UNIVERSITY OF CALIFORNIA,
IRVINE

Introducing Gradient-Based Methods and Diagnostic Benchmarks to Trace Errors Back to the Training Data

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical Engineering and Computer Science

by

Pouya Pezeshkpour

2022

DEDICATION

In memory of my father, who will always be my hero.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Professor Sameer Singh, who went way above and beyond as an advisor to help me grow as a Ph.D. student. It is not an easy job to be a good advisor, let alone a caring one, but Sameer plays this role perfectly, constantly caring for the future of his students. Through my P.h.D program, he tremendously helped me to shape my research skills, my mindset, and most importantly, my vision, and for that, I will be forever grateful. I would love to thank him for the continuous support of my research and for his patience, motivation, enthusiasm, and immense knowledge. I could not have imagined having a better advisor for my P.h.D.

I would also love to thank my committee members, Professor Padhraic Smyth and Professor Zhou Li, for their helpful and valuable feedback on my dissertation. I am deeply honored to have such esteemed and knowledgeable professors as my committee members.

I was fortunate to spend amazing summers as a research intern in Fujitsu lab, Allen AI, Apple, and Microsoft, where I learned a lot and collaborated with the most talented people. In particular, I am grateful to my mentors, Doug Downey, Xiao Ling, and Ben Van Durme, for their endless patience, kindness, and guidance they provided me during these internships.

I want to thank all of my lab mates from the UCI NLP research group for their kindness, their help, and all the good memories during the last few years. In particular, I am especially grateful to my collaborators Zhengli Zhao, Yifan Tian, Liyan Chen, and Preethi Seshadri. I would like to thank my friends who were always there for me in my most difficult times Ali Tazarv, Arash Gholami, Behnam Pourghasemi, Cyrus Aria, Hessam Pirzadeh, Mehdi Ganji, Moin Aminnaseri, Navid Rezazadeh, and Sina Shahhosseini. I can only name a few here, and hopefully, my other friends can forgive me for not being able to mention their names.

Last but not least, I would like to thank my family: my parents and my brothers, for supporting me spiritually throughout my research studies and my life in general.

In my Ph.D., I have been supported in part by Allen Institute for Artificial Intelligence (AI2), in part by NSF grants #IIS-1817183, #IIS-1756023, #IIS-2040989, #IIS-2008956, in part by the DARPA MCS program under Contract No. N660011924033 with the United States Office of Naval Research, in part by the Air Force Research Laboratory (AFRL), and in part by a Ph.D. fellowship gift from NEC Laboratories.

VITA

Pouya Pezeshkpour

EDUCATION

Doctor of Philosophy in Electrical Engineering and Computer Science	**2022**
University of California, Irvine	*Irvine, CA*
Master of Science in Electrical Engineering and Computer Science	**2018**
University of California, Irvine	*Irvine, CA*
Bachelor of Science in Electrical Engineering	**2015**
Sharif University	*Iran*

INTERNSHIP EXPERIENCE

Research Intern	**2021**
Microsoft Research	*Redmond, WA*
Research Intern	**2020**
Apple	*Cupertino, CA*
Research Intern	**2019**
Allen Institute for AI	*Seattle, WA*
Research Intern	**2018**
Fujitsu Laboratories of America	*Sunnyvale, CA*

TEACHING EXPERIENCE

Teaching Assistant	**2017–2018**
University of California, Irvine	*Irvine, CA*

ACADEMIC REVIEWING

NeurIPS, NAACL	**2021**
NeurIPS, ICLR, AAAI, EMNLP	**2020**
NeurIPS, ICLR, EMNLP	**2019**
EMNLP	**2018**

REFEREED CONFERENCE PUBLICATIONS

Combining Feature and Instance Attribution to Detect Artifacts **2022**
Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron Wallace, ACL Findings [67]

An Empirical Comparison of Instance Attribution Methods for NLP **2021**
Pouya Pezeshkpour, Sarthak Jain, Byron Wallace, and Sameer Singh, NAACL [68]

Revisiting evaluation of knowledge base completion models **2020**
Pouya Pezeshkpour, Yifan Tian, and Sameer Singh, AKBC [71]

On the Utility of Active Instance Selection for Few-Shot Learning **2020**
Pouya Pezeshkpour, Zhengli Zhao, and Sameer Singh, HAMLETS at NeurIPS [72]

Using Data Importance for Effective Active Learning **2020**
Pouya Pezeshkpour, Zhengli Zhao, and Sameer Singh, VL3 at CVPR [73]

Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications **2019**
Pouya Pezeshkpour, Yifan Tian, and Sameer Singh, NAACL [70]

Integrating Local Structure into Knowledge Graph Embeddings **2019**
Pouya Pezeshkpour, Yifan Tian, and Sameer Singh, SoCal NLP [69]

Generating User-friendly Explanations for Loan Denials Using GANs **2018**
Ramya Srinivasan, Ajay Chander,and **Pouya Pezeshkpour**, CAIF at NeurIPS [91]

Embedding multimodal relational data for knowledge base completion **2018**
Pouya Pezeshkpour, Liyan Chen, and Sameer Singh, EMNLP [65]

Compact Factorization of Matrices Using Generalized Round-rank **2018**
Pouya Pezeshkpour, Carlos Guestrin, and Sameer Singh, SoCal ML [66]

REFEREED JOURNAL PUBLICATIONS

Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models **2022**
Aarohi Srivastava, et al, In Submission [92]

The Extremal GDoF Gain of Optimal versus Binary Power Control in K User Interference Networks Is $O(\sqrt{K})$ **2022**
Yao-Chia Chan, **Pouya Pezeshkpour**, Chunhua Geng, and Syed A. Jafar, ITWC [18]

ParsiNLU: A Suite of Language Understanding Challenges for Persian **2021**
Daniel Khashabi, et al, TACL [41]

ABSTRACT OF THE DISSERTATION

Introducing Gradient-Based Methods and Diagnostic Benchmarks to Trace Errors Back to the Training Data

By

Pouya Pezeshkpour

Doctor of Philosophy in Electrical Engineering and Computer Science

University of California, Irvine, 2022

Associate Professor Sameer Singh, Chair

Deep neural networks that dominate NLP rely on an immense amount of parameters and require large text corpora for training. As these models are increasingly being deployed in the real world, there is an accompanying need to characterize potential failure modes of such models to avoid harm. In particular, it is now widely appreciated that training such models over large corpora commonly introduces biases into model predictions and other undesirable behaviors. Moreover, many of the training datasets are collected automatically or via crowdsourcing, and exhibit systematic biases or annotation artifacts. Considering the current trend in NLP, which relies on ever-growing sizes of models and datasets, identifying the origin of an issue when we encounter a problematic behavior is becoming extremely challenging.

To recognize any issue, traditionally, one might rely on explaining the model prediction using a variety of attribution methods. Although these methods demonstrate tremendous success in explaining models' predictions, the fragile nature of their explanations hinders their adoption in practice. In addition to looking into models' inner mechanism to identify the source of any issue, one can also try to follow breadcrumb trails into the training process, which is generally very expensive and time-consuming.

In this dissertation, we turn toward training data to systematically identify the root of errors. After providing the necessary background in the second chapter, introducing a novel interpretability method for the knowledge graph completion (KGC) task, in the third chapter, we propose an automatic approach to extract mislabeled instances. In the fourth chapter, identifying several shortcomings in the current evaluation setting of KGC, we propose an alternative benchmark for the knowledge graph completion task and demonstrate the poor performance of state-of-the-art models in our benchmark. Shifting toward textual data, in the fifth chapter, we establish reliable and efficient instance attribution methods for explaining large language models. Finally, in the sixth chapter, we incorporate different attribution methods to discover existing artifacts in commonly used NLP benchmarks.

Chapter 1

Introduction

"If you gaze for long into an abyss, the abyss gazes also into you."

Friedrich Nietzsche, *Beyond Good and Evil*

In the past few years, deep neural models have achieved tremendous success in NLP and are increasingly being deployed in the real-world. A problem with such techniques is that they are opaque; it is not easy to know why models make specific predictions. Consequently, modern models may make predictions on the basis of attributes we would rather not (e.g., demographic categories or 'artifacts' in data).

In this dissertation, we intend to probe the following research question: why do deep models demonstrate certain problematic behaviors? To answer this question, we turn toward the training data as a potential source of errors in models' predictions. Our goal is to provide systematic guidelines, new benchmarks, and novel methods to facilitate the troubleshooting of deep models for a variety of tasks in NLP.

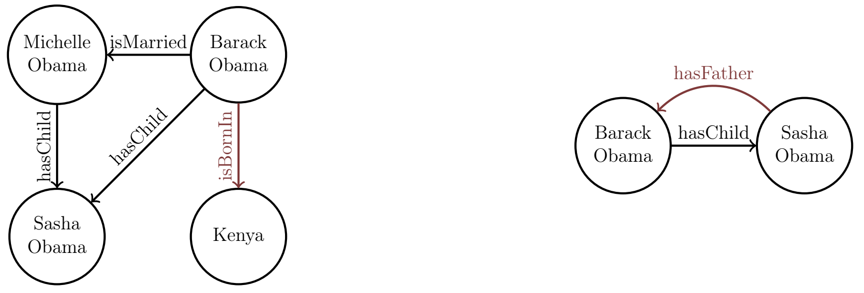
Surfacing issues by turning toward training data is hindered by the immense size of training data and deep models. Real-world knowledge graphs contain hundreds of millions of factual statements. Furthermore, current approaches that solve NLP tasks incorporate enormous models trained on huge corpora. As a result, to investigate the source of any issue, it is crucial to introduce efficient procedures. In this dissertation, to address this challenge, we first provide an efficient approximation for the impact of graph modifications on KGC models' predictions as a diagnostic tool for knowledge graphs related tasks. Then, by introducing a moderate size real-world aligned benchmark for the link prediction task, we facilitate the investigation of reasoning power for different link prediction models.

For textual data, we first resolve the Achilles' heels of instance attribution methods, i.e., computational complexity, by showing the high quality of simpler methods in explaining existing models and proposing efficient counterparts for instance attribution methods facilitating their adoption for enormous deep models. We then incorporate our findings on attribution methods to provide an efficient systematic guideline to diagnose problematic behaviors (artifacts) of models solving the textual tasks.

In this chapter, we first elaborate on the specific errors that we are tackling for each task. Then, we provide an outline for the remainder of the dissertation.

1.1 Types of Errors We Consider in This Work

As mentioned previously, our goal here is to dive deep into the training data to systematically identify the root of errors in models' predictions. We can divide the type of errors we are exploring in this dissertation into two categories: (1) errors in knowledge graphs and (2) errors in textual data.



(a) Incorrect Links (b) Existing Shortcuts

Figure 1.1: An overview of knowledge graph-related errors we consider in this work.

1.1.1 Errors Related to Knowledge Graphs

We consider two forms of issues in knowledge graphs and link prediction methods that cause problematic behaviors. An overview of these issues is provided in Figure 1.1.

Incorrect Links Since knowledge graphs are mostly created automatically from textual data, many incorrect links are being introduced into them. An example of this issue is provided in Figure 1.1a. These incorrect links can corrupt the link prediction method's representation of the graph. We address this form of issue in Chapter 3 by introducing automatic methods to identify these incorrect links.

Existing Shortcuts In addition to missing necessary and incorrect links in a graph, previous works [98, 25] observe that existing KGs suffer from data leakage due to the existence of inverse relations in the training data, which makes link prediction methods' performance untrustworthy. We extend previous findings into other forms of shortcuts in several knowledge graphs in Chapter 4 (an example of this issue is depicted in Figure 1.1b).

Task	Example	Artifact
NLI	**Premise**: The banker contacted the judge. **Hypothesis**: Hopefully, the banker contacted the judge.	Lexical Overlap
Toxicity Detection	Yo, I'mma very confused.	African-American Vernacular
Sentiment Analysis (IMDB)	... it was so boring. I give it - 8/10.	Ratings

Table 1.1: Examples of previously identified artifacts in a variety of NLP tasks.

