang$_2$, guo$_2$, yan$_2$, ying$_2$, peng$_2$
	fan$_4$	member$_4$, parody$_4$, protg$_4$, supporter$_4$, friend$_4$, follower$_4$, disciple$_4$
3	fan$_6$	fanbase$_6$, buzz$_6$, fans$_3$, fandom$_6$, video$_6$, gamer$_6$, sports$_6$
	fan$_7$	imprints$_7$, gnatcatchers$_7$, minuta$_7$, flat$_7$, subg$_7$, volutes$_7$, umbilicus$_7$, fasciata$_7$
	fan$_8$	impeller$_8$, inlet$_8$, spinner$_8$, spring$_8$, hot$_8$, hose$_8$, loyal$_8$, heater$_8$, ducts$_8$

Table 4.6: Word Expressivity Analysis: baseline vs. iteration #3

#	Word	Most Similar Words
	Chinese	Taiwanese, Singaporean, Japanese, Taiwan, Han, Cantonese, Huang, Chen
	mercury	cadmium, barium, centaur, jupiter, venus, lanthanum, fluoride, nickel
0	drought	droughts, flooding, floods, famine, flood, wildfires, desiccation, inundation
	Disneyland	Disneysea, Fantasyland, Epcot, Tomorrowland, Imagineers, Frontierland
	fan	fans, fanbase, fan-base, supporter, fandom, supporters, followings, gamers, ultras
	Chinese_0	Taiwan_0, Taiwanese_0, China_0, Malaysian_0, Singaporean_0, Japanese_0
	Chinese_1	Min_1, Han_1, Jin_1, Ming_1, Yuan_1, Tang_1, Yan_1, Hui_1, Ying_1, Mongolian_1
5	Chinese_4	Pinyin_4, Japanese_4, Thai_4, Hangul_4, Russian_4, Vietnamese_4, Romanization_4
	Chinese_8	traditional_8, pickled_8, sliced_8, persian_8, bean_8, kimchi_8, oxalis_8, bengal_8, arabica_8
	Chinese_9	Cantonese_9, Thai_9, Vietnamese_9, Mongolian_9, Yi_9, Burmese_9, Japanese_9
	mercury_3	media_3, rock_3, record_3, nbc_3, sky_3, fans_3, show_3, labels_3, cbc_3, abc_3
	mercury_5	arsenic_5, radiation_5, radioactivity_5, fluoride_5, sulfur_5, ammonia_5, sodium_5, methane_5
5	mercury_6	zinc_6, manganese_6, organs_6, technologies_6, minerals_6, aluminium_6, platinum_6
	mercury_7	radium_7, rolls-royce_7, cobalt_7, ammonia_7, titanium_7, orion_7, venus_7, mars_7
	mercury_8	chlorine_8, sulfur_8, methane_8, dioxins_8, nitrogen_8, phosphorus_8, cadmium_8, arsenic_8
	drought_0	reign_0, span_0, streak_0, recession_0, layoff_0, tenure_0, period_0, feud_0
	drought_3	floods_3, flooding_3, earthquake_3, rain_3, recession_3, famine_3, flood_3, shortages_3
5	drought_5	flooding_5, famine_5, rain_5, floods_5, rainfall_5, deforestation_5, malnutrition_5, frosts_5
	drought_8	paleoecology_8, vine_8, burundi_8, weedy_8, bonsai_8, shade_8, dry_8
	drought_9	flood_9, migration_9, eruption_9, sickness_9, stream_9, water_9, precipitation_9, soil_9
	disneyland_2	waterpark_2, ski_2, mandalay_2, rangeley_2, pinehurst_2, casino_2, gaylord_2, fairground_2
	disneyland_4	lakeside_4, brampton_4, haute_4, maxi_4, metrorail_4, vale_4, hermitage_4, bury_4, stratford_4
5	disneyland_5	mall_5, hotels_5, disney_5, mini_5, themed_5, attractions_5, rides_5, logo_5, nightlife_5
	disneyland_7	casino_7, monorail_7, sentosa_7, ride_7, hibiya_7, sheraton_7, tomorrowland_7
	disneyland_9	carnival_9, casino_9, amusement_9, tomorrowland_9, resort_9, carousel_9, fantasyland_9
	fan_1	zheng_1, cheng_1, chen_1, dai_1, jie_1, quan_1, fang_1, lu_1, meng_1, yan_1
	fan_3	song_3, comedian_3, fan_9, prank_3, andy_3, lyricist_3, songwriter_3, favorite_3, jockey_3
5	fan_4	spinner_4, rotors_4, flywheel_4, impeller_4, bulbous_4, nozzle_4, scoop_4, bowl_4, rotor_4
	fan_6	friend_6, protégé_6, supporter_6, follower_6, classmate_6, neighbor_6, member_6, promoter_6
	fan_9	supporter_9, member_9, fandom_9, fanbase_9, hobby_9, buzz_9, lot_9

Table 4.7: Word Expressivity Analysis: baseline vs. iteration #5

CHAPTER 5

Semantic Binder in Compositionality

We introduce SEBI, a portable semantic binder for short-phrase compositionality in natural language. We argue that the categorial grammar formalism is highly effective for semantic computation on a sequence of words — such as a phrase or sentence — and can be modeled in the same vector space as regular word embeddings. SEBI directly models the linguistic interactions of words and phrases when they are composed for meaning representation. The term 'compositionality' focuses on functional interactions between words and phrases, as they are combined/composed for meaning representation.

Specifically, we use high dimensional vectors, like the ones for word embeddings, to represent interactions between words and phrases. The experimental results demonstrate that our simple strategy is highly competitive on multiple semantic compositionality tasks, even when applied in a straightforward way through single dimensional vector calculations. This suggests a way to extend current single-word and short-phrase embedding methods to represent compositional semantics, with many possible benefits for natural language processing.

5.1 Introduction

Learning vectorial representation of words, also widely known as *word embeddings*, has been extensively studied in the literature during the recent surge in deep learning research [CWB11, KMK11, MCC13b, MYZ13, PSM14]. Concurrently, the research community has also reported on novel applications of word embeddings, with new state-of-the-art results. For word embeddings, `word2vec` has been a tool of choice for simplicity and scalability in large-scale training.

In applications, semantic compositionality in natural language has been widely studied, and is considered important not only for regular language processing tasks, but also as a general problem in artificial intelligence (AI): interpreting a larger structure in terms of the semantic interpretations of its components [Man15]. Typically, compositionality refers to a class of problems concerning the explanation of a large entity in terms of its components. Strategies involving compositionality have been important in advanced problems in linguistic research and in a broad spectrum of AI applications.

Specifically in NLP, compositionality concerns the computation of semantic representations for a (usually grammatically ordered) sequence of words, such as a phrase or sentence. Learning representations of this kind is difficult because of two fundamental challenges.

First, capturing the meaning of text is hard, especially when semantic ambiguity is involved. Current approaches for compositionality usually employ multidimensional arrays, or tensors, to capture different possible contextual settings. This, however, runs into the curse of dimensionality problem [SCA13]. For example, a rank-3 tensor needed to disambiguate different possible meanings of a verb described given a subject and an object can easily take up to 1GB (10^9 bytes) if the dimension is set to be 1000 and a byte is used to represent a real number [KS14]. This sort of combinatorial explosion clearly will be a challenge when systems must be scaled up to practical applications.

Second, semantic contributions of individual words in a phrase (or a sentence) are highly varied and depend on different linguistic regularities, including syntactic structure and linguistic attention. In addition, a given word meaning can have different impacts in different grammatical settings. It has been empirically shown, for example, that linguistic attention can be a surprising important feature in boosting traditional NLP applications, including machine translation [BCB14, LPM15], summarization [IKC16], and text classification [YYD16].

In this chapter, we address both challenges jointly, and base our methodology on linguistics studies. Specifically, we extend the application of the distributional hypothesis of Harris and Firth [Har54, Fir57] to learning of binding embeddings for phrases and semantic disambiguation. In addition, we rely on Frege's principle of compositionality and Montague's proposal of lambda calculus to model language and to compute meaning representation for phrases and sentences.

The remainder of the chapter is organized as follows: after presenting linguistic background, we describe our learning framework for compositionality in detail. In particular, it explains our study of word sense estimation as a way to handle disambiguation and composition embeddings. Experiments are discussed after that, and followed by discussion of related work.

5.2 Background

Formal representation of *meaning* is arguably the most important subject in linguistics, for either formal or natural languages. This study focuses on modeling the composition of linguistic entities and how it can be encoded, represented, computed, and interpreted in language.

5.2.1 Principle of Compositionality

In linguistics, Frege's principle establishes an arithmetic foundation on semantic compositionality. The principle is often considered the cornerstone of formal semantics studies. The principle states that:

> An English expression **e** is derived from expression $e_1, e_2, \cdots, e_n$ if and only if the interpretation of **e** is explicitly given as a function of the interpretations of $e_1, e_2, \cdots, e_n$.

Richard Montague adopted the principle and formalized it in *lambda calculus*, stating that: "the meaning of the whole is a function of the meaning of its parts and their *syntactic combination*."

The Montague formalism implies that syntax and semantics in natural languages can be viewed homomorphically in a single mathematical framework. This, in principle, distinguishes it from the generative grammar formalism, which views language as a system of generative rules on which all linguistic phenomena are set.

5.2.2 Combinatory Categorial Grammar

Categorial grammar [Ajd35] (CG) refers to a family of lexicalized formalisms motivated by Frege's principle and Montague formalism for the semantic compositionality in natural languages. In CG, words are assigned to their grammatical *categories* (or *types*) which, as a distinguishing feature, can be combined with a functional mechanism. A standard categorial grammar has two inference rules:

- B/A: the type of phrase x when preceded by A, $Ax \rightarrow B$
- $A\backslash B$: the type of phrase x when followed by A, $xA \rightarrow B$

Recent advances in distributional models for words have encouraged studies in modeling compositionality for language processing tasks. Combinatory Categorial Grammar [Ste00] (CCG) is introduced with this clear purpose — suggesting a linguistically-efficient calculus of text meaning. Figure 5.1 shows a sample CCG parse suggesting a construction path of the sentence meaning, where the combination of all constituents yields S.

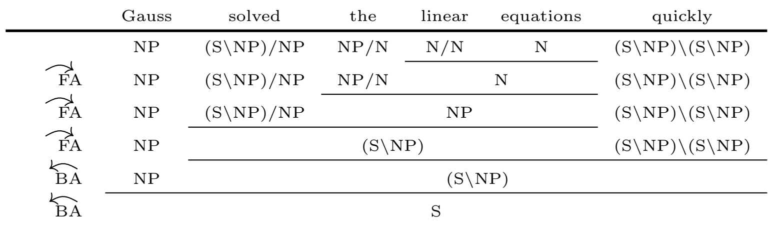


Figure 5.1: An example of CCG parse showing a derivation
NP · (S\NP)/NP · NP/N · N/N · N · (S\NP)\(S\NP) ⇒ S

The derivation in Figure 5.1 shows a sentence composition, where the combination of all constituents yields S. CCG has been used to model multiple linguistic phenomena in NLP applications. Currently, [CC07] is the state-of-the-art parser evaluated using CCG-Bank [HS07, Hoc06], a translation of the Penn Treebank (PTB) into CCG that can cover

99.4% of the sentences in PTB. In addition, there are many other works on CCG parsers, including [HS02, Hoc03, CC04, CC07]. For example, machine translation has adopted CCG categories and parsers [WCL12, BOK07, HSW09]. CCG is also employed for quantifier semantics acquisition modeling [PGE08], and grammar induction from strings [ZC12].