1.1.2 Errors Related to Textual Data

Training deep neural networks requires large datasets. These are often collected automatically or via crowdsourcing, and may exhibit systematic biases or *annotation artifacts*. By the latter, we mean spurious correlations between inputs and outputs that do not represent a generally held causal relationship between features and classes; models that exploit such correlations may appear to perform a given task well but fail on out-of-sample data. Examples of artifacts in different NLP tasks are provided in Table 1.1. Upon reevaluating existing techniques for extracting influential training samples for a given prediction in Chapter 5, using our findings, we provide a systematic guideline for discovering artifacts in Chapter 6. In addition to artifacts, language models (LMs) have been shown to carry social biases that can amplify harmful stereotypes and discriminatory practices, which we briefly discuss in Chapter 7 as future directions.

1.2 Dissertation Statement

In this dissertation, we set out to explore whether gradient-based approaches and instance attribution methods are effectively capable of debugging/testing NLP models. More specifically, by providing systematic guidelines, new benchmarks, and novel methods, our goal here is to probe the following research questions:

In knowledge graphs we are interested in:

- Can we introduce and utilize interpretability methods for the link prediction task to automatically extract incorrect links in the graph?
- Can KGC methods' performances reliably measure their reasoning capability?
- Do KGC methods exploit shortcuts in graphs to make their predictions?
- Do current evaluation approaches in knowledge graph completion align with the real-world applications of these models?

For textual data, we tackle these questions:

- Are instance attribution methods' explanations trustworthy?
- Can we utilize simple similarity-based instance attribution instead of complex gradient-based methods?
- Can we use attribution methods to discover existing artifacts in the data?

Addressing these questions in this dissertation, we take the first step toward the real-world adoption of these models and open the doors for a variety of future research directions.

1.3 Dissertation Outline and Contributions

The general outline for the remainder of this dissertation is as follows:

Chapter 2 In this chapter, we provide the necessary background on the tasks and techniques we are considering throughout this work.

Chapter 3 To automatically identify incorrect links in KGs, in this chapter, we propose adversarial modifications for link prediction models: identifying the fact to add into or remove from the knowledge graph that changes the prediction for a target fact after the model is retrained. Using these single modifications of the graph, we identify the most influential fact for a predicted link and evaluate the sensitivity of the model to the addition of fake facts. We introduce an efficient approach to estimate the effect of such modifications by approximating the change in the embeddings when the knowledge graph changes. To avoid the combinatorial search over all possible facts, we train a network to *decode* embeddings to their corresponding graph components, allowing the use of gradient-based optimization to identify the adversarial modification. We use these techniques to evaluate the robustness of link prediction models (by measuring sensitivity to additional facts), study interpretability through the facts most responsible for predictions (by identifying the most influential neighbors), and detect incorrect facts in the knowledge base.

This chapter is based on the work: **Pouya Pezeshkpour**, Yifan Tian, and Sameer Singh. "Investigating robustness and interpretability of link prediction via adversarial modifications." NAACL 2019 [70].

Chapter 4 In this chapter, in order to investigate existing shortcuts in KGs, we first study the shortcomings of link prediction methods evaluation metrics. Specifically, we demonstrate that these metrics (1) are unreliable for estimating how calibrated the models are, (2) make strong assumptions that are often violated, and 3) do not sufficiently and consistently differentiate embedding methods from each other or from simpler approaches. To address these issues, we gather a semi-complete KG referred to as YAGO3-TC, using a random subgraph from the test and validation data of YAGO3-10, which enables us to compute accurate triple classification accuracy on this data. Conducting thorough experiments on existing models, we provide new insights and directions for the KG completion research.

This chapter is based on the work: **Pouya Pezeshkpour**, Yifan Tian, and Sameer Singh. "Revisiting evaluation of knowledge base completion models." AKBC 2020 [71].

Chapter 5 The widespread adoption of deep models has motivated a pressing need for approaches to interpret network outputs and facilitate model debugging. Influence functions provide machinery for doing this by retrieving training instances that (may have) led to a particular prediction. However, approximating the IF is computationally expensive to the degree that may be prohibitive in many cases. Might simpler approaches (e.g., retrieving train examples most similar to a given test point) perform comparably? In this chapter, we evaluate the degree to which different potential instance attribution agree with respect to the importance of training samples. We find that simple retrieval methods yield training instances that differ from those identified via gradient-based methods (such as IFs), but that nonetheless exhibit desirable characteristics similar to more complex attribution methods.

This chapter is based on the work: **Pouya Pezeshkpour**, Sarthak Jain, Byron Wallace, and Sameer Singh. "An Empirical Comparison of Instance Attribution Methods for NLP." NAACL 2021 [68].

Chapter 6 In this chapter, we evaluate the use of different *attribution* methods for aiding the identification of training data artifacts. We propose new hybrid approaches that combine *saliency maps* with *instance attribution* methods. We show that this proposed *training-feature attribution* can be used to efficiently uncover artifacts in training data when a challenging validation set is available. We also carry out a small user study to evaluate whether these methods are useful to NLP researchers in practice, with promising results.

This chapter is based on the work: **Pouya Pezeshkpour**, Sarthak Jain, Sameer Singh, and Byron C. Wallace. "Combining feature and instance attribution to detect artifacts." ACL Findings 2021 [67].

Chapter 7 Although language models (LMs) have been shown to carry social biases, as these models are being adopted to downstream tasks, despite recent efforts, it is not clear how bias transfers upon fine-tuning these models. We discuss that there are two fundamental issues with common practices hindering future progress: (1) creating bias benchmarks based on templates mostly results in out-of-distribution and ambiguous samples, and (2) current metrics do not capture biases/artifacts properly. In this chapter, concluding this dissertation, we propose solving these two problems as open directions for future research.

Chapter 2

Background

This dissertation can be divided into two parts: (1) investigating problematic behavior over knowledge graphs which includes chapters 3 and 4, and (2) exploring errors in deep models over textual data (chapters 5 and 6). In this section, we first provide the required background for knowledge graphs and text-related tasks, and then we review the adopted attribution methods in this dissertation.

2.1 Knowledge Graphs

Knowledge graphs (KGs) are essential components of a wide range of tasks in scientific and industrial processes such as search, structured data management, recommendations, and question answering. A knowledge graph is a representation of a knowledge base in the form of a graph that captures factual statements by linking entities (nodes) through different relations (edges).

Factual statements in a knowledge graph are represented using a triple of a subject, relation, and object, $\langle s, r, o \rangle$, where $s, o \in \xi$, a set of entities, and $r \in \mathcal{R}$, a set of relations. Respectively,

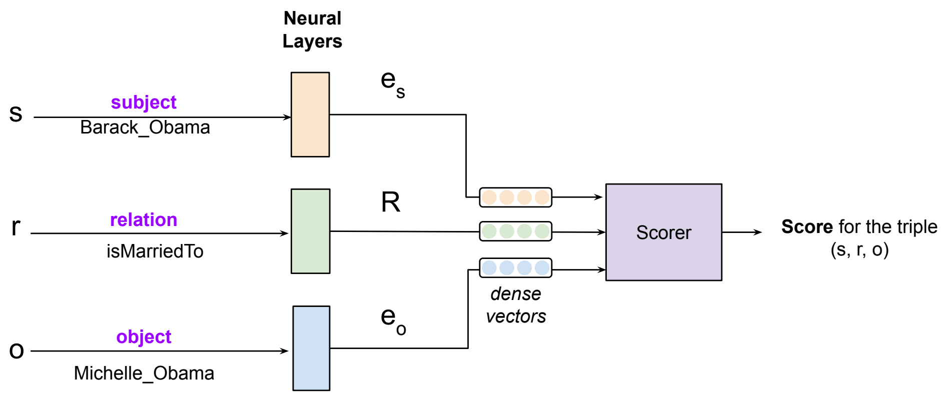


Figure 2.1: Example of embedding-based knowledge graph representation.

we consider two goals for relational modeling, (1) to train a machine learning model that can score the *truth* value of any factual statement, and (2) to predict missing links between the entities using their learned embeddings. In existing approaches, a scoring function $\psi : \xi \times \mathcal{R} \times \xi \rightarrow \mathbb{R}$ (or sometimes, $[0, 1]$) is learned to evaluate whether any given fact is true, as per the model. For predicting links between the entities, since the set ξ is small enough to be enumerated, missing links of the form $\langle s, r, ?\rangle$ are identified by enumerating all the objects and scoring the triples using ψ (i.e., assume the resulting entity comes from a known set).

Since KGs suffer from incompleteness, the *link prediction* task was introduced for the purpose of accurately inferring missing facts in the graph. Many of the recent advances in link prediction use embedding-based approaches. Embedding-based knowledge graph completion methods are built upon assigning a fixed-size embedding vector for each entity and relation in the KG; each entity in ξ and relation in $\mathcal{R}$ are assigned distinct, dense vectors, which are then used by ψ to compute the score.

2.1.1 Relational Embedding Methods

To learn the scoring function ψ, for each triple, we first need to map each one of the subject, object, and relation into the vector space. Recent approaches incorporate neural-based methods to provide vector embeddings for these components. An overview of embedding-based methods for the target triple ⟨Barack Obama, isMarriedTo, Michelle Obama⟩ is depicted in Figure 2.1. The major difference between previously proposed link perdition methods is in the way the scoring function is being defined. For example, DistMult scoring function [108] is defined as, $\psi(s, r, o) = \mathbf{e}_s \mathbf{R}_r \mathbf{e}_o$, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^d$ are embeddings of the subject, and object and $\mathbf{R}_r \in \mathbb{R}^{d \times d}$ is a diagonal matrix representing the relation r.

2.1.2 Negative Sampling and Loss Function

Since a knowledge graph only consists of positive samples (factual triples), to learn a scoring function, we need to fabricate negative samples for each true statement. To do so, existing methods mostly replace the object (or the subject) of a triple with multiple random entities from the graph. Since we chose these negative samples randomly, they are often trivial for the model to predict. As a result, several previous studies [16, 104] propose alternative methods to generate more challenging negative samples.

As for the loss function, most previous work either incorporate (1) a hinge loss [60]:

$$\mathcal{L}(G) = \sum_{i \in D_+} \sum_{j \in D_-} max(o, \gamma + \psi(\eta_j) - \psi(\eta_i)) \tag{2.1}$$

where D_+ and D_- denote the set of positive and negative samples, η captures triples, and

$\gamma > 0$ is the margin. Or (2) binary cross-entropy loss over triple scores [108, 101]:

$$\begin{aligned}\mathcal{L}(G) = \sum_{(s,r)\in G} \sum_{o} & y_o^{s,r} \log(\sigma(\psi(s,r,o))) \\ & +(1-y_o^{s,r}) \log(1-\sigma(\psi(s,r,o))). \end{aligned} \tag{2.2}$$

where $y_o^{s,r}$ represents negative and positive facts ($y_o^{s,r} = 1$ for observed facts, $y_o^{s,r} = 0$ otherwise).

2.2 Text-Related Tasks

In this dissertation, we consider three text-related tasks:

Text Classification is defined as categorizing a given document into a set of predefined labels. We consider two from of text classification here, (1) *sentiment analysis,* where the goal is to classify a text mostly as positive or negative. Examples of sentiment analysis are SST-2 benchmark [90]—consisting of single sentences from movie reviews—and IMDB reviews benchmark [50]. (2) *Toxicity detection,* defined as the task of effectively identifying hate speech. An example of this task that we consider in this work is the DWMW17 dataset [24] which is composed of 25K tweets labeled as *hate speech*, *offensive*, or *non-toxic.*

Natural Language Inference (NLI) or textual entailment is a task of identifying the relationship between fragments of text. Representing these fragments as premise and hypothesis, the goal here is to identify whether the premise *contradicts*, *entails*, or *is neutral toward* the hypothesis. An example of this task is provided in Table 1.1. We consider the Multi-Genre NLI (MNLI) dataset [106] for this task, which contains 393k pairs of premise and hypothesis from 10 different genres.

Question Answering (QA) is defined as automatically retrieving answers for questions extracted from a given document. In this dissertation, we only consider the BoolQ benchmark [20], a question answering dataset which contains 16k pairs of yes/no questions and corresponding passages (paragraphs from Wikipedia).

To solve these tasks, recent works proposed to utilize pre-trained language models [26, 48, 77, 76]—deep neural networks that are pre-trained over a large text corpus in an unsupervised manner. We can then finetune these models to solve any downstream task. In this work, we focus on BERT [26] introduced as a bidirectional transformer-based language model that learns contextual embeddings for words. BERT was pre-trained on two tasks: (1) masked tokens prediction and (2) next sentence prediction. To solve NLP tasks in this dissertation, we define a linear classification layer on top of BERT and finetune the whole model on downstream tasks by minimizing cross-entropy loss.

2.3 Attribution Methods

Attribution methods are a family of techniques introduced to explain the reasons behind any model's prediction. These methods not only can provide understanding and assess trust for deep models, but also can help with debugging and improving machine learning models. We consider two categories of attribution methods in this work: (1) *Feature Attributions* and (2) *Instance Attributions.*

Feature Attributions were introduced to highlight constituent input features (e.g., tokens) in proportion to their "importance" for the model prediction [80, 49, 1]. Examples of this category of attribution methods are *gradient-based* methods [82, 96] which explain model prediction by using output gradient with respect to the input.

Instance Attributions explain any model by retrieving training instances most responsible for a given prediction [44, 109, 78, 68]. An example of this category of attribution methods is *Influence Functions (IFs)*. Denoting model parameter estimates by $\hat{\theta}$, the IF approximates the effect that upweighting instance i by a small amount—ϵ_i—would have on the parameter estimates (here H is the Hessian of the loss function with respect to our parameters):

$$\frac{d\hat{\theta}}{d\epsilon_i} = -H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(x_i, y_i, \hat{\theta}) \tag{2.3}$$

This estimate can, in turn be used to derive the effect on a specific test point x_{test} :

$$\nabla_\theta \mathcal{L}(x_{\text{test}}, y_{\text{test}}, \hat{\theta})^T \cdot \frac{d\hat{\theta}}{d\epsilon_i} \tag{2.4}$$

Using this approximation, we can then sort the training instances based on their influence on the prediction of a test sample extracting the most influential ones.

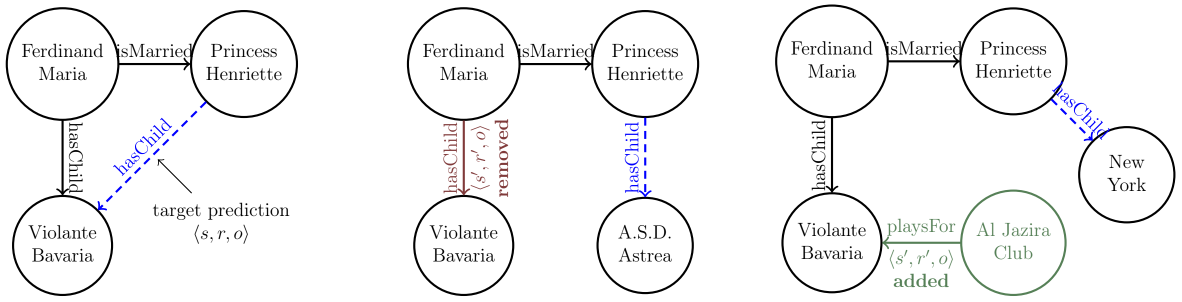
Chapter 3

Identifying Incorrect Information in Knowledge Bases by Explaining Completion Models

3.1 Introduction

As knowledge graph representation methods are increasingly being deployed in the real-world, it is crucial to be able to explain their prediction (for inducing trust) and characterize their potential failure modes. However, there are only a few studies [40, 87] that investigate the quality of the different KG models and introduce automatic methods to identify incorrect links. There is a need to go beyond just the accuracy on link prediction, and instead focus on whether these representations are robust and stable, and what facts they make use of for their predictions.

In this chapter, our goal is to design approaches that minimally change the graph structure such that the prediction of a target fact changes the most after *the embeddings are relearned*,



(a) KG, with the target (b) After removing a fact (c) After adding a fact

Figure 3.1: **Adversarial Modifications for Link Prediction (CRIAGE):** Change in the structure of the graph that results in the change in the prediction of the *retrained* model. (a) is the original sub-graph of our KG, (b) we remove an important link in the neighborhood of the target resulting in a change in the prediction of the target, and (c) shows the effect of adding an attack triple on the target. Provided modifications are generated by our approach.

which we call adversarial modifications on link prediction (CRIAGE). First, we consider perturbations that remove a neighboring link for the target fact, thus identifying the *most influential* related fact, providing an explanation for the model's prediction. As an example, consider the excerpt from a KG in Figure 3.1a with two observed facts, and a target *predicted fact* that `Princes Henriette` is the parent of `Violante Bavaria`. Our proposed graph's perturbation, shown in Figure 3.1b, identifies the existing fact that `Ferdinal Maria` is the father of `Violante Bavaria` as the one when removed and model retrained, will change the prediction of `Princes Henriette`'s child. We also study attacks that add a new, fake fact into the KG to evaluate the robustness and sensitivity of link prediction models to small additions to the graph. An example attack for the original graph in Figure 3.1a, is depicted in Figure 3.1c. Such perturbations to the training data are from a family of adversarial modifications that have been applied to other machine learning tasks, known as *poisoning* [11, 21, 10, 115].

Since the setting is quite different from traditional adversarial attacks, the search for link prediction adversaries brings up unique challenges. To find these minimal changes for a target link, we need to identify the fact that, when added into or removed from the graph, will have the biggest impact on the predicted score of the target fact. Unfortunately, computing this change in the score is expensive since it involves retraining the model to recompute the embeddings. We propose an efficient estimate of this score change by approximating the change in the embeddings using a Taylor expansion. The other challenge in identifying adversarial modifications for link prediction, especially when considering the addition of fake facts, is the combinatorial search space over possible facts, which is intractable to enumerate. We introduce an *inverter* of the original embedding model to *decode* the embeddings to their corresponding graph components, making the search for facts tractable by performing efficient gradient-based continuous optimization.

We evaluate our proposed methods through the following experiments. First, on relatively small KGs, we show that our approximations are accurate compared to the true change in the score. Second, we show that our additive attacks can effectively reduce the performance of state-of-the-art models [108, 25] up to 27.3% and 50.7% in Hits@1 for two large KGs WN18 and YAGO3-10. We also explore the utility of adversarial modifications in explaining the model predictions by presenting rule-like descriptions of the most influential neighbors. Finally, we use adversaries to detect errors in the KG, obtaining up to 55% accuracy in detecting errors.

3.2 Background

In this chapter, we focus on *multiplicative* models of link prediction[1], specifically DistMult [108] because of its simplicity and popularity, and ConvE [25] because of its high accuracy.

We use the same setup as [25] for training, i.e., incorporate binary cross-entropy loss over the triple scores. In particular, for subject-relation pairs (s, r) in the training data G, we use binary $y_o^{s,r}$ to represent negative and positive facts. Using the model's probability of truth as

[1]As opposed to *additive* models, such as TransE [12], as categorized in [87].

$\sigma(\psi(s,r,o))$ for $\langle s,r,o\rangle$, the loss is defined as:

$$\mathcal{L}(G) = \sum_{(s,r)} \sum_{o} y_o^{s,r} \log(\sigma(\psi(s,r,o))) \\ + (1 - y_o^{s,r}) \log(1 - \sigma(\psi(s,r,o))). \tag{3.1}$$

Gradient descent is used to learn the embeddings $\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{o}$, and the parameters of $\mathbf{f}$, if any.

3.3 Adversarial Modifications on Link Prediction (CRIAGE)

For adversarial modifications on KGs, we first define the space of possible modifications. For a target triple $\langle s,r,o\rangle$, we can remove (or inject) an attack triple in the form of $\langle s',r',o'\rangle$, where any of s', r', and o' are same as in the target, or are all different. We focus only on $\langle s',r',o\rangle$ triples as possible changes (we consider other modifications in Appendix A).

3.3.1 Removing a fact (CRIAGE-Remove)

For explaining a target prediction, we are interested in identifying the observed fact that has the most influence (according to the model) on the prediction. We define *influence* of an observed fact on the prediction as the change in the prediction score if the observed fact was not present when the embeddings were learned. There were previous works that define the concept of influence in the same way for several different tasks [45, 44]. Formally, for the target triple $\langle s,r,o\rangle$ and observed graph G, we want to identify a neighboring triple $\langle s',r',o\rangle \in G$ such that the score $\psi(s,r,o)$ when trained on G and the score $\overline{\psi}(s,r,o)$ when trained on $G - \{\langle s',r',o\rangle\}$ are maximally different, i.e.

$$\underset{(s',r') \in \mathrm{Nei}(o)}{\operatorname{argmax}} \; \Delta_{(s',r')}(s,r,o) \tag{3.2}$$

where $\Delta_{(s',r')}(s,r,o) = \psi(s,r,o) - \overline{\psi}(s,r,o)$, and $\text{Nei}(o) = \{(s',r')|\langle s',r',o\rangle \in G\}$. We restrict the search area to the neighboring links because they are likely to have the highest influence on the target triple and provide us with a reasonable search space; we leave larger neighborhoods of the observed graph to future work.

3.3.2 Adding a new fact (CRIAGE-Add)

We are also interested in investigating the robustness of models, i.e., how sensitive the predictions are to small additions to the knowledge graph. Specifically, for a target prediction $\langle s,r,o\rangle$, we are interested in identifying a single fake fact $\langle s',r',o\rangle$ that, when added to the knowledge graph G, changes the prediction score $\psi(s,r,o)$ the most. Using $\overline{\psi}(s,r,o)$ as the score after training on $G \cup \{\langle s',r',o\rangle\}$, we define the adversary as:

$$\underset{(s',r')}{\operatorname{argmax}} \Delta_{(s',r')}(s,r,o) \tag{3.3}$$

where $\Delta_{(s',r')}(s,r,o) = \psi(s,r,o) - \overline{\psi}(s,r,o)$. The search here is over any possible $s' \in \xi$, which is often in the millions for most real-world KGs, and $r' \in \mathcal{R}$. We also identify adversaries that *increase* the prediction score for specific false triple, i.e., for a target fake fact $\langle s,r,o\rangle$, the adversary is $\operatorname{argmax}_{(s',r')} -\Delta_{(s',r')}(s,r,o)$, where $\Delta_{(s',r')}(s,r,o)$ is defined as before.

3.3.3 Challenges

There are a number of crucial challenges when conducting a such an adversarial attack on KGs. First, evaluating the effect of changing the KG on the score of the target fact ($\overline{\psi}(s,r,o)$) is expensive since we need to update the embeddings by retraining the model on the new graph; a very time-consuming process that is at least linear in the size of G. Second, since there are many candidate facts that can be added to the knowledge graph, identifying the most promising adversary through search-based methods is very expensive. Specifically, the search size for unobserved facts is $|\xi| \times |\mathcal{R}|$, which, for example, in YAGO3-10 KG, can be as many as $4.5M$ possible facts for a single target prediction.

3.4 Efficiently Identifying the Modification

In this section, we propose algorithms to address mentioned challenges by (1) approximating the effect of changing the graph on a target prediction, and (2) using continuous optimization for the discrete search over potential modifications.

3.4.1 First-order Approximation of the Change

We first study the addition of a fact to the graph and then extend it to cover removal as well. To capture the effect of an adversarial modification on the score of a target triple, we need to study the effect of the change on the vector representations of the target triple. We use $\boldsymbol{s}$, $\boldsymbol{r}$, and $\boldsymbol{o}$ to denote the embeddings of s, r, o at the solution of $\operatorname{argmin} \mathcal{L}(G)$, and when considering the adversarial triple $\langle s', r', o \rangle$, we use $\overline{\mathbf{e}_s}$, $\overline{\mathbf{e}_r}$, and $\overline{\mathbf{e}_o}$ for the new embeddings of s, r, o, respectively. Thus $\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r}, \overline{\mathbf{e}_o}$ is a solution to $\operatorname{argmin} \mathcal{L}(G \cup \{\langle s', r', o \rangle\})$, which can also be written as $\operatorname{argmin} \mathcal{L}(G) + \mathcal{L}(\langle s', r', o \rangle)$. Similarly, $\mathbf{f}(\boldsymbol{s}, \boldsymbol{r})$ changes to $\mathbf{f}(\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r})$ after retraining.