5.2.3 Related Works in Compositionality

Semantic compositionality in language has been considered an important and impactful application in the current NLP literature [Man15]. Previous work shares the same principal strategy in combining vectorial representations of words in certain ways. Combinations are usually done via two algebraic operators: $+$ as simple vector addition and $\otimes$ as tensor product.

Tensor-based compositionality has been widely adopted for its power in capturing multi-dimensional aspects of entities in linguistics [GS11, SHP11, SPW13, KS14, MKS14, FPC15]. Tensor-based compositionality strategies also hold multiple state-of-the-art performance records in different applications, including sentiment analysis [SPW13], and text similarity [FPC15]. One limitation of tensor-based representation is its high space complexity; consequently low-rank tensor modeling has also been studied as a solution very recently [FPC15, YDA16].

Another approach for compositionality is direct computation of the representation of meaning. CCG is usually employed as the linguistic formalism for the problem [HB13]. Our proposed semantic binder follows this approach.

5.3 The Proposed Semantic Binder – SEBI

We introduce SEBI — a semantic binder for compositionality in this section. SEBI is an attempt to model linguistic compositionality under the Montague formalism using CCG. Specifically, we formalize semantic computation for a sequence of words in two distinct tasks: 1) computing the true representation of words with strong polysemy; and 2) suggesting syntactic ways of combining them. We base this syntactic combination on combinatory categorial grammar. We begin this section by presenting a generic framework for learning *semantic binder representations* — binder embeddings that are used to combine word embeddings into sequence embeddings — for compositionality in section 5.3.1. We then describe the two components of the binder in 5.3.2 and 5.3.3 respectively. Finally, we discuss different structural representations for sequences and their alignment strategy.

5.3.1 Binder Embeddings for Compositionality

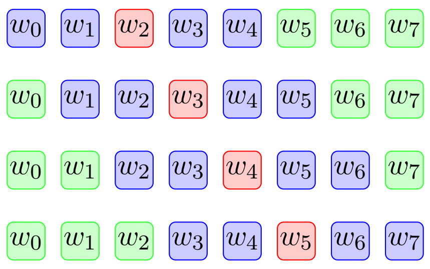
Compositionality in linguistics seeks to explain the compositional meaning of an expression of multiple words. In computational terms, it concerns the semantic representation of a sequence of words and the ability to compute, compose, and compare representations.

The word is a linguistic unit that constitutes the content of an expression. The task of learning representations for words, or word embeddings, has been studied extensively in the recent literature. Most methods, including `word2vec`, base their learning strategy on the distributional hypothesis of Harris and Firth [Har54, Fir57], stating that "linguistic items with similar distributions have similar meanings." The learned word embeddings are shown to capture a wide range of linguistic regularities when trained with a large dataset.

We extend the application of the distributional hypothesis to phrases — to learn representationas for phrases. Specifically, we consider a word sequence as one linguistic unit and learn its representation based on its contextual words, similar to the word embedding training. In addition, we compute the representation for a word sequence by a function combining its individual word embeddings — using semantic binders. This is motivated by the Montague formalism presented in section 5.2. To the best of our knowledge, no work has considered this binder-based strategy to compute embeddings for multi-word expressions using the distributional hypothesis. The main reason is probably that the number of phrases grows exponentially in the phrase length [MSC13], a result of the curse of dimensionality in training for N-gram language models [SCA13].

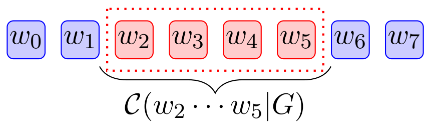
Figure 5.2 shows the distinction in training between word embeddings and multi-word sequence embeddings. In word embedding training, shown in Figure 5.2.a, (word,context)-

tuples are extracted directly from the text. Similarly, we also use context words to compute the expected phrase embedding. The tuples (phrase,context) depicted in Figure 5.2.b are the input. Unlike word embedding training, however, we do not train an embedding for every instance of a phrase due to the exponential growth. Instead, we build a model to compose individual word embeddings using our proposed semantic binders.



w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7
w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7
w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7
w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7

(a) (word,context) tuples for word embedding trainings



w_0 w_1 w_2 w_3 w_4 w_5 w_6 w_7
$\mathcal{C}(w_2 \cdots w_5|G)$

(b) (phrase,context) tuples for training compositionality

Figure 5.2: Input for training representation for words and phrases

Formally, let $\mathcal{C}$ be the compositionality model and $s = w_0w_1 \ldots w_n$ be an ordered sequence of words (usually a phrase or sentence) in the training dataset. The representation for a sequence of words in s, from i to j, under a parse G is denoted by $\mathcal{C}(w_i \cdots w_j|G)$. In Figure 5.2.b, for example, $\mathcal{C}(w_2 \cdots w_5|G)$ denotes the representation of phrase $w_2w_3w_4w_5$ under a parse G. Estimating the representation of phrase $w_2w_3w_4w_5$ is not straightforward. A naive approach would be to concatenate or average embeddings of words of the phrase. However, doing so would overlook two important aspects in language:

- Polysemy is a major issue in language processing, and the same word can play different semantic roles in different contextual settings. Therefore, having a fixed representation for a word, even a polysemous word, is not adequate. We address this problem in compositionality in the next section (Section 5.3.2).

- A sentence can have different parses for different emphases. Therefore, syntactic contributions of the same word can also be different given different parses. Most previous works agree on this observation and propose special treatment for certain syntactic roles in the sentence. For example, verbs are believed to play a more important role in marrying subject and object, and thus are given higher weight when concatenating. However, multiple parses become a problem when we deal with regular sentences of varied lengths. We present our strategy for this problem in Section 5.3.3

We address both issues in separate components due to the fact that the two are independent linguistically.

5.3.2 Word Sense Estimation (WSE)

In order to generate a good compositional representation of an expression, identifying the correct concepts or senses of individual words is crucial. The problem is usually related to polysemy in linguistics.

Polysemy has been studied extensively in the literature. Strategies often focus on discrimination using syntactic and semantic information. In neural word embeddings, a straightforward approach for polysemy is to build a multi-faceted embedding or multiple embeddings for a word. Many different strategies have been reported [EP08, RM10b, HSM12, TDB14, NSP14, CXH15, VP16a]. We employ our K-embedding for words, proposed in Chapter 4, due to its simplicity in training multiple representations for words without special annotation. The technique assumes a maximum of K different concepts for each word [VP16a].

Our proposed word sense estimation is based on the observation that the meaning of a word should be in accordance with its context, as stated in the distributional hypothesis. This suggests that representation of a word given a context can be computed as a mixture of its *related* (with respect to the context) conceptual embeddings. Specifically, let k be a conservative limit on the number of related concepts, k $(1 \leq k \leq K)$. For example, $k = 1$ means we only consider the best related conceptual embedding. If $k \geq 1$, we compute a mixture representation by averaging all related conceptual embeddings.

Formally, let $A(w_i|c)$ be the estimated representation for word w_i given context c, $c = \{w_{i-W+j} : j \in [0..2W], j \neq W\}$ with context window W. The representation of word w_i in context c is computed as:

$$A\left(w_i|c\right) = \frac{\sum_{j=1}^{k} \left\{ A(w_i^{(j)}) : A(w_i^{(j)}) \odot A\left(c^{(j)}\right) \geq T \right\}}{k} \tag{5.1}$$

where, $A(w_i^{(j)})$ and $A(c^{(j)})$ are the embeddings of word w_i and context c given concept j respectively. $A(w_i^{(j)})$ is derived directly from our proposed K-embeddings in Chapter 4. In general, $A(c^{(j)})$ can be computed using different methods. Here we introduce two possible suggestions:

- **Averaging Embedding (AE)** is a straightforward way to estimate representation for a bag of words or, in this scenario, the context words c of the same concept:

$$A(c) = \frac{\sum_{w \in c} A(w)}{|c|} \tag{5.2}$$

- **Attention-based Composition (AC)** is a new effective strategy to compute context embedding [BCB14]. The context representation is computed as a weighted sum of context words:

$$A(c) = \sum_{w \in c} \alpha_{iw} A(w) \tag{5.3}$$

  where α_{iw} is the weight that denotes the relatedness between the word w_i and a context word $w \in c$ computed by:

$$\alpha_{iw} = \frac{\exp(e_{iw})}{\sum_{v \in c} \exp(e_{iv})} \tag{5.4}$$

5.3.3 Semantic Binder Training

Word embedding training computes a vectorial representation for each word. However, modeling the relations between words as they are combined for compositional sequence is not straightforward. Previous work has aimed to address the problem by using multi-dimensional tensors to capture all possible *mutual* interactions between words. This requires a large number of parameters to train and represent these relations. In addition, representing them is even more challenging when dealing with open text.

We instead model the word composition functionally using a lambda calculus simulation to compute the interactions between words directly. We base our computation on the combinatory categorical grammar (CCG) formalism, which offers a unified model using syntactic and semantic information for meaning representation. Generally, we share similar motivations with [HB13], believing that the compositional power of recursive neural networks should benefit the CCG formalism in linguistics. However, our strategy is different, using three separate representation models that describe a word by (1) itself, (2) a function (functor), and (3) an argument for a function (functee).

Specifically, categorial-based compositionality is, as described, computed via a binary tree in which we merge consecutive words and phrases through combinatory rules. We base our composition training on CCG parsing of a large-scale dataset.

In CCG, a composition is of form `(LeftPhrase, RightPhrase, Rule)`. There are multiple types of rules, including functional application and functional composition rules. We view these rules as functions. For example, a rule `Rule` combining `LeftPhrase` and `RightPhrase` from right to left is seen as a function corresponding to `RightPhrase` that is applied on `LeftPhrase`. Thus, `RightPhrase` then contributes its functor embedding and `LeftPhrase` contributes its functee embedding in the composition.

For example, the (phrase,context) tuples in Figure 5.2 are converted into tuples of four components (LC, LP, RP, RC), where LP and RP are the left and right parts of the phrase before composing; and LC and RC are context words to the left and right of the phrase $LP \cdot RP$, respectively. Figure 5.3 shows a tuple from the example in Figure 5.1. The solid lines are functors and dotted ones are functees; and each will have a representation given a word.