Since we only consider adversaries in the form of $\langle s', r', o \rangle$, we only consider the effect of the attack on $\boldsymbol{o}$ and neglect its effect on $\boldsymbol{s}$ and $\boldsymbol{r}$. This assumption is reasonable since the adversary is connected with o and directly affects its embedding when added, but it will only have a secondary, negligible effect on $\boldsymbol{s}$ and $\boldsymbol{r}$, in comparison to its effect on $\boldsymbol{o}$. Further, calculating the effect of the attack on $\boldsymbol{s}$ and $\boldsymbol{r}$ requires a third order derivative of the loss, which is not practical ($O(n^3)$ in the number of parameters). In other words, we assume that

$\overline{\mathbf{e}_s} \simeq \boldsymbol{s}$ and $\overline{\mathbf{e}_r} \simeq \boldsymbol{r}$. As a result, to calculate the effect of the attack, $\overline{\psi}(s,r,o) - \psi(s,r,o)$, we need to compute $\overline{\mathbf{e}_o} - \boldsymbol{o}$, followed by:

$$\overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \boldsymbol{o}) \tag{3.4}$$

where $\mathbf{z}_{s,r} = \mathbf{f}(\boldsymbol{s}, \boldsymbol{r})$. We now derive an efficient computation for $\overline{\mathbf{e}_o} - \boldsymbol{o}$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$ over $\boldsymbol{o}$ is:

$$\nabla_{e_o}\mathcal{L}(\overline{G}) = \nabla_{e_o}\mathcal{L}(G) - (1-\varphi)\mathbf{z}_{s',r'} \tag{3.5}$$

where $\mathbf{z}_{s',r'} = \mathbf{f}(\boldsymbol{s}', \boldsymbol{r}')$, and $\varphi = \sigma(\psi(s', r', o))$. At convergence, after retraining, we expect $\nabla_{e_o}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_o}\mathcal{L}(\overline{G})$ to get:

$$\begin{aligned} 0 \simeq & -(1-\varphi)\mathbf{z}^{\intercal}_{s',r'} + \\ & (H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})(\overline{\mathbf{e}_o} - \boldsymbol{o}) \end{aligned} \tag{3.6}$$

where H_o is the $d \times d$ Hessian matrix for o, i.e., second order derivative of the loss w.r.t. $\boldsymbol{o}$, computed sparsely. Solving for $\overline{\mathbf{e}_o} - \boldsymbol{o}$ gives us, $\overline{\mathbf{e}_o} - \boldsymbol{o} =$:

$$(1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\intercal}_{s',r'}.$$

In practice, H_o is positive definite, making $H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\begin{aligned} \overline{\psi}(s,r,o) - \psi(s,r,o) &= \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \boldsymbol{o}) \\ &= \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\intercal}_{s',r'}). \end{aligned} \tag{3.7}$$

Calculating this expression is efficient since H_o is a $d \times d$ matrix (d is the embedding dimension), and $\mathbf{z}_{s,r}, \mathbf{z}_{s',r'} \in \mathbb{R}^d$. Similarly, we estimate the score change of $\langle s, r, o\rangle$ after *removing* $\langle s', r', o\rangle$

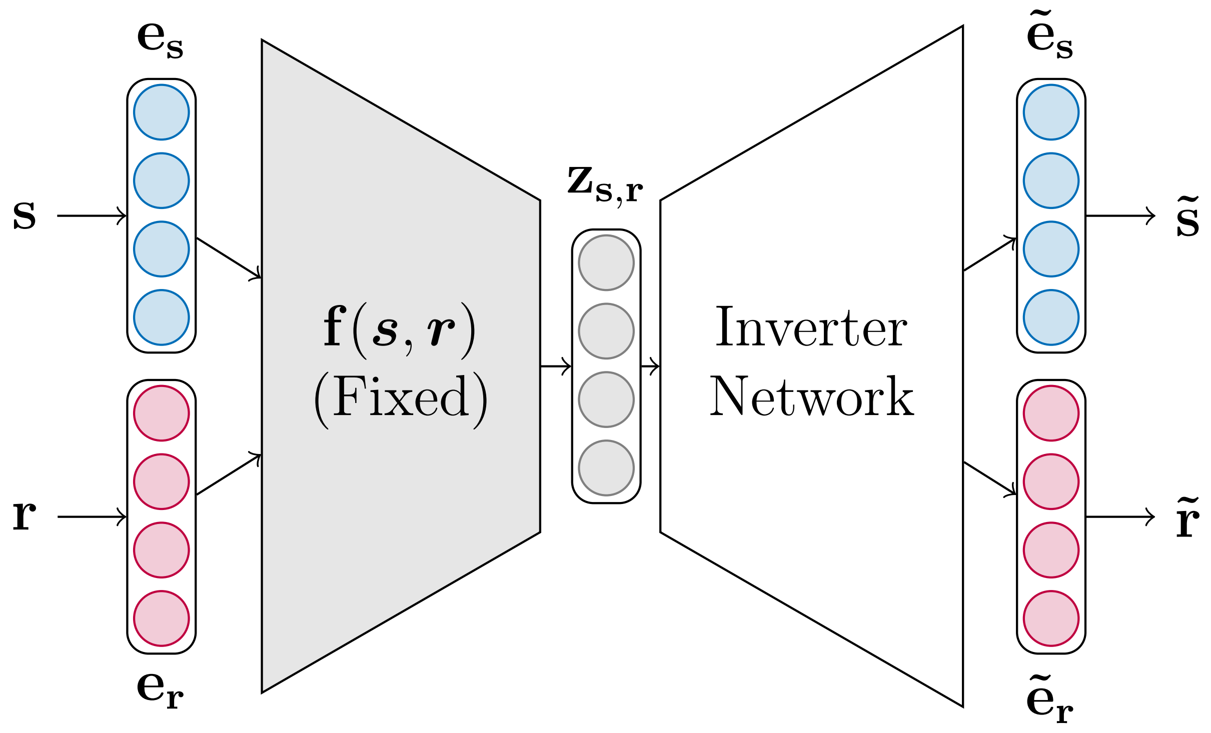


Figure 3.2: **Inverter Network** The architecture of our inverter function that translate $\mathbf{z}_{s,r}$ to its respective $(\tilde{s}, \tilde{r})$. The encoder component is fixed to be the encoder network of DistMult and ConvE, respectively.

as:

$$-\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s',r'}^\intercal \mathbf{z}_{s',r'})^{-1}\mathbf{z}_{s',r'}^\intercal). \tag{3.8}$$

3.4.2 Continuous Optimization for Search

Using the approximations provided in the previous section, Eq. equation 3.7 and equation 3.8, we can use brute force enumeration to find the adversary $\langle s', r', o \rangle$. This approach is feasible when removing an observed triple since the search space of such modifications is usually small; it is the number of observed facts that share the object with the target. On the other hand, finding the most influential unobserved fact to add requires a search over a much larger space of all possible unobserved facts (that share the object). Instead, we identify the most influential unobserved fact $\langle s', r', o\rangle$ by using a gradient-based algorithm on vector $\mathbf{z}_{s',r'}$ in the embedding space (reminder, $\mathbf{z}_{s',r'} = \mathbf{f}(\boldsymbol{s}', \boldsymbol{r}')$), solving the following continuous optimization problem in $\mathbb{R}^d$:

$$\underset{\mathbf{z}_{s',r'}}{\operatorname{argmax}}\, \Delta_{(s',r')}(s, r, o). \tag{3.9}$$

After identifying the optimal $\mathbf{z}_{s',r'}$, we still need to generate the pair (s', r'). We design a network, shown in Figure 3.2, that maps the vector $\mathbf{z}_{s',r'}$ to the entity-relation space, i.e., translating it into (s', r'). In particular, we train an auto-encoder where the encoder is fixed to receive the s and r as one-hot inputs, and calculates $\mathbf{z}_{s,r}$ in the same way as the DistMult and ConvE encoders, respectively (using trained embeddings). The decoder is trained to take $\mathbf{z}_{s,r}$ as input and produce s and r, essentially inverting $\mathbf{f}$ and the embedding layers. As our decoder, for Distmult, we pass $\mathbf{z}_{s,r}$ through a linear layer and then use two other linear layers for the subject and the relation separately, providing one-hot vectors as $\tilde{s}$ and $\tilde{r}$. For ConvE, we pass $\mathbf{z}_{s,r}$ through a deconvolutional layer and then use the same architecture as the DistMult decoder. Although we could use maximum inner-product search [88] for DistMult instead of our defined inverter function, we were looking for a general algorithm that would work across multiple models.

We evaluate the performance of our inverter networks (one for each model/dataset) on correctly recovering the pairs of subject and relation from the test set of our benchmarks, given the $\mathbf{z}_{s,r}$. The accuracy of recovered pairs (and of each argument) is given in Table 3.1. As shown, our networks achieve a very high accuracy, demonstrating their ability to invert vectors $\mathbf{z}_{s,r}$ to $\{s, r\}$ pairs.

	WordNet		**YAGO**	
	DistMult	ConvE	DistMult	ConvE
Recover s	93.4	96.1	97.2	98.1
Recover r	91.3	95.3	99.0	99.6
Recover $\{s, r\}$	89.5	94.2	96.4	98.0

Table 3.1: **Inverter Functions Accuracy**, we calculate the accuracy of our inverter networks in correctly recovering the pairs of subject and relation from the test set of our benchmarks.

	# **Rels**	#**Entities**	# **Train**	#**Test**
Nations	56	14	1592	200
Kinship	26	104	4,006	155
WN18	18	40,943	141,442	5000
YAGO3-10	37	123,170	1,079,040	5000

Table 3.2: **Data Statistics** of the benchmarks.

3.5 Experiment Setup

Datasets To evaluate our method, we conduct several experiments on four widely used KGs. To validate the accuracy of the approximations, we use smaller-sized Kinship and Nations KGs for which we can make comparisons against more expensive but less approximate approaches. For the remaining experiments, we use YAGO3-10 and WN18 KGs, which are closer to real-world KGs in their size and characteristics. Table 3.2 provides the statistics of the datasets.

Models We implement all methods using the same loss and optimization for training, i.e., AdaGrad and the binary cross-entropy loss. We use validation data to tune the hyperparameters and use a grid search to find the best hyperparameters, such as the regularization parameter, and the learning rate of the gradient-based method. To capture the effect of our method on the link prediction task, we study the change in commonly-used metrics for evaluation in this task: mean reciprocal rank (MRR) and Hits@K. Further, we use the same hyperparameters as in [25] for training link prediction models in the knowledge graphs.

Influence Function We also compare our method with the influence function approach (IF) [44]. As described in Chapter 1, the influence function approximates the effect of upweighting a training sample on the loss for a specific test point. We use IF to approximate the change in the loss after removing a triple as:

$$\begin{aligned}&\mathcal{I}_{\text{up,loss}}(\langle s', r', o\rangle, \langle s, r, o\rangle) = \\ &-\nabla_{\theta}\mathcal{L}(\langle s, r, o\rangle, \hat{\theta})^{\intercal} H_{\hat{\theta}}^{-1} \nabla_{\theta}\mathcal{L}(\langle s', r', o\rangle, \hat{\theta})\end{aligned} \tag{3.10}$$

where $\langle s', r', o\rangle$ and $\langle s, r, o\rangle$ are training and test samples respectively, $\hat{\theta}$ represents the optimum parameters and $\mathcal{L}(\langle s, r, o\rangle, \hat{\theta})$ represents the loss function for the test sample $\langle s, r, o\rangle$. Influence function does not scale well, so we only compare our method with IF on the smaller size KGs.

3.6 Experiments

We evaluate CRIAGE by (3.6.1) comparing CRIAGE estimate with the actual effect of the attacks, (3.6.2) studying the effect of adversarial attacks on evaluation metrics, (3.6.3) exploring its application to the interpretability of KG representations, and (3.6.4) detecting incorrect triples.

3.6.1 Influence Function vs CRIAGE

To evaluate the quality of our approximations and compare them with influence function (IF), we conduct leave-one-out experiments. In this setup, we take all the neighbors of a random target triple as candidate modifications, remove them one at a time, retrain the model each time, and compute the *exact* change in the score of the target triple. We can

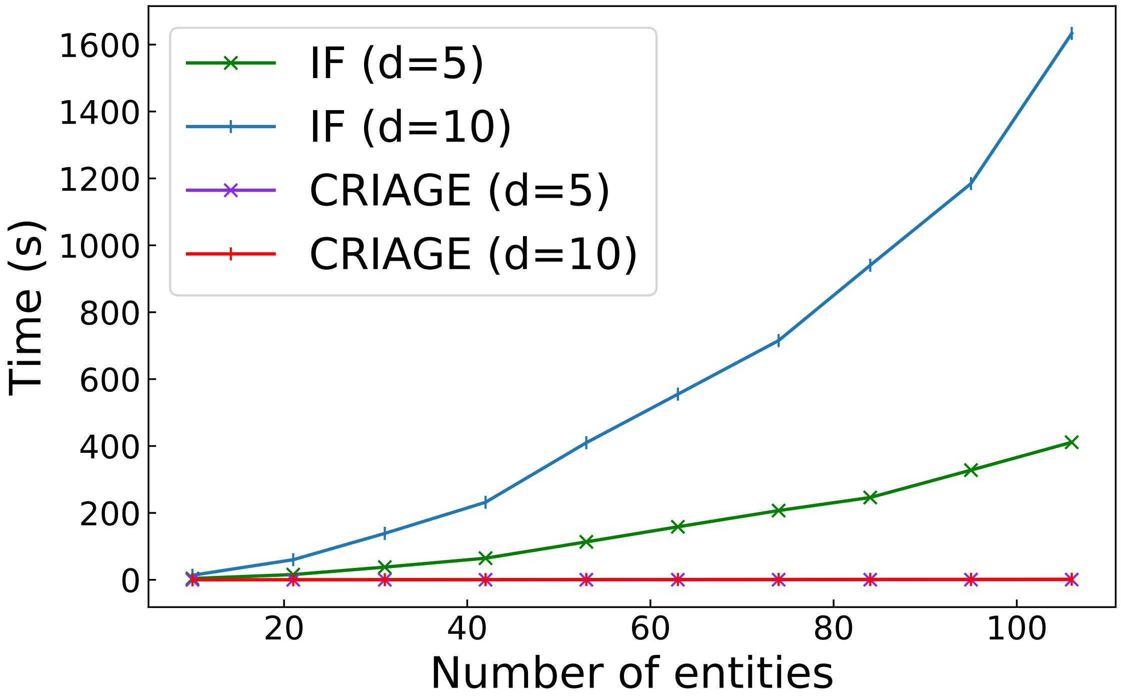


Figure 3.3: **Influence function vs CRIAGE**. We plot the average time (over 10 facts) of influence function (IF) and CRIAGE to identify an adversary as the number of entities in the Kinship KG is varied (by randomly sampling subgraphs of the KG). Even with small graphs and dimensionality, IF quickly becomes impractical.

use the magnitude of this change in score to rank the candidate triples, and compare this *exact* ranking with ranking as predicted by: CRIAGE-Remove, influence function with and without Hessian matrix, and the original model score (with the intuition that facts that the model is most confident of will have the largest impact when removed). Similarly, we evaluate CRIAGE-Add by considering 200 random triples that share the object entity with the target sample as candidates, and rank them as above.

The average results of Spearman's ρ and Kendall's τ rank correlation coefficients over 10 random target samples is provided in Table 3.3. CRIAGE performs comparably to the influence function, confirming that our approximation is accurate. Influence function is slightly more accurate because it uses the complete Hessian matrix over all the parameters, while we only approximate the change by calculating the Hessian over $\boldsymbol{o}$. The effect of this

Methods	**Nations**				**Kinship**			
	Adding		**Removing**		**Adding**		**Removing**	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Ranking Based on Score	0.03	0.02	-0.01	-0.01	-0.09	-0.06	0.01	0.01
Influence Function without Hessian	0.15	0.12	0.12	0.1	0.77	0.71	0.77	0.71
CRIAGE (Brute Force)	0.95	0.84	0.94	0.85	0.99	0.97	0.99	0.95
Influence Function	0.99	0.95	0.99	0.96	0.99	0.98	0.99	0.98

Table 3.3: **Ranking Modifications by Impact on Target**. We compare the *true* ranking of candidate triples with a number of approximations using ranking correlation coefficients. We compare our method with influence function (IF) with and without Hessian, and ranking the candidates based on their score, on two KGs ($d = 10$, averaged over 10 random targets). For the sake of brevity, we represent Spearman's ρ and Kendall's τ rank correlation coefficients simply as ρ and τ.

difference on scalability is dramatic, limiting IF to very small graphs and small embedding dimensionality ($d \leq 10$) before we run out of memory. In Figure 3.3, we show the time to compute a single adversary by IF compared to CRIAGE, as we steadily grow the number of entities (randomly chosen subgraphs), averaged over 10 random triples. As it shows, CRIAGE is mostly unaffected by the number of entities while IF increases quadratically. Considering that real-world KGs have tens of thousands of times more entities, making IF infeasible for them.

3.6.2 Robustness of Link Prediction Models

Now we evaluate the effectiveness of CRIAGE to successfully *attack* link prediction task by adding false facts. The goal here is to identify the attacks for triples in the test data, and measuring their effect on MRR and Hits@ metrics (ranking evaluations) after conducting the attack and retraining the model.

Since this is the first work on adversarial attacks for link prediction, we introduce several baselines to compare against our method. For finding the adversarial fact to add for the target triple $\langle s, r, o\rangle$, we consider two baselines: (1) choosing a random fake fact $\langle s', r', o\rangle$

	Models	**YAGO3-10**				**WN18**			
		All-Test		**Uncertain-Test**		**All-Test**		**Uncertain-Test**	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
DistMult	DistMult	0.458	37 (0)	1.0	100 (0)	0.938	93.1 (0)	1.0	100 (0)
	+ Adding Random Attack	0.442	34.9 (-2.1)	0.91	87.6 (-12.4)	0.926	91.1 (-2)	0.929	90.4 (-9.6)
	+ Adding Opposite Attack	0.427	33.2 (-3.8)	0.884	84.1 (-15.9)	0.906	87.3 (-5.8)	0.921	91 (-9)
	+ CRIAGE-Add	0.379	29.1 (-7.9)	0.71	58 (-42)	0.89	86.4 (-6.7)	0.844	81.2 (-18.8)
	+ CRIAGE-FT	0.387	27.7 (-9.3)	0.673	50.5 (-49.5)	0.86	79.2 (-13.9)	0.83	74.5 (-25.5)
	+ CRIAGE-Best	0.372	26.9 (-10.1)	0.658	49.3 (-50.7)	0.838	77.9 (-15.2)	0.814	72.7 (-27.3)
ConvE	ConvE	0.497	41.2 (0)	1.0	100 (0)	0.94	93.3 (0)	1.0	100 (0)
	+ Adding Random Attack	0.474	38.4 (-2.8)	0.889	83 (-17)	0.921	90.1 (-3.2)	0.923	89.7 (-10.3)
	+ Adding Opposite Attack	0.469	38 (-3.2)	0.874	81.9 (-18.1)	0.915	88.9 (-4.4)	0.908	88.1 (-11.9)
	+ CRIAGE-Add	0.454	36.9 (-4.3)	0.738	61.5 (-38.5)	0.897	87.8 (-5.5)	0.895	87.6 (-12.4)
	+ CRIAGE-FT	0.441	33.2 (-8)	0.703	57.4 (-42.6)	0.865	80 (-13.3)	0.874	79.5 (-20.5)
	+ CRIAGE-Best	0.423	31.9 (-9.3)	0.677	54.8 (-45.2)	0.849	79.1 (-14.2)	0.858	78.4 (-21.6)

Table 3.4: **Robustness of Representation Models**, the effect of adversarial attack on link prediction task. We consider two scenarios for the target triples, (1) choosing the whole test dataset as the targets (All-Test) and (2) choosing a subset of test data that models are uncertain about them (Uncertain-Test).

(**Random Attack**); (2) finding (s', r') by first calculating $\mathbf{f}(\boldsymbol{s}, \boldsymbol{r})$ and then feeding $-\mathbf{f}(\boldsymbol{s}, \boldsymbol{r})$ to the decoder of the inverter function (**Opposite Attack**). In addition to CRIAGE-Add, we introduce two other alternatives of our method: (1) **CRIAGE-FT**, which uses CRIAGE to *increase* the score of fake fact over a test triple, i.e., we find the fake fact the model ranks second after the test triple, and identify the adversary for them, and (2) **CRIAGE-Best** that selects between CRIAGE-Add and CRIAGE-FT attacks based on which has a higher estimated change in score.

All-Test The result of the attack on all test facts as targets is provided in the Table 3.4. CRIAGE-Add outperforms the baselines, demonstrating its ability to effectively attack the KG representations. It seems DistMult is more robust against random attacks, while ConvE is more robust against designed attacks. CRIAGE-FT is more effective than CRIAGE-Add since changing the score of a fake fact is easier than of actual facts; there is no existing evidence to support fake facts. We also see that YAGO3-10 models are more robust than those for WN18. Looking at sample attacks (provided in Appendix A), CRIAGE mostly tries to change the *type* of the target object by associating it with a subject and a relation for a

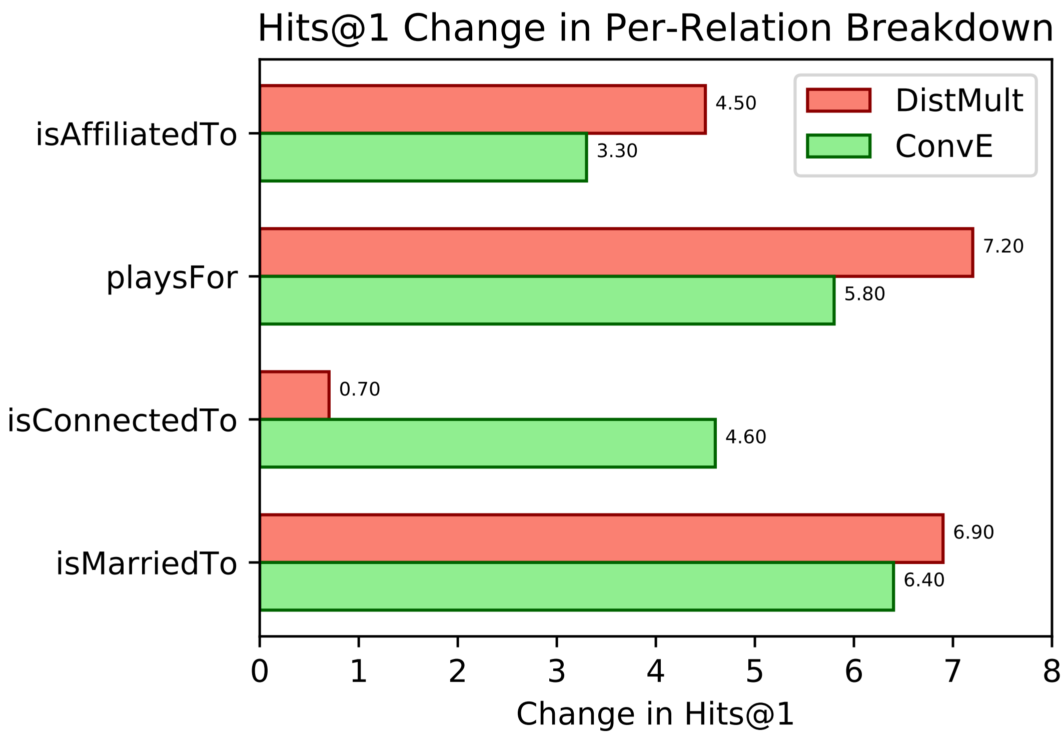


Figure 3.4: Per-Relation Breakdown showing the effect of CRIAGE-Add on different relations in YAGO3-10.

different entity type.