Our objective function for training is similar to that in `word2vec`. Given a combinatory rule of a sentence and a tuple (LC, LP, RP, RC), assuming the application direction (arrow)

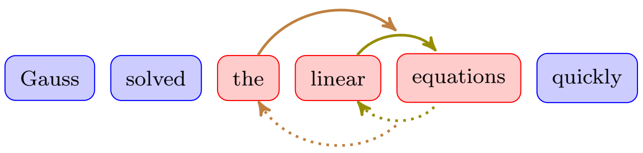


Figure 5.3: An example showing how words are composed

are left to right (as in Figure 5.3), a functor of LP and functee of RP should be adjusted to fit in the given context. Formally, let $F(P)$ and $D(P)$ be the functor and functee representation of phrase P and $P \equiv LP \cdot RP$. In addition, let $A(w)$ be the embedding for word w. The composition embedding of P is defined as:

$$\mathcal{C}(P) = F(P) + D(P) + A(P) \tag{5.5}$$

There are multiple ways to compute $F(P)$, $D(P)$, and $A(P)$. In this chapter, we propose to do this by averaging as follows, assuming rules are applied in a forward (left-to-right) direction:

$$A(P) = \frac{\sum_{w \in P} A(w)}{|P|} \tag{5.6}$$

$$F(P) = \frac{\sum_{w \in LP} F(w)}{|LP|} \tag{5.7}$$

$$D(P) = \frac{\sum_{w \in RP} D(w)}{|RP|} \tag{5.8}$$

There are three sets of embeddings to train in our proposal: A, F, and D. First, the embeddings of words A can be obtained using any standard pre-trained word embeddings, including `word2vec`. Second, training for functor embedding F and functee embedding D is straightforward — when A is fixed, thanks to the proposed formula (5.5). Specifically, we rely on an optimization technique that is similar to one used in `word2vec`, but instead the updating factor goes into F and D, split equally for each combinatory rule, as defined by the CCG parse. In other words, updates for F and D are the residues when training with (phrase,context)-tuples.

One reasonable concern is the convergence of the training. Even though convergence has not been proven mathematically, training has consistently converged in all of our experiments, including those described in Section 5.4. This can be explained intuitively by noting that, even though the functional interaction modeling sounds empirical, it is in fact driven by a converging grammar. Mathematical convergence derivations are presented in [Ron14].

5.4 Experiments

We analyze the proposed work in two different experimental scenarios: subject-verb-object (SVO) agreement (constrained) and sentence paraphrasing (unconstrained).

For consistency, we perform all the benchmarks using only one trained model for compositionality without special treatment for a particular evaluation set. On the one hand, this offers a sense of the practical performance of the reported results. On the other hand, this helps simplify the experimental configuration to allow direct interpretation of the results. The trained model still achieves competitive results in all comparisons.

5.4.1 Training Dataset

We trained our embedding using a snapshot of Wikipedia for English that was downloaded in 2014. It includes about 4.6 million articles, with roughly 2-billion tokens and around 10 million unique tokens. The text was segmented at the sentence and word levels using the NLTK toolkit. The text was also transformed into lowercase prior to training.

We annotated categorial tags for the dataset using the C&C tools and Boxer, developed by Clark and Curran and Bos [CCB07, Bos08]. Of all the 94,611,174 sentences in the dataset, 88,413,545 (roughly 93.5%) sentences could be parsed successfully in CCG.

We also filtered out words having fewer than 50 occurrences in the text. This resulted in having roughly 330,564 different unique words[1]. To put this in perspective, this is roughly equivalent to the size of the English lexicon. There are 272,858 entries in the Oxford English Dictionary (OED) project[2] in 2013, though this is not the reason we set the minimum count to be 50.

Regarding the training of embeddings, we set the dimension to be 200. We considered a window size $|W| = 5$ for the context words, meaning 5 words to the left and 5 words to the right. Optimization was performed using a negative sampling technique. The number of negative samples was set to 10. Finally, we considered K=10 in K-embedding training for word sense estimation.

5.4.2 Subject-Verb-Object Agreement

We first present the evaluation in two standard SVO agreement benchmarks: transitive verb disambiguation GS11 [GS11], and sentence similarity KS14 [KS14]. We consider these constrained evaluations because the input is formatted in subject-verb-object (SVO) triples.

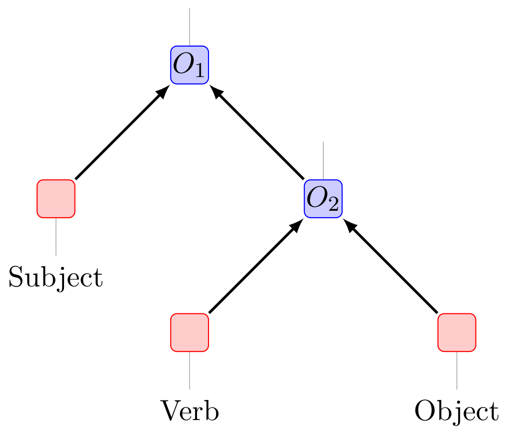
GS11 aims to evaluate disambiguation of different meanings of a verb in a given context. For example, *meet* and *satisfy* will share the same meaning in the (student–requirement) context, which is scored 6.77 by a human; but will not share it with *visit*, scored 1.46.

Similarly, KS14 focuses on evaluating the composition of SVO, when two sentences (technically, two SVO tuples) have similar meanings. For example, *meet* in (commodity–requirement) does not share the same meaning as *win* in the (force–battle) context, scored 1.32; but might largely share similar meaning with *satisfy* for (product–demand), scored 5.0. Figure 5.4 shows a CCG parse for the standard S-V-O structure.

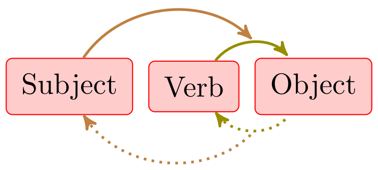
Table 5.1 reports the performance of our model, comparing with recent state-of-the-art results. The evaluation metric is Spearman's correlation, measuring similarity of rankings between human annotators and our model. The score for a tuple is the averaged score for all human ratings. Even though they are based on the same testing sets, the results in this

[1]we also performed evaluation with full text without frequency filtering; however, the performance differences were not significant.

[2]`public.oed.com`



(a) A CCG parse tree for S-V-O



(b) functional interactions between S-V-O tuple

Figure 5.4: A CCG parse for S-V-O disambiguation tasks

table are not necessarily comparable. This is because the training data used in different reports are of different types, and also are pre-processed differently. As described, we assume no advance knowledge about the SVO tasks, and train the model using the text by itself, without performing extra normalization (e.g. stemming) or preprocessing (e.g., extracting only related SVO tuples).

The results show that SEBI is effective in improving the composition beyond baseline models – using only word embedding addition for single verbs (Verbs Only), or for the entire SVO (SVO Addition using A). “Normalized” refers to the half deduction of F and D in the computation. The result also suggests that an attention mechanism can effectively compute context embedding.

Improvements were recorded for KS14, when we considered the effect of an additional semantic binder $\mathcal{C}$ in the Addition models. However, a slight drop on GS11 was recorded

Model	**GS11**	**KS14**
Verbs Only	0.155	0.363
SVO Addition using A	0.177	0.409
SVO Addition using $\mathcal{C}$ ($\equiv$ A + F + D)	0.173	0.645
Normalized SVO Addition using $\mathcal{C}$	0.173	0.669
SVO Addition using $\mathcal{C}$ + WSE_{AE}	0.362	0.670
Normalized SVO Addition using $\mathcal{C}$ + WSE_{AE}	**0.398**	**0.675**
Normalized SVO Addition using $\mathcal{C}$ + WSE_{AM}	**0.449**	**0.689**
Kartsaklis et al. [KS14] Addition	0.234	0.581
Kartsaklis et al. [KS14] Frobenius Tensor	0.412	0.332
Hashimoto et al. [HSM14] Addition	0.290	N/A
Hashimoto et al. [HSM14] Supervised	0.420	N/A
Milajevs et al. [MKS14] Addition	0.149	**0.689**
Milajevs et al. [MKS14] Copy Object	0.456	0.655
Milajevs et al. [MKS14] Frobenius Tensor	0.375	0.622
Fried et al. [FPC15] Addition	0.130	0.560
Fried et al. [FPC15] Full-rank Tensor (d=1M)	**0.460**	0.680
Human Agreement	0.750	0.660

Table 5.1: SVO Agreement Evaluation

instead (even though it is insignificant). This drop was predictable, since GS11 yielded poor results without disambiguation power, because the context words of both inputs are the same, Performance on GS11 was significantly elevated as WSE was introduced, making it closer to the state of the art results, although the computation was still using a 200-dimensional space.

In general, we can clearly see the impact of each component of our SEBI framework in both tasks. While they appear to be similar, the targeted linguistic problems are indeed distinct.

Finally, it is worth mentioning that, even though our results do not exceed the best reported performance, we can achieve reasonable stability and consistency when dealing with different tasks — using one single framework. This is indeed the fundamental difference between our evaluation and previous reported results.

5.4.3 Sentence Paraphrasing

We analyzed the performance of SEBI in an unconstrained setting in this section. Unlike the previous task in Section 5.4.2, when sentences were normalized in SVO tuples, we dealt with regular sentences of varied structures in this experiment. Specifically, we evaluated our compositionality in a paraphrase detection task. (A pair of compositional representations of two sentences are considered a paraphase if their embeddings' cosine similarity is larger than a certain threshold.)

We use the paraphrasing identification evaluation from the *Microsoft Research Paraphase Corpus* [DB05]. The sentences are first CCG parsed, and representations are computed using the method discussed earlier. The training set and testing set consist of 4076 and 1725 sentence pairs.

Table 5.2 shows the results and our comparisons with the current state of the art of unsupervised [U] and supervised [S] methods. The results show that our performance is within a range that is competitive with the current state of the art for unsupervised methods. It is again worth noting that the same trained compositionality model, including "Normalized Addition" models, is used for evaluation of both the constrained and unconstrained tasks.

5.5 Conclusions and Future Work

In this chapter, we have proposed a semantic binder, SEBI, for compositionality in language. In particular, we propose to model compositionality by 1) extending the distributional hypothesis of words to phrases, in order to train compositional embeddings; and 2) computing representations for phrases and sentences using the CCG formalism.

The experimental results show that our proposed SEBI approach has achieved competitive performance in different tasks, ranging from constrained disambiguation and text similarity to unconstrained sentence paraphrasing. The results support our idea of modeling

Model	**Accuracy**	**F1**
All True	0.664	0.798
Addition using A	0.700	0.798
Addition using $\mathcal{C}$	0.711	0.807
Normalized Addition using $\mathcal{C}$ + WSE_{AE}	0.713	0.811
Normalized Addition using $\mathcal{C}$ + WSE_{AM}	**0.719**	**0.816**
[U] Fernando et al. [FS08]	**0.741**	**0.824**
[U] Islam et al. [II09]	0.726	0.813
[U] Hassan [Has11]	0.725	0.814
[U] Milajevs et al. [MKS14]	0.730	0.820
[S] Das et al. (2009)	0.761	0.827
[S] Socher et al. [SHP11]	0.768	0.836
[S] Ji et al. (2013)	0.804	0.859
[S] He et al. [HGL15]	0.786	0.853

Table 5.2: MSPD Paraphrasing Detection Evaluation

the functional interaction of words. In addition, our proposed compositionality framework is generic and portable. Many extensions can be explored further, both in the way ambiguity is handled and compositional representation is computed.

Finally, the differences in performance gain between subject-verb-object agreement versus sentence paraphrasing can be explained by limitations in the way arbitrary lengths were handled in our proposed compositionality model. In the next chapter, we address this limitation by studying Gated Unit Recurrent Neural Networks for compositionality.

CHAPTER 6

Constituent-based Compositionality

In this chapter, we introduce a novel three-layer framework to learn compositionality representations for sentences of arbitrary length. Specifically, we focus on three factors that define compositionality: (1) in-context representation for words, (2) semantic cohesion in constituents, and (3) constituent-based propagation in the recurrent neural network (RtNN) defined by the combinatory rules of CCG. In particular, we propose our generalization of the gated recurrent unit to handle binary-tree-structured input and apply it both to LSTM and to GRU, the two most popular forms of RtNN, for language compositionality. We evaluate our approach with tasks in sentiment analysis.

6.1 Introduction

Compositionality has long been a principal interest of linguistics. Like the development of deep learning research, the problem has also received great attention from researchers for major NLP applications, including sentiment analysis [SPW13], neural machine translation [BCB14], sentence paraphrasing [ZSG16], and so on. These tasks have been considered hard, because they require in-depth understanding and representation of linguistic expressions.

In these tasks, a compositionality model typically projects a linguistic expression of arbitrary length into a fixed high dimensional representation using different neural network architectures. Strategies for this problem, despite the variety for each particular application, can be factored in two aspects: the mathematical structure and the construction of the representation.

First, an expression can be represented in different mathematical forms, from vectors to matrices and even high rank tensors. Theoretically, tensors of higher rank give better representation in terms of expressiveness [FPC15, SPW13]. However, this strategy is exponentially inefficient in space, and the performance gain of using high rank tensors is not always well supported in practice.

Second, the computation of the representation is complicated due to the complexity of linguistics and the constraints in representation forms. Many deep learning architectures have been explored to produce good representation, including the recursive neural network, convolutional neural network, and recurrent neural network. Among all, the long-short term memory (LSTM) model, a variant of recurrent neural network, is the most popular learning method since it is capable of *blending* — remembering while forgetting — information from words. LSTM currently holds multiple state-of-the-art performance records related to compositionality. Recently, the Gated Recurrent Unit (GRU), a simplified variant of LSTM, has become favored since it is more computationally efficient and manages to outperform LSTM on multiple tasks [JZS15].

Both standard LSTM and GRU share two limitations: (1) low interpretability: different ordering of the propagation results in very different performance, and (2) unit-wise sequential propagation: the models process an expression word by word, and thus overlook the constituency aspect of compositionality in linguistics.

In this chapter, we address both limitations in a three-layer framework to compute compositionality representation. In particular, we first propose to compute an in-context representation for words using our K-Embeddings presented in Chapter 4. To our knowledge, this is the first attempt to address this problem for compositionality. Second, we propose to compute the representation using the guidance of combinatory rules, which suggest how individual words of a sentence should be read bottom up semantically, from the CCG syntax tree. This directs the propagation to compute the representation by growing the interpretable meaning of an expression. Finally, we incorporate the semantic binder we developed in Chapter 5, thus improving cohesion of constituents.

In the next section, we present some background from the recent literature on compositionality. We then describe our proposed constituent-based recurrent neural networks in detail. Specifically, we explain our modified recurrent unit and propagation based on combinatory rules, as well as our strategy to cluster semantic constituents. The following section presents our overall compositionality framework. Experiments are discussed after that and followed by our conclusion.

6.2 Background

Compositionality plays a key role in multiple language processing applications, including tasks on sentiment analysis [SPW13], neural machine translation [BCB14], sentence paraphrasing [ZSG16], and so on. Among all, machine translation (MT) has been considered the pinnacle application in NLP and also in AI [SW71]. In the literature, the phrase-based translation system (PBMT), a complex system combining multiple sophisticated linguistic and statistic features, has defined the state of the art for more than a decade [KOM03]. This sophistication in fact explains why compositionality is important and is hard. Practically, it is widely understood that even *reasonably sound* features usually fail to result in better translations. The feature engineering for PBMT itself is a difficult task [OGK04, CKW09]. Therefore, MT is considered an "AI-complete" problem, meaning that solving it would make it possible to solve all other AI problems [Sha92].

Since the growing adoption of DL, a new architecture for MT — neural machine translation (NMT) — has been used to build a single large neural network to translate sentences from a source language into a target language [CMG14, SVL14, BCB14]. In 2015, NMT outperformed PBMT and set a new state of the art [BCB14, LSL15, LM16]. As of October 2016, Google Translate switched its online translation engine to this new architecture, and named it Google Neural Machine Translation (GNMT) [WSC16].

This achievement depended in particular on two mechanisms related to compositionality: a memory mechanism, modeled by a recurrent neural network to blend information, and an attention mechanism, constructed with special consideration of the intermediate embeddings.

6.2.1 Memory Mechanism using Recurrent Neural Network

In linguistics, two basic types of information of a language are the words and the syntactic structures that combine these words into expressions. Languages are usually modeled in terms of contiguous sequences (such as n-grams) or syntactic tree structures. Therefore, in modeling for NLP applications, recursive neural networks (RsNN) and recurrent neural networks (RtNN) are the two most popular DL architectures because they handle well the input described in trees and sequences respectively. In practice, however, modeling with trees usually introduces significant complications, such as that the number of trees on n nodes grows as $O(2^n)$. Therefore, the RtNN has become more popular since it is simpler and can handle sequences of arbitrary length.

The RtNN is comprised of multiple similar recurrent units, in which the output of one unit is the input of another unit. Typically, the RtNN processes inputs in a certain direction, for example from left to right. For an RtNN to succeed, the recurrent unit has to be capable of *blending* information properly, and thus RtNNs are usually sophisticated. The long-short term memory unit (LSTM) and the gated recurrent unit (GRU) are currently the most widely-used.

Two important components in gated RtNN modeling are the recurrent units and the propagation of the input. We introduce our improved gated RtNN and describe both of these components in Section 6.3.

6.2.2 Attention Mechanism

An attention mechanism was first introduced for translation tasks by Bahdanau et al. [BCB14]. The key idea, in the translation context, is to preserve the intermediate embeddings as well as the final compositional representation to do informed target words translation. To some extent, this is a practical attempt to try to control the sequential composition of embeddings since different — and uninterpretable — performance results have been reported. In other words, this strategy helps to alleviate the memorization pressure on one single final representation by keeping all intermediate snapshots after each word of the source sentence is consumed — instead of just relying on the final output.

The attention mechanism was immediately applied to a wide range of NLP applications. For example:

- machine translation [LPM15, EHT16, FCB16, CHV16, YHD16]
- sentence embedding [WHF16, SLC16]
- logical form conversion [DL16]
- text understanding [KSB16, WLZ16]
- relation classification [LTA15, LSL16]
- summarization [RCW15]
- word embedding [LTA15]

In this chapter, we instead propose to compute the composition of embeddings based on the combinatory rules of CCG. In addition, we introduce improvements on the RtNN in three different stages of the computation. These proposed changes, even though they differ from the attention mechanism, solve the similar problem of low interpretability in RtNN.

6.3 Constituent-based Recurrent Neural Networks

Beside the gated mechanism in the recurrent units, gated recurrent neural networks have been successful in modeling languages thanks mostly to the simple sequential propagation. Yet, this is also one of its major limitations since varied performances are recorded when changing even just the propagation direction. In other words, the model lacks the capability to emphasize aspects as it propagates. Tree-LSTM is an attempt to address this limitation [TSM15]. However, the efficiency of the Tree-LSTM is not consistently supported. We hypothesize that this is because general tree structure is too complicated to control. In addition, we empirically found that the suggested handling of the tree branches in [TSM15] is too sophisticated to train effectively.

We introduce instead a novel constituent-based RtNN architecture for compositionality. Intuitively, we impose the constituent structure of the input expression on propagation in the RtNN. this guides the RtNN to process the input This guides the RtNN to process the input in the same way that the meaning of the expression is constructed, which is linguistically described by the CCG formalism.

For example, the meaning of the sentence in Figure 5.1 can be read as: "linear equation" $\rightarrow$ "the linear equation" $\rightarrow$ "solved the linear equation" $\rightarrow$ "solved the linear equation quickly" $\rightarrow$ "Gauss solved the linear equation quickly". The constituent structure, as described by the combinatory rules of the CCG, can be represented as a binary tree. We describe the recurrent unit and the propagation of our proposed RtNN in Section 6.3.1 and Section 6.3.2 respectively.

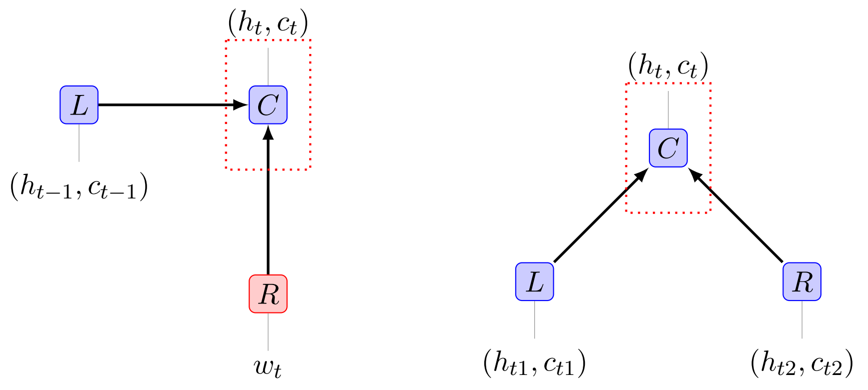
In addition, we propose to cluster words of an expression into semantic constituents (words or phrases) and perform a layered (tree-based) constituent-based propagation to produce compositionalities at different granularities. In other words, the strategy aims to control the depth of the constituent structures by sequentializing parts of the tree that are beyond a limit. We introduce the strategy in Section 6.3.3.

6.3.1 Combinatory Layered Recurrent Unit

As presented, we employ combinatory rules in combinatory categorial grammar parsing to drive the network propagation of a RtNN in semantic constituents. Each rule combines two constituents, words or phrases, into one larger constituent linguistically and directionally. Specifically, a combinatory rule can be one of the following types: forward application (fa), backward application (ba), forward composition (fc), backward composition (bc), forward crossing composition (fcc), and backward crossing composition (bcc). Therefore, the first step is to normalize the direction of the input, consisting of two constituent representations, into a left-unit L and right-unit R. Hence, we call the recurrent unit receiving these two inputs the "LR unit".

Unlike in standard sequential propagation, our LR recurrent unit could be one of the two types:

- sequential unit: at least one input is a word, as in Figure 6.1.a. Standard sequential RtNNs are a special case of our constituent-based RtNNs, in which all nodes are of this type.
- compositional unit: merges two constituents into one, as in Figure 6.1.b. These units create constituent layers in the propagation.



(a) sequential recurrent unit (b) combinatory recurrent unit

Figure 6.1: The two types of recurrent units.

We normalize both types of LR units into one. Specifically, we compute the initial state for each word to standardize the input for each LR unit, assuming that the node has no previous input. Below we present our normalization methods for LSTM and GRU; however, a similar strategy can be considered for other types of RtNN.

- For LSTM, the initial (h_w, c_w) for a word w is computed as:

$$c_w = \sigma(W^{(i)} x_w + b^{(i)}) \odot \tanh(W^{(u)} x_w + b^{(u)}) \tag{6.1}$$

$$h_w = \sigma(W^{(o)} x_w + b^{(o)}) \odot \tanh(c_w) \tag{6.2}$$

- For GRU, the initial (h_w) for a word w is computed as:

$$h_w = \sigma(W^{(z)} x_w + b^{(z)}) \odot \tanh(W x_w + b) \tag{6.3}$$

where x_w is the embedding of the word w. As a preprocessing step, all input embeddings of words are normalized by the computations.