Uncertain-Test To better understand the effect of attacks, we consider a subset of test triples that (1) the model predicts correctly, (2) difference between their scores and the negative sample with the highest score is minimum. This "Uncertain-Test" subset contains 100 triples from each of the original test sets, and we provide results of attacks on this data in Table 3.4. The attacks are much more effective in this scenario, causing a considerable drop in the metrics. Further, in addition to CRIAGE significantly outperforming other baselines, they indicate that ConvE's confidence is much more robust.

Relation Breakdown We perform additional analysis on the YAGO3-10 dataset to gain a deeper understanding of the performance of our model. As shown in Figure 3.4, both of the DistMult and ConvE provide a more robust representation for `isAffiliatedTo`

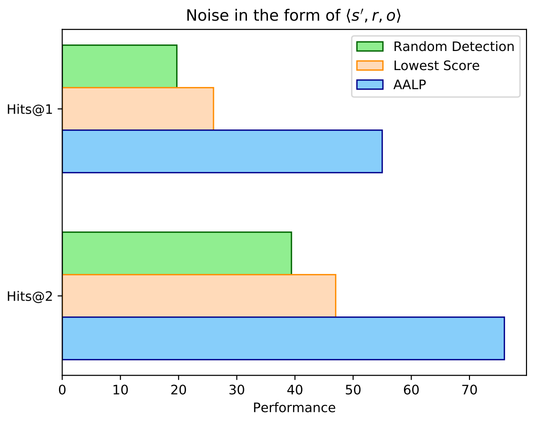
Rule Body, $R_1(a,c) \wedge R_2(c,b) \Rightarrow$	**Target,** $R(a,b)$
Common to both	
isConnectedTo$(a,c)\wedge$ isConnectedTo(c,b)	isConnectedTo
isLocatedIn$(a,c)\wedge$ isLocatedIn(c,b)	isLocatedIn
isAffiliatedTo$(a,c)\wedge$ isLocatedIn(c,b)	wasBornIn
isMarriedTo$(a,c)\wedge$ hasChild(c,b)	hasChild
only in DistMult	
playsFor$(a,c)\wedge$ isLocatedIn(c,b)	wasBornIn
dealsWith$(a,c)\wedge$ participatedIn(c,b)	participatedIn
isAffiliatedTo$(a,c)\wedge$ isLocatedIn(c,b)	diedIn
isLocatedIn$(a,c)\wedge$ hasCapital(c,b)	isLocatedIn
only in ConvE	
influences$(a,c)\wedge$ influences(c,b)	influences
isLocatedIn$(a,c)\wedge$ hasNeighbor(c,b)	isLocatedIn
hasCapital$(a,c)\wedge$ isLocatedIn(c,b)	exports
hasAdvisor$(a,c)\wedge$ graduatedFrom(c,b)	graduatedFrom
Extractions from DistMult [107]	
isLocatedIn(a,c) $\wedge$ isLocatedIn(c,b)	isLocatedIn
isAffiliatedTo(a,c) $\wedge$ isLocatedIn(c,b)	wasBornIn
playsFor(a,c) $\wedge$ isLocatedIn(c,b)	wasBornIn
isAffiliatedTo(a,c) $\wedge$ isLocatedIn(c,b)	diedIn

Table 3.5: **Extracted Rules** for identifying the most influential link. We extract the patterns that appear more than 90% times in the neighborhood of the target triple. The output of CRIAGE-Remove is presented in red.

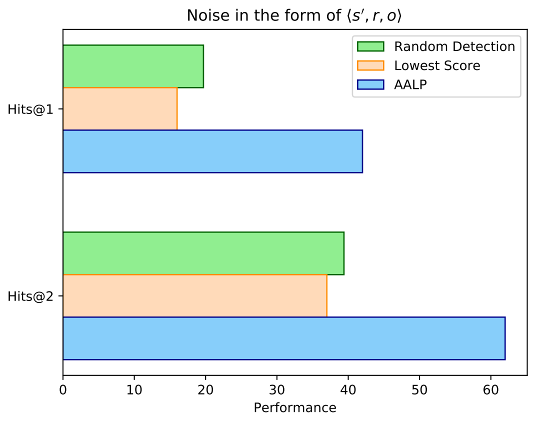
and `isConnectedTo` relations, demonstrating the confidence of models in identifying them. Moreover, the CRIAGE affects DistMult more in `playsFor` and `isMarriedTo` relations while affecting ConvE more in `isConnectedTo` relations.

3.6.3 Interpretability of Models

To be able to understand and interpret why a link is predicted using the opaque, dense embeddings, we need to find out which part of the graph was most influential on the prediction. To provide such explanations for each predictions, we identify the most influential fact using CRIAGE-Remove. Instead of focusing on individual predictions, we aggregate the



(a) Adding noise in the form of $\langle s', r, o \rangle$.

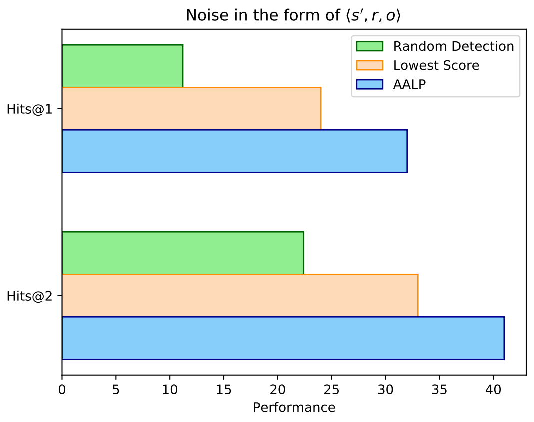


(b) Adding noise in the form of $\langle s', r', o \rangle$.

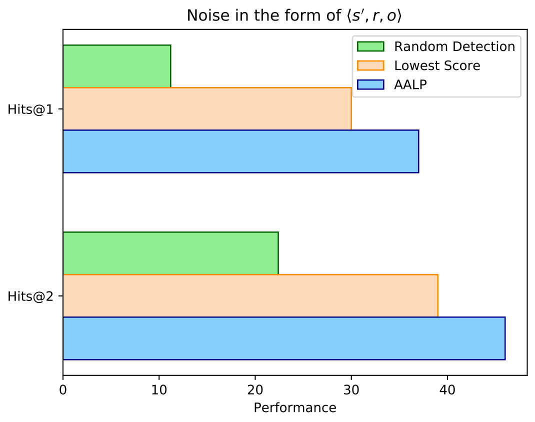
Figure 3.5: The accuracy of detecting errors in the neighborhood of 100 chosen samples. We choose the neighbor with the highest value according to Eq equation 3.11 as the incorrect fact. This experiment assumes we know each target fact has exactly one error.

explanations over the whole dataset for each relation using a simple rule extraction technique: we find simple patterns on subgraphs that surround the target triple and the removed fact from CRIAGE-Remove, and appear more than 90% of the time. We only focus on extracting length-2 horn rules, i.e., $R_1(a, c) \wedge R_2(c, b) \Rightarrow R(a, b)$, where $R(a, b)$ is the target and $R_2(c, b)$ is the removed fact.

Table 3.5 shows extracted YAGO3-10 rules that are common to both models, and ones that are not. The rules show several interesting inferences, such that `hasChild` is often inferred via married parents, and `isLocatedIn` via transitivity. There are several differences in how the models reason as well; DistMult often uses the `hasCapital` as an intermediate step for `isLocatedIn`, while ConvE *incorrectly* uses `isNeighbor`. We also compare against the rules extracted [107] for YAGO3-10 that utilizes the structure of DistMult: they require domain knowledge on types and cannot be applied to ConvE. Interestingly, the extracted rules contain all the rules provided by CRIAGE, demonstrating that CRIAGE can be used to accurately interpret models, including ones that are not interpretable, such as ConvE. These are preliminary steps toward the interpretability of link prediction models, and we leave more analysis of interpretability to future work.



(a) Adding noise in the form of $\langle s', r, o\rangle$.



(b) Adding noise in the form of $\langle s', r', o\rangle$.

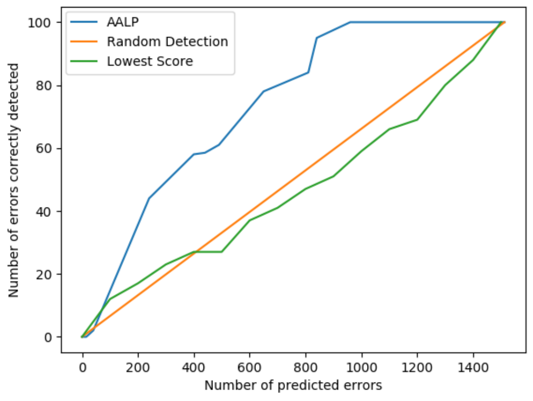
Figure 3.6: The accuracy of detecting errors in the neighborhood of 100 chosen samples. We choose the neighbor with the highest value according to Eq equation 3.11 as the incorrect fact. This experiment assumes we know each target fact has exactly one error.

3.6.4 Finding Errors in Knowledge Graphs

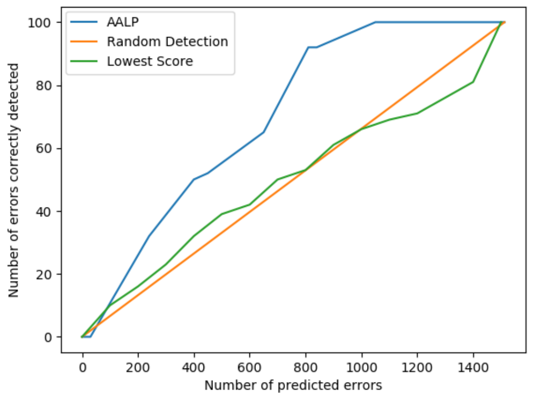
Here, we demonstrate another potential use of adversarial modifications: finding erroneous triples in the knowledge graph. Intuitively, if there is an error in the graph, the triple is likely to be inconsistent with its neighborhood, and thus the model should put the least trust in this triple. In other words, the error triple should have the least influence on the model's prediction of the training data. Formally, to find the incorrect triple $\langle s', r', o\rangle$ in the neighborhood of the train triple $\langle s, r, o\rangle$, we need to find the triple $\langle s', r', o\rangle$ that results in the *least* change $\Delta_{(s',r')}(s, r, o)$ when removed from the graph, i.e.:

$$\operatorname*{argmin}_{(s',r')} \Delta_{(s',r')}(s, r, o) \tag{3.11}$$

To evaluate this application, we inject random triples into the graph, and measure the ability of CRIAGE to detect the errors using our optimization. We consider two types of incorrect triples: (1) incorrect triples in the form of $\langle s', r, o\rangle$ where s' is chosen randomly from all of the entities, and (2) incorrect triples in the form of $\langle s', r', o\rangle$ where s' and r' are chosen



(a) Adding noise in the form of $\langle s', r, o \rangle$.



(b) Adding noise in the form of $\langle s', r', o \rangle$.

Figure 3.7: We plot the number of predicted errors vs the number of correctly detected errors by assigning a global threshold on the approximation of the change of targets' score.

randomly. We choose 100 random triples from the observed graph, and for each of them, add an incorrect triple (in each of the two scenarios) to its neighborhood. Then, after retraining DistMult on this noisy training data, we identify error triples through a search over the neighbors of the 100 facts. The result of choosing the neighbor with the least influence on the target is provided in Figures 3.5 and 3.6. When compared with baselines that randomly choose one of the neighbors, or assume that the fact with the lowest score is incorrect, we see that CRIAGE-based approximation outperforms both of these with a considerable gap, and obtains an accuracy of 42% and 55% in detecting errors.