6.3.2 Layered Constituent-based Propagation

In this section, we introduce our proposed generalization for RtNN, and the adaptation for LSTM and GRU, to allow propagation driven by combinatory rules. As introduced, a combinatory rule describes one combination operation, also known as an propagation from time t to $t+1$. Each propagation takes in two sets of input vectors and compute a compositional set through a computation graph of transition functions. For LSTM and GRU, the input vectors are tuples of type (h, c) and (h) respectively. Figure 6.2 is an example of propagation over the combinatory rules in Figure 5.1.

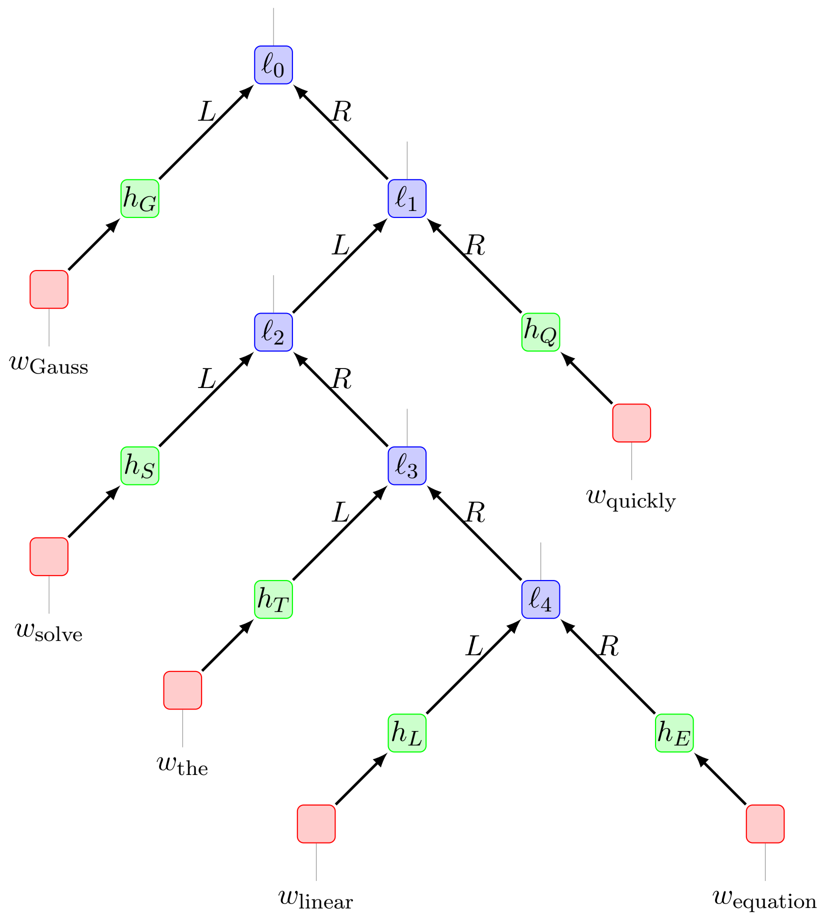


Figure 6.2: An example of propagation by combinatory rules

Formally, let F be the set of transition equations that compute the output for each propagation. In standard RtNN, F usually computes the output at time t as $(h_t, c_t) = F(h_{t-1}, c_{t-1}, x_t)$. In our setting, F is standardized as $(h_{LR}, c_{LR}) = F[(h_L, c_L), (h_R, c_R)]$, where L and R are the left and right constituents of sequence LR. Below are our adaptations of F for LSTM and GRU.

- Transition equations for constituent-based LSTM:

$$f_L = \sigma(U_L^{(f)} h_L + b^{(f)}) \tag{6.4}$$

$$f_R = \sigma(U_R^{(f)} h_R + b^{(f)}) \tag{6.5}$$

$$i_{LR} = \sigma(U_L^{(i)} h_L + U_R^{(i)} h_R + b^{(i)}) \tag{6.6}$$

$$o_{LR} = \sigma(U_L^{(o)} h_L + U_R^{(o)} h_R + b^{(o)}) \tag{6.7}$$

$$u_{LR} = \tanh(U_L^{(u)} h_L + U_R^{(u)} h_R + b^{(u)}) \tag{6.8}$$

$$c_{LR} = f_L \odot c_L + f_R \odot c_R + i_{LR} \odot u_{LR} \tag{6.9}$$

$$h_{LR} = o_{LR} \odot \tanh(c_{LR}) \tag{6.10}$$

- Transition equations for constituent-based GRU:

$$z_{LR} = \sigma(U_L^{(z)} h_L + U_R^{(z)} h_R + b^{(z)}) \tag{6.11}$$

$$r_{LR} = \sigma(U_L^{(r)} h_L + U_R^{(r)} h_R + b^{(r)}) \tag{6.12}$$

$$o_{LR} = \tanh(U_L^{(o)} (r_{LR} \odot h_L) + h_R + b^{(o)}) \tag{6.13}$$

$$h_{LR} = z_{LR} \odot (h_L + h_R) + (1 - z_{LR}) \odot o_{LR} \tag{6.14}$$

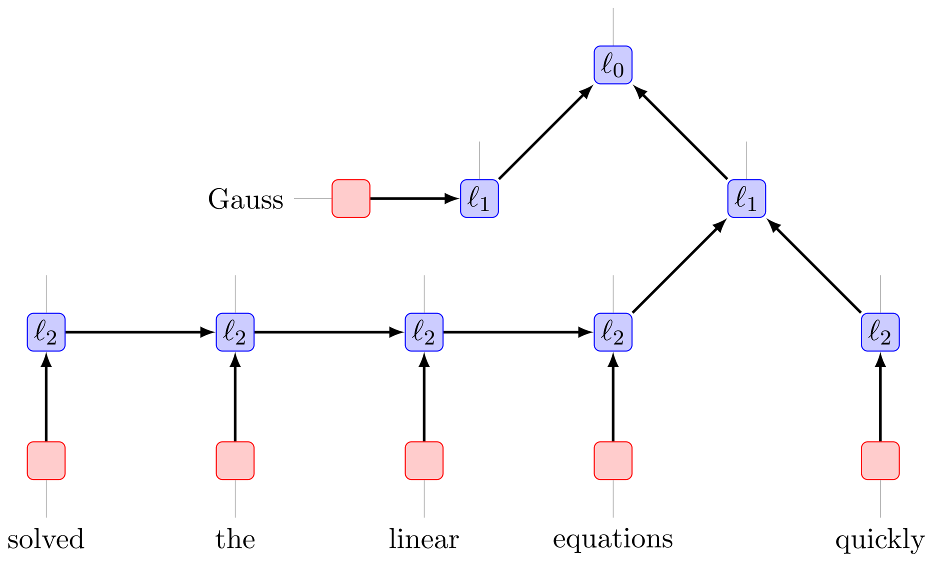
It is interesting to note that, even though our transition equations appear to be even simpler than the original models (having fewer parameters to estimate), they have proven very effective empirically.

6.3.3 Semantic Constituent Clustering

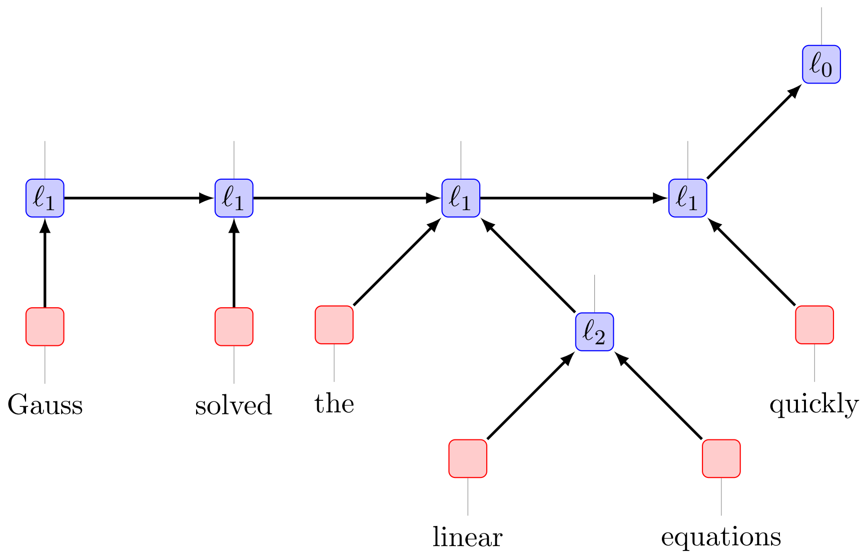
Sequential RtNN has been proven effective for a wide range of compositionality-related applications in NLP. We believe that the simplicity of the sequential model is important for its robustness. Our proposed CCG-based RtNN also considers simplicity as one key factor.

Dealing with regular tree structures is difficult, especially with inputs of arbitrary length. In this section, we propose extra processing to reduce the complexity of the proposed RtNN — by simplifying the input. Specifically, we introduce two methods to *linguistically* adjust the granularity of the constituent-tree structure created by combinatory rules by sequentializing either the sub-constituents (Tree of Sequential Constituents, ToSC) or the composition of the sub-constituents (Sequence of Tree-based Constituents, SoTC).

Examples of ToSC and SoTC for the example in Figure 6.2 are presented in Figure 6.3.



(a) Tree of Sequential Constituents

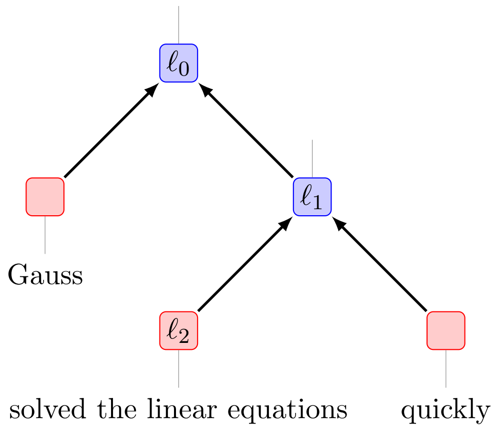


(b) Sequence of Tree-based Constituents

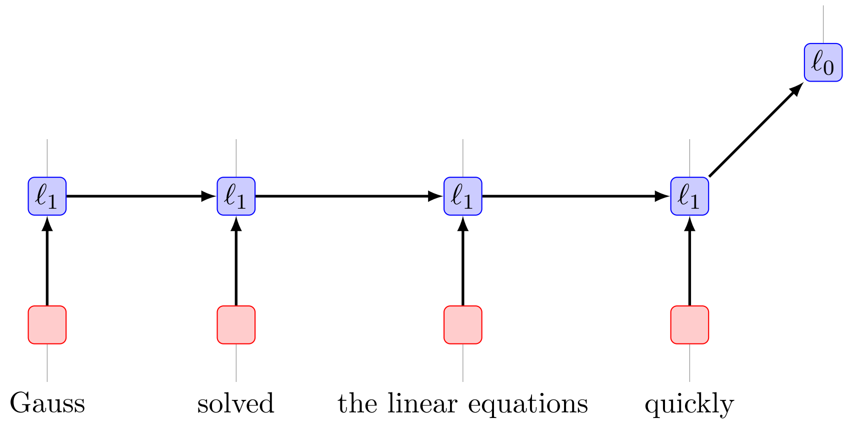
Figure 6.3: ToSC (a) and SoTC (b) examples for the CCG-tree in in Figure 6.2

Generating ToSC and SoTC for a CCG-tree is straightforward. As studied in Chapter 5, combinatory categorial grammar offers a strong linguistic interpretation of semantic composition defined by a set of combinatory rules. The rules explain the connection between syntactic and semantic representations in a constituency-based structure. Therefore, the formalism is already of type *phrase structure grammar*, which makes the task of generating ToSC and SoTC easy. Our strategy is to split the tree vertically in two halves, and flatten (sequentialize) one half for either ToSC or SoTC.