We also study the scenario where we do not assume we know that each target fact has a single error in its neighbor; instead only know about the total number of errors. We plot the number of predicted errors vs. the number of correctly detected errors by assigning a global threshold on the approximation of the change of scores for 1512 neighbors of our target samples in the Figure 3.7. Similar to accuracy, our method provides a more accurate detection of errors compared to the baselines.

3.7 Related Work

Learning relational knowledge representations has been a focus of active research in the past few years, but to the best of our knowledge, this is the first work on conducting an adversarial modifications on the link prediction task.

Knowledge graph embedding There is a rich literature on representing knowledge graphs in vector spaces that differ in their scoring functions. Although CRIAGE is primarily applicable to multiplicative scoring functions [61, 89, 108, 101], these ideas are expandable to additive scoring functions [12, 105, 47, 59] as well.

Furthermore, there is a growing body of literature that incorporates an extra type of evidence to provide a more informative embeddings by combining the entity and its feature representations. We can further utilize the CRIAGE on those that build their embeddings on top of a multiplicative scoring function. As a result, using CRIAGE, we can gain a deeper understanding of these methods, which use an extra type of evidence such as numerical values [29], images [62], text [99, 100, 102], and a combination of them [65].

Interpretability and Adversarial Modification There has been a significant recent interest in conducting adversarial attacks on different machine learning models [10, 63, 27, 112, 113] to attain interpretability, and further evaluate the robustness of those models. [44] uses influence function to provide a novel approach to understand black-box models by studying the changes in the loss occurring as a result of changes in the training data. In addition to incorporating their established method on KGs, we derive a novel approach that differs from their procedure in two ways: (1) instead of changes in the loss, we consider the changes in the scoring function, which is more appropriate for KG representations, and (2) in addition to searching for the attack, we introduce a gradient-based method that is much faster, especially for "adding an attack triple" (the size of search space makes the influence function method infeasible). Previous work has also considered adversaries for KGs, but as part of training to improve their representation of the graph [55, 16].

Adversarial Attack on KG Although this is the first work on adversarial attacks for link prediction, there are two approaches [22, 115] that consider the task of adversarial attack on graphs. There are a few fundamental differences from our work: (1) they build their method on top of a path-based representation while we focus on embeddings, (2) they consider node classification as the target of their attacks while we attack link prediction, and (3) they conduct the attack on small graphs due to restricted scalability, while the complexity of our method does not depend on the size of the graph, but only the neighborhood, allowing us to attack real-world graphs.

3.8 Conclusions

Motivated by the need to analyze the robustness and interpretability of link prediction models, in this chapter, I present a novel approach for conducting adversarial modifications to knowledge graphs. I introduce CRIAGE, an adversarial modification for link prediction models: identifying the fact to add into or remove from the KG that changes the prediction for a target fact. CRIAGE uses (1) an estimate of the score change for any target triple after adding or removing another fact, and (2) a gradient-based algorithm for identifying the most influential modification. We show that CRIAGE can effectively reduce ranking metrics on link prediction models upon applying the attack triples. Further, we incorporate the CRIAGE to study the interpretability of KG representations by summarizing the most influential facts for each relation. Finally, using CRIAGE, we introduce a novel automated error detection method for knowledge graphs. Code to reproduce the results is available here: `https://pouyapez.github.io/criage`.

Chapter 4

Problematic Patterns in Knowledge Bases and Shortcomings of Evaluation of Completion Models

4.1 Introduction

As mentioned in previous chapters, most knowledge graphs, in practice, are often substantially incomplete and contain noise. Unfortunately, the lack of a complete and accurate KG is a problem for the evaluation of link prediction models as well. Since it is not possible to list *all possible* true and false facts for a KG of interest, existing evaluation of KGC consists of gathering known *true* facts, and using: (1) *ranking metrics*, such as Hits@N and Mean Reciprocal Rank (MRR), to calculate the relative rank of these known true facts against all unknown facts (thus implicitly treated as negative), and (2) *classification accuracy* of individual facts, by treating random corruptions of a known true fact as negative/false facts. In spite of steady and significant progress on these models, it is not clear whether these metrics correspond to the true performance on link prediction, making it difficult to decide whether they are ready for real-world deployment. Further, due to the strong assumptions made by these evaluation metrics, the strengths, shortcomings, and reasoning capabilities underlying these link prediction methods is difficult to determine, hindering further progress of the field.

In this chapter, we study significant issues with the current evaluation metrics for knowledge graph completion models, in particular, highlighting the impact of the assumptions made by these metrics on model performance. We show that the ranking metrics often do not correlate well with the actual performance of the model, making it incredibly challenging to determine whether these models are well-calibrated or not (an essential property for real-world deployment), and do not correlate well with the reasoning power of the models. For triple classification, upon a detailed examination of several commonly used benchmarks, we show that the metric is heavily sensitive to the choice of negative sampling, and that there is a significant mismatch between accuracy and the ranking metrics.

To address these shortcomings in existing benchmarks, we introduce **YAGO3-TC**, a high-quality, manually-annotated dense sub-graph of the YAGO3-10 KG. Along with the true facts that are already present in test and validation splits of the existing benchmark, YAGO3-TC also includes related facts involving the same entities that are annotated to be true or false via crowdsourcing. These related facts are designed to be somewhat challenging to discriminate since they are high-scoring by recent accurate models, resulting in 28,364 labeled facts out of which 2,976 are positive. Since we ensure the quality of the annotations, classification metrics such as accuracy, precision/recall, etc., can be used to appropriately evaluate models of knowledge graph completion.

We also provide a comprehensive analysis of recent KG completion models, given the high-quality annotations in YAGO3-TC, using triple classification metrics. We are able to provide accurate calibration results for completion models, showing that they are significantly overconfident (consistent with existing results for neural networks, but different from other observations for KGC). Further, we observe that there is a significant mismatch between ranking metrics and performance on the completion task (e.g., there is more than 20% gap between Hits@1 and Precision). Most importantly, we show that the progress in performance indicated by ranking metrics does not align with the actual completion task; simple methods achieve similar performance to state-of-art models.

4.2 Background and Notation

In this section, we introduce benchmarks, evaluation procedures, and a brief overview of existing relational embedding approaches to knowledge graph completion. More details on existing benchmarks and implementation details are provided in Appendix B.

Embedding Based KGC: As described previously, To model a KG for link prediction, a scoring function $\psi : \xi \times \mathcal{R} \times \xi \rightarrow \mathbb{R}$ is learned to evaluate whether any given fact is true. In this chapter, as scoring functions, we will primarily study DistMult [108] due to its simplicity and popularity, and RotatE [94] and Tucker [7] because of their state-of-the-art performance. Also, we use the same setup as previous chapters for training, i.e., using the binary cross-entropy loss on the score of positive and negative triples.

Ranking Metrics: To evaluate the performance of the KG completion models, we rank test triples against all possible negative samples, generated by corrupting the subject or object of the target triple. Ranking metrics have been used since existing KGs are open-world, and the ground truth label for all negative and positive samples is not available. In the *filtered* setting, which we consider in this chapter, we only treat triples that do not appear in the training, validation, or test set, as possible negative samples. To quantify the ranking of target triples, we use standard evaluation metrics such as Mean Reciprocal Rank (MRR), which is the average inverse rank of all test triples, and Hits@N, which is the percentage of test triples whose rank is lower (better) than or equal to N.

Triple classification: Triple classification is the task of binary classification on the KGs triples. This task is important because, if appropriately set up, it directly evaluates the capability of KGC models in identifying missing links. Specifically, given a target triple $\langle s, r, o \rangle$, we want to identify if this is a positive/true fact or a negative one. For this task, previous approaches learn a specific threshold τ_r for each relation, over validation data. In order to create the negative samples in these approaches, for both validation and test data, they corrupt the subject or object of the target triple with a random entity from the KG. After learning thresholds τ_r, a triple is assumed to be positive if its score is higher than the threshold for the triple's relation.

4.3 Issues in Existing KG Completion Evaluation

In this section, we discuss some issues prevalent in current evaluation metrics, and provide empirical evidence for their shortcomings. First, we observe that the assumptions underlying ranking metrics are often incompatible with the goals of the completion itself. Then, we show that evaluating how well completion models are calibrated is challenging since the results are incredibly sensitive to the setup design choices. Finally, we show that the results of the ranking metrics are often inconsistent with the results of the triple classification evaluation.

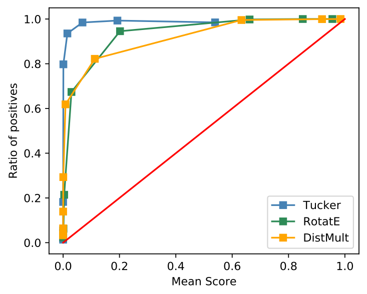
4.3.1 Assumptions in Ranking Metrics

In the past few years, we have observed tremendous progress in the performance of KG completion models, based on ranking metrics. As these models become increasingly accurate and potentially ready for real-world deployment, it is now useful to understand the extent to which these ranking metrics align with the actual goals of the *completion* task.

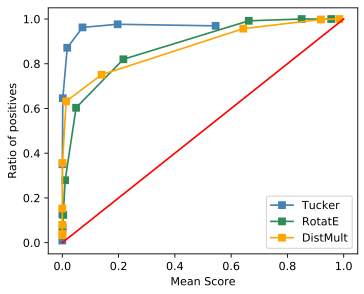
Let us consider a simple example. Assume we want to validate whether triple $\langle s, r, o \rangle$ is true or not. According to the current procedure, our only option is to rank the score of all possible objects (triples of the form $\langle s, r, o' \rangle$) and subjects (of the form $\langle s', r, o \rangle$) and compute the rank of our target triple. In this case, the ranking metrics such as Hits@N can only tell us whether this triple appears in the top N possible triples. If the relation of our target triple can accept only one true object, i.e., the relation is N-1, this ranking is meaningful; however, if multiple objects can be true for our target subject and relation, this ranking is incomplete since does not capture other triples that are ranked higher than the target fact are themselves true or not. A similar observation holds for relations for which multiple subjects can be true for the same object, i.e., a 1-N relation. Unfortunately, on studying two commonly used KGs WN18RR and YAGO3-10, we notice that this phenomenon happens in a huge portion of the data, as a result of the existence of semi-inverse relations.

Semi-Inverse Relations in WN18RR: On conducting a simple statistical analysis on WN18RR, we notice that this KG is not completely free of relations that are the inverse of each other (which makes the completion task trivial). We notice in more than 90% appearance of three relations *_derivationally_related_form*, *_verb_group*, and *_similar_to*, and also more than 60% appearance of *_also_see*, another triple with the same entities but in the opposite direction of the original relation appears in the training data. Together, these relations consist 37% of this KG. Moreover, around one-third of triples in the test data contain one of these relations; the KGC models achieve more than 0.95 MRR performance on this subset, significantly affecting the overall performance.

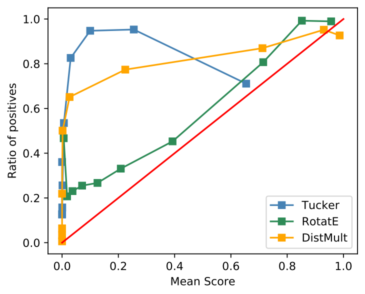
Semi-Inverse Relations in YAGO3-10: Along the similar lines, in YAGO3-10, for 75% of the triples with relation *isAffiliatedTo*, the triple with relation *playsFor* appears between



(a) Random-N on YAGO3-10



(b) Constraint-N on YAGO3-10



(c) Careful-N on YAGO3-10

Figure 4.1: Calibration study on different KGs based on three negative sampling procedures. We plot reliability diagrams of the fraction of positive triples to all the triples vs the link prediction models' score for a target triple. Being closer to the diagonal means the model is more calibrated.

the same object and subject (87% for reverse). These relations consist of 63% of test data, 81% of which have the other relation between the same subject and object in the training data. Note that embedding methods achieve around 0.95 of MRR performance on these triples.