Let ℓ_d be the level having a distance of d combinatory rules from the root node. To construct ToSC at ℓ_d, we reserve the tree up to distance d and flatten nodes of the tree from depth d. In other word, at a node p_d at level ℓ_d, words derived from p_d are grouped into one (sequential) constituent — making a tree of constituents having depth d. To construct SoTC, we instead reserve the tree structure of the constituent and flatten all nodes up to level ℓ_d. Figure 6.4 displays the input after converting to ToSC and SoTC at layer ℓ_3.



(a) Simplified Visualization of Tree of Sequential Constituents at ℓ_3



(b) Simplified Visualization of Sequence of Tree-based Constituents at ℓ_3

Figure 6.4: An example of combinatory rules described using a binary tree.

One of the key properties of ToSC and SoTC is fixed-depth structure for any tree, which is especially useful in dealing with non-traditional text.

6.4 Constituent-based Representation — CORE

In this section, we introduce our proposed three-layer constituent-based compositionality framework, namely CORE. Besides the constituent-based propagation mechanism introduced in Section 6.3, we focus on two other factors that define compositionality:

- in-context representations for words
- semantic binding of constituents.

6.4.1 In-context Representation for Words

Polysemy is a one of the main problems in modeling language. For example, the token "bank" can have different meanings in a financial context or scenery context. Also, there are related issues: "Chinese", for example, can have multiple informational interpretations in the context of history or of dialect. Thus, the representation of a word should not be fixed as an embedding using a pre-trained model. We have shown in Chapter 4 that we can compute contextualized representations effectively using our K-embeddings technique. In addition, our work in Chapter 5 also has shown the effectiveness of K-embeddings in verb disambiguation tasks.

In our CORE framework, we propose to incorporate the result in Chapter 4 via an ensemble strategy. Specifically, we compute the representation for word w in a given context by a softmax classifier on the averaged embedding c of its context words. Formally, let $\mathbf{e}(w|c)$ be the expected representation of word w in context c. $\mathbf{e}(w|c)$ is computed by combining embeddings of word w using a base word embedding ($V(w)$) and the conceptual K-embedding:

$$\mathbf{e}(w|c) = V(w) + \text{softmax}(c \odot W_k + b_k) \odot K(w) \tag{6.15}$$

where:

- c is the averaged word embeddings of context words of w
- $W_k \in \mathbb{R}^{d \times k}$ and $b_k \in \mathbb{R}^k$ are classifier parameters
- $K(w) \in \mathbb{R}^{k \times d}$ are all possible K-embeddings of w
- $V(w) \in \mathbb{R}^d$ returns the base embedding of w.

6.4.2 Semantic Cohesion in Constituents

Besides the representation of individual words, we also incorporate binder embeddings for words in a constituent. Unlike the word-based computation presented in Chapter 5, we build two additional compositionality representations of the input using the functor (F) and functee (D) embeddings. The final representation is the combination of the output from Section 6.4.1 (model A) and these two additional embeddings:

$$\mathbf{P}(h) = \text{softmax}(W_A h + W_F h + W_D h + b) \tag{6.16}$$

$$\mathbf{y}(h) = \text{argmax}(\mathbf{P}(h)). \tag{6.17}$$

Output of our RtNN is an embedding comprised of all *necessary* information from the expression. This embedding is passed to a softmax classifier for further classification tasks.

6.5 Experiments

We study the effectiveness of our proposed modeling of structural constituents in this section. Specifically, we consider the task of sentiment analysis of sentences extracted from movie reviews. We evaluate our proposed compositionality model on sentiment classification task using the Stanford Sentiment Treebank [SPW13]. This dataset has become a widely-used benchmark for research in embedding, and thus any significant performance improvements are viewed as important.

The dataset consists of 11,855 sentences with sentiment annotation. Each sentence is classified into one of five different classes: very negative (– –), negative (–), neutral (0), positive (+), very positive (+ +). The dataset is split into 8544/1101/2210 sentences for the training/development/testing sets.

6.5.1 Training Dataset

Our main training data is Wikipedia for English, downloaded on November, 2014. It consists of 4.6 million articles, with a total of 2 billion words. using the NLTK toolkit. The dataset was pre-processed with sentence segmentation and word tokenization using NLTK toolkit. We converted text to lower-case prior to training. The same training data was used to build our K-embeddings and semantic binder, as described in Chapter 4 and Chapter 5. We annotated categorial tags on the dataset using the C&C tools and Boxer, developed by Clark and Curran and Bos [CCB07, Bos08]. Of all the 94,611,174 sentences in the dataset, 88,413,545 (roughly 93.5%) sentences could be parsed successfully in CCG.

As an attempt at fair comparison, however, we employ the pre-trained word embeddings from Google[1], trained using 100 billion words from Google News. Table 6.1 shows the baseline performances for LSTM and GRU using word embeddings trained by our Wikipedia snapshot and the Google News word embeddings.

Model	**2-billion Wikipedia**	**100-billion Google News**
LSTM	40.99	45.6
GRU	42.52	45.3

Table 6.1: Comparison between word vector usages

We also filtered out words having fewer than 50 occurrences in the text. This resulted in having roughly 330,564 different unique words[2]. To put this in perspective, this is roughly equivalent to the size of English lexicon. There are 272,858 entries in Oxford English Dictionary (OED) project[3] in 2013, though this is not the reason we set the minimum count to be

[1] https://code.google.com/archive/p/word2vec/

[2] we also performed evaluation with full text without frequency filtering; however, the performance differences were not significant.

[3] public.oed.com

50.

Regarding the training of embeddings, we set the size of the dimension d to be 300 (we use d=200 for previous Chapters) to keep it compatible with Google News word embeddings. We choose $K = 10$ for our word K-embeddings.

6.5.2 Baseline

We present in Table 6.2 results reported for this task in the literature. It should be noted that, due to the performance gap created by different configuration and randomization seeds, our LSTM baselines are at least 0.8 points behind that of [TSM15]. One possible explanation would be that our results are based on different word embeddings (they use Glove word embeddings trained from 840-billion words of Common Crawl data). We set our NumPy randomization seed to be 123 in all the experiments in this paper for reproducibility.

Model	**Accuracy**
NB [SPW13]	41.0
SVM [SPW13]	40.7
BiNB [SPW13]	41.9
VecAvg [SPW13]	32.7
RsNN [SPW13]	43.2
MV-RsNN [SPW13]	44.4
RsNTN [SPW13]	45.7
DCNN [KGB14]	48.5
Paragraph-Vec [LM14]	48.7
CNN-non-static [Kim14]	48.0
CNN-multichannel [Kim14]	47.4
DRNN [IC14]	49.8
LSTM [TSM15]	46.4
Bidirectional LSTM [TSM15]	49.1
2-layer LSTM [TSM15]	46.0
2-layer Bidirectional LSTM [TSM15]	48.5
Dependency Tree-LSTM [TSM15]	48.4
Stanford PCFG Tree-LSTM [TSM15]	49.7
LSTM [Our Implementation]	45.6
Bidirectional LSTM [Our Implementation]	46.5
2-layer LSTM [Our Implementation]	46.6
2-layer Bidirectional LSTM [Our Implementation]	47.7

Table 6.2: Baselines and Reported State-of-the-art Results

6.5.3 Concept-oriented RtNN with K-Embeddings

We present the results of our proposed RtNN using in-context representation of words (see Section 6.4.1) in Table 6.3. Because words have the ability to obtain different representations in our compositionality framework, our model shows consistent improvements in almost all comparisons. In general, GRU also outperforms LSTM by significant margins.

Model	**LSTM**	**GRU**
Standard	45.6	45.3
Bidirectional	46.5	46.0
2-layer	46.6	44.1
2-layer Bidirectional	47.7	47.6
Standard + K-Embeddings	45.7	47.4
Bidirectional + K-Embeddings	47.4	48.3
2-layer + K-Embeddings	47.4	46.7
2-layer Bidirectional + K-Embeddings	46.7	48.1

Table 6.3: Baselines and Our Proposed In-context Representation of Words in RtNN

The result suggests yet another application of our proposed K-embeddings for compositionality and proves its effectiveness. This supports the idea that K-embeddings can be used as an extra semantic layer on top of a well-trained word embedding. The improvement is consistent with different variants of the recurrent models.

6.5.4 RtNN Augmented by Semantic Binder

Table 6.4 reports performance measures using the trained semantic binder embeddings from Chapter 5 as an extra supporting embedding.

Model	**LSTM**	**GRU**
Standard	45.6	45.3
Bidirectional	46.5	46.0
2-layer	46.6	44.1
2-layer Bidirectional	47.7	47.6
Standard + Semantic Binder Embeddings	46.1	46.0
Bidirectional + Semantic Binder Embeddings	46.3	46.5
2-layer + Semantic Binder Embeddings	46.8	45.9
2-layer Bidirectional + Semantic Binder Embeddings	47.4	48.0

Table 6.4: Baselines and Our Proposed Augmentation of Semantic Cohesion in Constituents in RtNN

The results also report consistent gains on top of the baselines. However, the gains are smaller compared to those reported in Table 6.3. This emphasizes the fact that disambiguation is necessary for this problem, yet it receives inadequate attention in the community.

6.5.5 RtNN with Propagation driven by Combinatory Rules

We present the results of our constituent-based RtNN in Table 6.5. We also show the performance of SoTC and ToSC at different settings of ℓ, where $\ell = 1$ implies that there is little to no change to the original constituent structure, and ℓ is at its maximum when the entire structure is totally sequentialized, meaning the input sequence is appended monotonically left to right.

First, as a sanity check, the performance of sequentialized constituent-based RtNN should be approximately equivalent to the standard modeling of RtNN. The result in Table 6.5 also indicates this, showing the performance gaps (between the first and last lines) are no larger than 0.2%. This also supports the idea that our implementations of the proposed constituent-based RtNN are robust.

Second, the performance when using full CCG constituent structure is consistently better than that of the sequentialized structures, up to almost 2%. It also reaches the performance of a more complicated 2-layer bidirectional RtNN, which requires 4 iterations on the input to compute the final embedding. The result also confirms again that the proper propagation is beneficial not only in performance gains, but also in interpretability.

Third, and most interestingly, the layer-wise results at multiple ℓ settings reveal that the optimal structure for our constituent-based RtNN might not necessarily be the complete constituent structure. As we have hypothesized, the complexity of the tree structures might hurt convergence in training. In other words, even though constituent structure can produce much better performance than sequential structure, it should be normalized and simplified for better training.

Model	**LSTM**	**GRU**
Standard	45.6	45.3
Bidirectional	46.5	46.0
2-layer	46.6	44.1
2-layer Bidirectional	47.7	47.6
Full CCG Constituent-based RtNN	**47.6**	46.5
SoTC ; ToSC at $\ell = 1$	**47.6** ; **47.6**	**46.4** ; **46.4**
SoTC ; ToSC at $\ell = 2$	46.9 ; **47.6**	46.1 ; 46.0
SoTC ; ToSC at $\ell = 3$	46.0 ; 46.7	45.7 ; 45.2
SoTC ; ToSC at $\ell = 4$	45.9 ; 45.4	45.3 ; 44.8
SoTC ; ToSC at $\ell = 5$	**47.1** ; 45.6	44.8 ; 44.3
SoTC ; ToSC at $\ell = 6$	45.1 ; **47.0**	43.8 ; 44.0
SoTC ; ToSC at $\ell = 7$	**47.1** ; **47.1**	**46.4** ; 43.3
SoTC ; ToSC at $\ell = 8$	46.0 ; **47.5**	**47.1** ; 45.2
SoTC ; ToSC at $\ell = 9$	46.2 ; 45.6	**47.1** ; 45.2
Sequentialized Constituent-based RtNN	45.7	45.1

Table 6.5: Baselines and Our Proposed Constituent-based RtNN

6.5.6 Three-Layer Constituent-based RtNN

Our previous experimental studies were trained separately as suggested by the pre-training strategy for deep neural network architectures. In this section, we combine our best results at each layer for the final three-layer model.

Model	**LSTM**	**GRU**
Standard	45.6	45.3
Bidirectional	46.5	46.0
2-layer	46.6	44.1
2-layer Bidirectional	47.7	47.6
Standard + KEM + SEBI	46.6	47.3
Bidirectional + KEM + SEBI	47.7	48.5
2-layer + KEM + SEBI	47.6	47.1
2-layer Bidirectional + KEM + SEBI	48.1	48.6
Full CCG Constituent-based RtNN + KEM + SEBI	48.6	49.1

Table 6.6: Comparisons

6.6 Conclusion

In this chapter, we continued to investigate the compositionality representation problem for linguistic expressions. Specifically, we leveraged recent advances in recurrent neural network modeling for compositionality. After studying these models, we proposed three aspects that could be exercised for better compositionality. First, we introduced an integration of our proposed K-embeddings to build in-context representations for words. Second, we also introduced our binder representations to help with computation of embedding for each constituent. Finally, and most importantly, we introduced our constituent-based recurrent neural network framework to compute compositional representations. We have shown the effectiveness of our proposed strategy on the widely-studied sentiment analysis benchmark, and succeeded in obtaining significant performance gains. Our proposed method, yet again, is simple and flexible, and can be integrated with other available systems for better performance. Indeed, the three components we have studied in this chapter are independent, and can be easily combined in other NLP applications for performance gains.

CHAPTER 7

Conclusion

7.1 Thesis Summary

This dissertation addresses several basic questions in Data Mining (DM) and Natural Language Processing (NLP). Specifically, it presents my attempts to bridge recent advances in Artificial Intelligence (AI) and Machine Learning (ML), with recently-developed theories in Linguistics, in order to solve problems in DM and NLP.

For DM, I have empirically shown that representations of nodes (node embeddings) are useful not only in mining node relations in social network analysis, but also in uncovering novel structure in the network. In other words, node embeddings can be a beneficial source of information in mining.

For NLP, I have proposed a unified framework for linguistic compositionality. In particular, I have shown that compositionality can benefit from being handled differently than with traditional DNN approaches — by focusing on managing linguistic phenomena. Specifically, our proposed three-layer compositionality model has been proven effective by addressing polysemy, functional interactions, and constituent structure. Even more surprisingly, these phenomena can be well-handled, not by using expensive annotated datasets, but by exploiting linguistic features — and structures — using DNN techniques with redundant large scale (unannotated) text dataset. The experimental results strongly support this approach, and emphasize the benefits of integrating theory and practice in multidisciplinary research.

7.2 Beyond Deep Learning

The recent advances in machine learning suggests that deep neural networks (DNNs) are a potential candidate for current and future challenges in large scale data science. Descriptively, the DNN framework has proven its maturity in the popular *five Vs of Big Data*:

- **Volume** refers to the amount of available data. It has been demonstrated that DNN frameworks can handle large influxes of data. In addition, the learned models can be easily updated thanks to the incremental training and updating. Training can thus be scaled up with the current parallelism capability and the recent advances in GPU development.

- **Velocity** refers to the instant updating capability with new data. This goes in line with the Volume challenge, and is highly feasible with recent developments in hardware and algorithms. This also underscores the necessity of communication and collaboration between researchers in both fields.

- **Variety** concerns the various sources and types of data and information being generated. However, this aspect has not been receiving sufficient attention in the research community. In most studies, experiments rely on a standardized dataset to prove some particular hypothesis. In fact, the Variety problem is closely related to compositionality, one of my focuses in this dissertation. Compositionality is a grand challenge, still very open, and a central problem of Linguistics. It is worthwhile for researchers to pay more attention to the great potential importance of this area.

- **Veracity** refers to the fidelity of models. For DNN frameworks, it might raise questions about the weak interpretability, which is a concern since in this aspect kernel methods excel. Interestingly, this dissertation, especially works in Chapter 6, has shown that an interpretable DNN is possible if we can employ structural information in the learning process properly. The results also encourage more interdisciplinary research.

- **Value** refers to the final outcome that the findings can bring to daily life. Even though this is generally not easily quantifiable, we can see the increasing pervasiveness of AI in our daily encounters, from mobile to medical.

Deep learning has been proven effective when applied to problems in different disciplines, while in many cases achieving state-of-the-art performance. In this thesis, I have demonstrated its potential for working in many popular applications. Specifically, I have focused on the mutual benefit of combining neural networks for learning innovative representations, and advances in Linguistics, NLP, and DM.

7.3 Conclusion

Modeling *intelligence* is hard, and will remain hard for another decade or two. The results in this dissertation underscore the necessity of interdisciplinary communication and integration. Indeed, we already see trends towards interdiscipinary approaches on a daily basis, from news to the academic literature. I believe this trend will continue in the future, and bring great benefits.

References

[Ajd35] K. Ajdukiewicz. “Die syntaktische Konnexität.” *Stud. Philos.*, **1**:1–27, 1935.

[AK14] Jacob Andreas and Dan Klein. “How much do word embeddings encode about syntax?” In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 822–827, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate.” *CoRR*, **abs/1409.0473**, 2014.

[BGL08] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. “Fast unfolding of community hierarchies in large networks.” *CoRR*, **abs/0803.0476**, 2008.

[BH14] Samy Bengio and Georg Heigold. “Word Embeddings for Speech Recognition.” In *Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech*, 2014.

[BOK07] Alexandra Birch, Miles Osborne, and Philipp Koehn. “CCG Supertags in Factored Statistical Machine Translation.” In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT ’07, pp. 9–16, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[Bos08] Johan Bos. “Wide-coverage Semantic Analysis with Boxer.” In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP ’08, pp. 277–286, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[CC04] Stephen Clark and James R. Curran. “Parsing the WSJ Using CCG and Log-linear Models.” In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL ’04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[CC07] Stephen Clark and James R. Curran. “Wide-coverage Efficient Statistical Parsing with Ccg and Log-linear Models.” *Comput. Linguist.*, **33**(4):493–552, December 2007.

[CCB07] James Curran, Stephen Clark, and Johan Bos. “Linguistically Motivated Large-Scale NLP with C&C and Boxer.” In *ACL 2007 Demo and Poster Sessions*, pp. 33–36, Prague, Czech Republic, June 2007.

[CHV16] Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. “Incorporating Structural Alignment Biases into an Attentional Neural Translation Model.” *CoRR*, **abs/1601.01085**, 2016.

[CKW09] David Chiang, Kevin Knight, and Wei Wang. "11,001 New Features for Statistical Machine Translation." In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 218–226, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[CMG14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[CSN09] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. "Power-Law Distributions in Empirical Data." *SIAM Rev.*, **51**(4):661–703, November 2009.

[CWB11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural Language Processing (Almost) from Scratch." *J. Mach. Learn. Res.*, **12**:2493–2537, November 2011.

[CXH15] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. "Improving Distributed Representation of Word Sense via WordNet Gloss Composition and Context Clustering." In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 15–20, Beijing, China, July 2015. Association for Computational Linguistics.

[CZZ15] Wenliang Chen, Min Zhang, and Yue Zhang. "Distributed Feature Representations for Dependency Parsing." *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, **23**(3):451–460, March 2015.

[DB05] William B. Dolan and Chris Brockett. "Automatically Constructing a Corpus of Sentential Paraphrases." In *IWP2005*. Asia Federation of Natural Language Processing, 2005.

[DL16] Li Dong and Mirella Lapata. "Language to Logical Form with Neural Attention." *CoRR*, **abs/1601.01280**, 2016.

[EHT16] Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. "Tree-to-Sequence Attentional Neural Machine Translation." *CoRR*, **abs/1603.06075**, 2016.

[EP08] Katrin Erk and Sebastian Padó. "A Structured Vector Space Model for Word Meaning in Context." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 897–906, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[FCB16] Orhan Firat, KyungHyun Cho, and Yoshua Bengio. “Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism.” *CoRR*, **abs/1601.01073**, 2016.

[FG] Elise Feingold and Peter Good. “ENCODE Pilot Project.” `http://www.genome.gov/26525202`.

[Fir57] J. Firth. *A Synopsis of Linguistic Theory 1930-1955*. Studies in Linguistic Analysis, Philological. Longman, 1957.

[FL09] Santo Fortunato and Andrea Lancichinetti. “Community Detection Algorithms: A Comparative Analysis: Invited Presentation, Extended Abstract.” In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS ’09, pp. 27:1–27:2, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[FPC15] Daniel Fried, Tamara Polajnar, and Stephen Clark. “Low-Rank Tensors for Verbs in Compositional Distributional Semantics.” In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 731–736, Beijing, China, July 2015. Association for Computational Linguistics.

[FS08] Samuel Fernando and Mark Stevenson. “A Semantic Similarity Approach to Paraphrase Detection.”, 2008.

[GBL10] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. “Approximation Analysis of Influence Spread in Social Networks.” *CoRR*, **abs/1008.2005**, 2010.

[Geh09] P.J. Gehrke. *The Ethics and Politics of Speech: Communication and Rhetoric in the Twentieth Century*. Southern Illinois University Press, 2009.

[GS11] Edward Grefenstette and Mehrnoosh Sadrzadeh. “Experimental Support for a Categorical Compositional Distributional Model of Meaning.” In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP ’11, pp. 1394–1404, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Han05] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[Har54] Zellig Harris. “Distributional Structure.” *Word*, **10**(23):146–162, 1954.

[Has11] Samer Hassan. *Measuring Semantic Relatedness Using Salient Encyclopedic Concepts*. PhD thesis, Denton, TX, USA, 2011. AAI3507009.

[HB13] Karl Moritz Hermann and Phil Blunsom. “The Role of Syntax in Vector Space Models of Compositional Semantics.” In *Proceedings of ACL*, August 2013.

[HCC14] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition." *CoRR*, **abs/1412.5567**, 2014.

[HGL15] Hua He, Kevin Gimpel, and Jimmy Lin. "Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks." In *EMNLP 2015*, pp. 1576–1586, Lisbon, Portugal, September 2015.

[Hoc03] Julia Hockenmaier. "Parsing with Generative Models of Predicate-Argument Structure." In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 359–366, Sapporo, Japan, July 2003. Association for Computational Linguistics.

[Hoc06] Julia Hockenmaier. "Creating a CCGbank and a Wide-coverage CCG Lexicon for German." In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pp. 505–512, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets." *Neural Comput.*, **18**(7):1527–1554, July 2006.

[HS02] Julia Hockenmaier and Mark Steedman. "Acquiring Compact Lexicalized Grammars from a Cleaner Treebank." In *LREC*. European Language Resources Association, 2002.

[HS07] Julia Hockenmaier and Mark Steedman. "CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank." *Comput. Linguist.*, **33**(3):355–396, September 2007.

[HSM12] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. "Improving Word Representations via Global Context and Multiple Word Prototypes." In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pp. 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[HSM14] Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. "Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1544–1555, Doha, Qatar, October 2014. Association for Computational Linguistics.

[HSW09] Hany Hassan, Khalil Sima'an, and Andy Way. "A Syntactified Direct Translation Model with Linear-time Decoding." In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pp. 1182–1191, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[IC14] Ozan Irsoy and Claire Cardie. "Opinion Mining with Deep Recurrent Neural Networks." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 720–728, Doha, Qatar, October 2014. Association for Computational Linguistics.

[II09] Aminul Islam and Diana Inkpen. "Semantic Similarity of Short Texts." *Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing*, 2009.

[IKC16] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. "Summarizing Source Code using a Neural Attention Model." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2073–2083, Berlin, Germany, August 2016. Association for Computational Linguistics.

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." *Journal of Machine Learning Research*, 2015.

[KCC08] Terry Koo, Xavier Carreras, and Michael Collins. "Simple Semi-supervised Dependency Parsing." In *Proceedings of ACL-08: HLT*, pp. 595–603. Association for Computational Linguistics, 2008.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A Convolutional Neural Network for Modelling Sentences." In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[Kim14] Yoon Kim. "Convolutional Neural Networks for Sentence Classification." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[KMK11] Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. "Recurrent Neural Network Based Language Modeling in Meeting Recognition." In *INTERSPEECH 2011*, pp. 2877–2880, 2011.

[KOM03] Philipp Koehn, Franz Josef Och, and Daniel Marcu. "Statistical Phrase-based Translation." In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pp. 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[KS03] E.L. Keenan and E.P. Stabler. *Bare Grammar: Lectures on Linguistic Invariants.* Center for the Study of Language and Information - Lecture Notes Series. CSLI Publications, 2003.

[KS14] Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. "A Study of Entanglement in a Categorical Framework of Natural Language." In *Proceedings of the 11th workshop on Quantum Physics and Logic, QPL 2014, Kyoto, Japan, 4-6th June 2014*, pp. 249–261, 2014.

[KSB16] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. "Text Understanding with the Attention Sum Reader Network." *CoRR*, **abs/1603.01547**, 2016.

[LG14a] Omer Levy and Yoav Goldberg. "Dependency-Based Word Embeddings." In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pp. 302–308. The Association for Computer Linguistics, 2014.

[LG14b] Omer Levy and Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization." In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pp. 2177–2185. Curran Associates, Inc., 2014.

[LM14] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." *CoRR*, **abs/1405.4053**, 2014.

[LM16] Minh-Thang Luong and Christopher D. Manning. "Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models." In *Association for Computational Linguistics (ACL)*, Berlin, Germany, August 2016.

[LPM15] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In *EMNLP 2015*, pp. 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[LSL14] Li-Jia Li, Hao Su, Yongwhan Lim, and Fei-Fei Li. "Object Bank: An Object-Level Image Representation for High-Level Visual Recognition." *International Journal of Computer Vision*, **107**(1):20–39, 2014.

[LSL15] Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. "Addressing the Rare Word Problem in Neural Machine Translation." In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 11–19, Beijing, China, July 2015. Association for Computational Linguistics.

[LSL16] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. "Neural Relation Extraction with Selective Attention over Instances." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2124–2133, Berlin, Germany, August 2016. Association for Computational Linguistics.

[LTA15] Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso, and Chu-Cheng Lin. "Not All Contexts Are Created Equal: Better Word Representations with Variable Attention." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1367–1372, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Man15] Christopher D. Manning. "Computational Linguistics and Deep Learning." *Computational Linguistics*, **41**(4):701–707, 2015.

[MCC13a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." *CoRR*, **abs/1301.3781**, 2013.

[MCC13b] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." *CoRR*, **abs/1301.3781**, 2013.

[MKS14] Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. "Evaluating Neural Word Representations in Tensor-Based Compositional Settings." In *EMNLP 2014*, pp. 708–719, Doha, Qatar, October 2014.

[MMR55] J. McCarthy, M. L. Minsky, N. Rochester, and C.E. Shannon. "A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE." http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html, 1955.

[MSC13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed Representations of Words and Phrases and their Compositionality." In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

[MYZ13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations." In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013.

[New03] M.E.J. Newman. "Fast algorithm for detecting community structure in networks." *Physical Review E*, **69**, September 2003.

[New04] M. E. J. Newman. "Analysis of weighted networks." *Phys. Rev. E*, **70**:056131, Nov 2004.

[New06] M. E. J. Newman. "Finding community structure in networks using the eigenvectors of matrices." *Physical review E*, **74**(3), 2006. cite arxiv:physics/0605087Comment: 22 pages, 8 figures, minor corrections in this version.

[NG04] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks." *Phys. Rev. E*, **69**(2):026113, February 2004.

[NSP14] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1059–1069, Doha, Qatar, October 2014. Association for Computational Linguistics.

[OGK04] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. "A Smorgasbord of Features for Statistical Machine Translation." In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pp. 161–168, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[OLC11] Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. "On Accuracy of Community Structure Discovery Algorithms." *CoRR*, **abs/1112.4134**, 2011.

[PGE08] Steven T. Piantadosi, N.D. Goodman, B.A. Ellis, and J.B. Tenenbaum. "A Bayesian model of the acquisition of compositional semantics." In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, 2008.

[PL05] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks (long version)." *Computer and Information Sciences-ISCIS 2005*, pp. 284–293, 2005. arXiv:arXiv:physics/0512106v1.

[PMG97] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence: A Logical Approach.* Oxford University Press, Oxford, UK, 1997.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation." In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[QCN14] Lin Qiu, Yong Cao, Zaiqing Nie, Yong Yu, and Yong Rui. "Learning Word Representation Considering Proximity and Ambiguity." 2014.

[RB08] Martin Rosvall and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure." *Proceedings of the National Academy of Sciences*, **105**(4):1118–1123, 2008.

[RCW15] Alexander M. Rush, Sumit Chopra, and Jason Weston. "A Neural Attention Model for Abstractive Sentence Summarization." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 379–389, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[RFP12] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. "Efficient Community Detection in Large Networks using Content and Links." *CoRR*, **abs/1212.0146**, 2012.

[RK14] Matteo Riondato and Evgenios M. Kornaropoulos. "Fast Approximation of Betweenness Centrality Through Sampling." In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pp. 413–422, New York, NY, USA, 2014. ACM.

[RM10a] Joseph Reisinger and Raymond J. Mooney. "Multi-prototype Vector-space Models of Word Meaning." In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pp. 109–117, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[RM10b] Joseph Reisinger and Raymond J. Mooney. "Multi-Prototype Vector-Space Models of Word Meaning." In *HLT-NAACL*, pp. 109–117. The Association for Computational Linguistics, 2010.

[RMR15] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. "Mining, Ranking and Recommending Entity Aspects." In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pp. 263–272, New York, NY, USA, 2015. ACM.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[Ron14] Xin Rong. "word2vec Parameter Learning Explained." *CoRR*, **abs/1411.2738**, 2014.

[SCA13] Abhinav Sethy, Stanley F. Chen, Ebru Arisoy, Bhuvana Ramabhadran, Kartik Audhkhasi, Shrikanth Narayanan, and Paul Vozila. "Joint training of interpolated exponential n-gram models." In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pp. 25–30, 2013.

[SH12] Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

[Sha92] Stuart C. Shapiro. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1992.

[SHP11] Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection." In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pp. 801–809. Curran Associates, Inc., 2011.

[SLC16] Paul Hongsuck Seo, Zhe Lin, Scott Cohen, Xiaohui Shen, and Bohyung Han. "Hierarchical Attention Networks." *CoRR*, **abs/1606.02393**, 2016.

[SPW13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

[Ste00] Mark Steedman. *The Syntactic Process.* MIT Press, Cambridge, MA, USA, 2000.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks." In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014.

[SW71] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* The University of Illinois Press, 1971.

[SYH09] Yizhou Sun, Yintao Yu, and Jiawei Han. "Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema." In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 797–806, New York, NY, USA, 2009. ACM.

[TDB14] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. "A Probabilistic Model for Learning Multi-Prototype Word Embeddings." In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 151–160, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[THP08] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. "Efficient Aggregation for Graph Summarization." In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 567–580, New York, NY, USA, 2008. ACM.

[TRB10] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. "Word Representations: A Simple and General Method for Semi-Supervised Learning." In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394. Association for Computational Linguistics, 2010.

[TSM15] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks." In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[Tur50] A. M. Turing. "Computing Machinery and Intelligence.", 1950. One of the most influential papers in the history of the cognitive sciences: http://cogsci.umn.edu/millennium/final.html.

[VP15] Thuy Vu and D. Stott Parker. "Node Embeddings in Social Network Analysis." In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pp. 326–329, New York, NY, USA, 2015. ACM.

[VP16a] Thuy Vu and D. Stott Parker. "K-Embeddings: Learning Conceptual Embeddings for Words using Context." In *NAACL 2016*, 2016.

[VP16b] Thuy Vu and D. Stott Parker. "Mining Community Structure with Node Embeddings." In *From Social Data Mining and Analysis to Prediction and Community Detection*, 2016.

[WCL12] Jonathan Weese, Chris Callison-Burch, and Adam Lopez. "Using Categorial Grammar to Label Translation Rules." In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pp. 222–231, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[WHF16] Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. "CSE: Conceptual Sentence Embeddings based on Attention Model." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[WLZ16] Bingning Wang, Kang Liu, and Jun Zhao. "Inner Attention based Recurrent Neural Networks for Answer Selection." In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1288–1297, Berlin, Germany, August 2016. Association for Computational Linguistics.

[WSC16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." *CoRR*, **abs/1609.08144**, 2016.

[YDA16] Mo Yu, Mark Dredze, Raman Arora, and Matthew R. Gormley. "Embedding Lexical Features via Low-rank Tensors." In *Proceedings of NAACL*, 2016.

[YHD16] Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alexander J. Smola. "Neural Machine Translation with Recurrent Attention Modeling." *CoRR*, **abs/1607.05108**, 2016.

[YJC09] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. “Combining Link and Content for Community Detection: A Discriminative Approach.” In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’09, pp. 927–936, New York, NY, USA, 2009. ACM.

[YYD16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. “Hierarchical Attention Networks for Document Classification.” In *NAACL 2016*, pp. 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

[YZD13] Mo Yu, Tiejun Zhao, Daxiang Dong, Hao Tian, and Dianhai Yu. “Compound Embedding Features for Semi-supervised Learning.” In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 563–568. Association for Computational Linguistics, 2013.

[ZC12] Luke S. Zettlemoyer and Michael Collins. “Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars.” *CoRR*, **abs/1207.1420**, 2012.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. “Graph Clustering Based on Structural/Attribute Similarities.” *Proc. VLDB Endow.*, **2**(1):718–729, August 2009.

[ZSG16] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. “DAG-Structured Long Short-Term Memory for Semantic Compositionality.” In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 917–926, San Diego, California, June 2016. Association for Computational Linguistics.