The existence of these semi-inverse relations results in two important conclusions. First, it indicates that the performance based on ranking metrics on these KGs is not trustworthy since it is very easy to predict these triples. Second, since these relations can accept many objects (and subjects) for the same subject (and object), and embedding methods score all of those objects (and subject) *very highly* (since their semi-inverse version of them appear in the training data), low values of ranking metrics are clearly not an accurate assessment of completion on these triples. More specifically, for a target triple with one of the mentioned relations, as the number of *true* objects or subjects with semi-inverse relation increases, the chance of obtaining a worse ranking increases as well.

4.3.2 Evaluating Calibration of the Models

Calibration is a very important aspect of KG completion that has only recently received attention [97]. Treating the probability of the truth of a fact ($\sigma(\psi(s, r, o))$ for triple $\langle s, r, o\rangle$) as the confidence of the model for the triple, we consider our model to be calibrated if the confidence aligns with the rate of true facts. In other words, if confidence is equal to 0.5, we expect to have around 50% of triples with this confidence to be true. If this proportion is far from 50%, then the model is not calibrated (the model is under-confident if the proportion is higher, and over-confident if it is lower). Since the evaluation only consists of true facts, we need to obtain negative/false facts by sampling. We use three different negative sampling procedures: (1) randomly replacing the subject or object with an entity from all possible ones (*Random-N*), commonly used in KG completion literature, (2) randomly replacing the subject or object with an entity that has appeared at least once with the target relation in the training data (*Constraint-N*), which was used by [89] to generate more challenging negative samples, and (3) choose the highest scoring negative sample that has the object (or subject) with a different type than the target triple object (*Careful-N*). By choosing the object (or subject) that has a different type than the target triple entities, we enforce the chosen negative sample be a true negative. We define entity type as the set of entities that have appeared with similar relations (see Appendix B for a precise definition).

The result of the calibration study based on above negative samples is shown for YAGO3-10 in Figure 4.1 (calibration plot for WN18RR and FB15k-237 and histogram plot of score distributions is provided in Appendix B). Note that even though negative samples for these three negative sampling methods are different, we generate each plot on the same set of negative samples for the three models. Although these results show that RotatE provides more calibrated models compared to Tucker in all the negative sampling procedures, the advantage of RotatE over DistMult changes with different negative samples (in *Random-N*, DistMult appears better than RotatE). Further, we suspect the reason behind the peculiar behavior of Tucker in Figure 4.1c is due to the fact that Tucker tends to score many triples (both positive and negative triples) very highly for specific relations, such as *hasGender* and *isLocatedIn*. Moreover, as we make the negative sampling more challenging, we see extremely different behavior from the models, some result in a much more calibrated plot compared to others (e.g., RotatE looks calibrated for *Careful-N*, but not for the rest), making these benchmarks inconclusive for calibration. For WN18RR and FB15k-237, although DistMult outperforms the other two methods completely, we observe similar behaviors in calibration plots. Last, these plots also indicate that the models are under-confident, which is inconsistent with similar studies on neural networks [33]. For a comparison that takes the model complexity into account, we include results for models that have the same number of parameters in Appendix B.

4.3.3 Simple Models Look Accurate

In this section, we evaluate the reasoning capabilities of current link prediction methods. More specifically, we wanted to see how far in performance on ranking metrics we can get to by adopting very simple methods and see if ranking metrics can properly differentiate between SOTA models and these simple approaches. We first study rule-based methods that only predict ranking for triples that have their semi-inverse relations in the training data. Then, introducing a *local score* that learns simple neighborhood patterns, we see unexpectedly high performance on ranking metrics, casting doubt on the capability of ranking metrics to accurately evaluate KGC methods.

Rule-based Link Prediction: To see the effect of semi-inverse relations on the performance of link prediction methods, we provide a very simple rule-based method. For WN18RR and YAGO3-10 target triples, we identify all objects for which the target subject appears with a semi-inverse relation in the training data, and vice versa for the subjects. Then we rank these entities based on their popularity (their degree in the graph) in the KG. The result of this rule-based method is provided in Table 4.1. As shown, for both of YAGO3-10 and WN18RR, this method achieves high performance.

Local Score: We also study an alternate, simple model, using just the local structure around the target triple. For each target triple, we compute a *local score* by finding all paths from the subject to object and score them in the context of the relation of the target triple (a simpler version of this local score is studied in [98]). Specifically, we define the local score as: $\mathcal{L}oc(s, r, o) = \sigma(\sum_{p \in P(s,o)} W_r^p)$, where $P(s, o)$ denotes the set of all the paths between s and o. To learn W_r^p we generate negative samples by randomly corrupting the r. A visual representation of the local score is provided in Appendix B. For FB15K-237 that is denser than WN18RR and YAGO3-10, for each relation, we consider the top 5 most frequent paths with length 2 between the subject and the object of triples with that relation[1]. For WN18RR and YAGO3-10, since most of the triples do not have paths with length 2 between their entities, we only score simple patterns with length 3 in our model. More specifically, these paths comprise of patterns that have one edge with the same relation as the target sample (with the same direction). Further, they should have the same relation for the other two edges but in a different direction. More details and visualization of these patterns are provided in Appendix B. The result of link prediction on our benchmarks is provided in Table 4.1. As it shows, in all three KGs, our local score performs comparably to embedding methods while having a much fewer number of parameters. The high performance of both these simple models raises questions about the utility of ranking metrics and existing benchmarks.

[1]Around 70% the test triples have at least one of these patterns in their neighborhood

Models	**FB15k-237**			**WN18RR**			**YAGO3-10**		
	MRR	Hits@1	# Param	MRR	Hits@1	# Param	MRR	Hits@1	# Param
DistMult	0.295	19.8	5.8M	0.428	39.2	8.1M	0.409	31.2	24.6M
RotatE	0.331	23.4	29.3M	0.478	43.4	40.9M	0.471	38.2	123.2M
Tucker	0.342	25.1	11M	0.456	42.8	9.4M	0.468	37.9	63.9M
Rule-Based	-	-	-	0.338	32.1	-	0.286	24.2	-
Local	0.181	12.7	0.3M	0.364	33.4	2k	0.322	25.8	50k

Table 4.1: **Link Prediction** result for FB15k-237, WN18RR, and YAGO3-10 KGs. All results are generated using perspective models' SOTA hyperparameters.

Models	**FB15k-237**		**WN18RR**		**YAGO3-10**	
	Random-N	Careful-N	Random-N	Careful-N	Random-N	Careful-N
DistMult	95.2	47.6	83.3	39.5	94.9	45.5
RotatE	94.4	49.1	84.8	42.0	86.1	42.9
Tucker	77.6	57.4	72.0	55.8	75.4	45.2

Table 4.2: Triple classification accuracy for random and careful negative sampling.

4.3.4 Problems with Triple Classification with Negative Sampling

To demonstrate that the current approach to evaluating accuracy via triple classification is inaccurate, we create a fact classifier from completion models (that only provide a score) by learning a threshold for each relation (following standard practice). The performance of state-of-art embedding models over several KGs is provided in Table 4.2. As it shows, all the models achieve very high accuracy performance (around 80%) when we choose both validation and test negative samples randomly (*Random-N*). The reason behind this high performance is mostly due to the naive way of random negative sampling. Moreover, Distmult and RotatE outperform Tucker's performance in all three KG (although Tucker often achieves higher or very similar performance on ranking metrics).

Negative Sampling: There are a few fundamental issues with the current approach to triple classification: (1) randomly choosing the negative samples results in a very simple classification, which is not informative for evaluation, and (2) training classifiers (estimating thresholds) based on random negative samples makes the results brittle, i.e., choosing slightly more challenging negative samples can reduce the performance dramatically. Instead of random samples, we instead use the challenging *Careful-N* samples described in Section 4.3.2, and show results in Table 4.2. As it shows, these negative samples dramatically reduce the accuracy. Further, although Tucker performs the worst in the random setting, here we see a smaller reduction in accuracy compared to the other two methods (RotatE appears better than DistMult). We suspect the reason behind this smaller reduction in accuracy is that Tucker distinguishes between positive and hard negative samples better, i.e., on average, assigns considerably higher scores to positive samples in comparison to hard negatives.

Mismatch between Ranking Metrics and Accuracy: We also evaluate these models on triple classification in Table 4.3 for Kinship and Nations, which have *all* the true facts available (all missing facts are false, and can be enumerated). As the negative samples, we consider the union of the top 10 negative objects and subjects based on trained RotatE and Tucker models. Although these models achieve around 0.8 MRR and 100% Hits@10 performance, they demonstrate much lower performance on accuracy metrics showing that these ranking metrics are not trustworthy. Moreover, if we classify the negative and positive samples for Kinship and Nations KGs only based on the compatibility of their subject and object with the relation, based on our defined notion of type (*Type Constraint*), we see that *Type Constraint* achieves comparable results with these embedding methods, questioning the credibility of the ranking metrics and performance of these embedding methods.

Models	**Kinship**				**Nations**			
	Acc	F1	Recall	Precision	Acc	F1	Recall	Precision
Distmult	58.8	7.6	34.8	4.3	86	24	25.6	22.9
RotatE	10.6	10	97.3	5.3	66.9	27.2	69.6	16.9
Tucker	86.2	38.8	83.7	25.2	55.6	18.9	66.6	11
Type Constraint	28.9	12.0	94.6	6.4	47.6	22.8	87.0	13.1

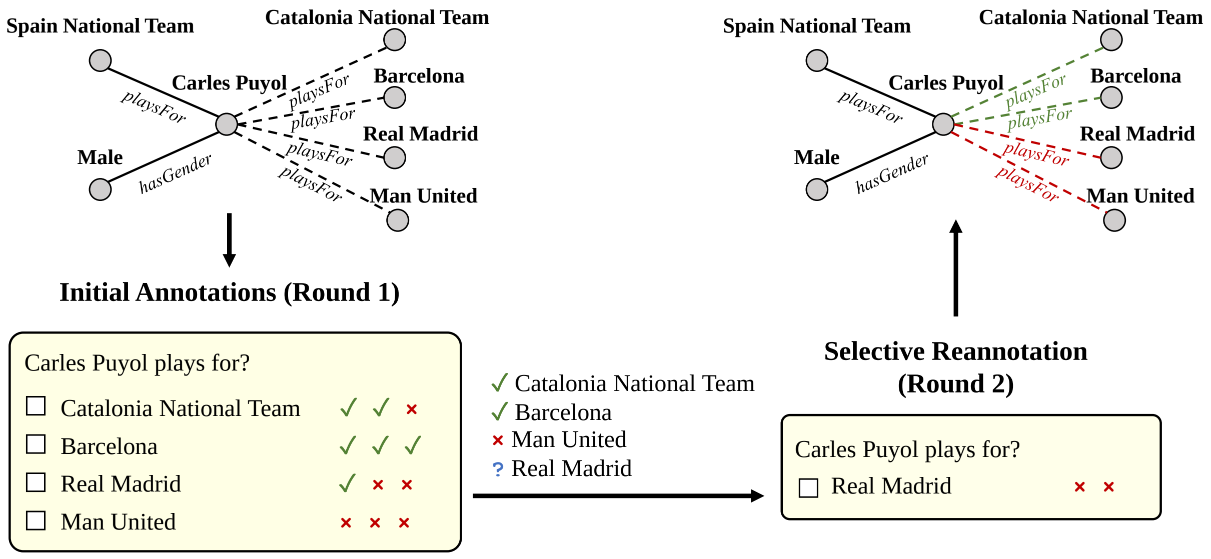
Table 4.3: Triple classification accuracy on ground truth labels. The results are averaged over 5 runs.

4.4 YAGO3-TC: A New Benchmark for Evaluating KG Completion

In this section, we first describe our procedure to gather YAGO3-TC dataset, and then, we explain our plans to continuously update YAGO3-TC as new KG completion models are proposed.

4.4.1 Creating YAGO3-TC

To solve these issues with KG completion evaluation, we gather a dataset that contains true and false facts, but is also challenging for current models. Note that in this chapter, we are not suggesting that we should completely replace the ranking metrics for all use cases, but point out their shortcomings, and introduce a benchmark to compute other metrics. Since each embedding model scores triples differently, we use RotatE [94] and Tucker [7] as our judges for identifying important triples. More specifically, we first sample 1000 random triples from the test set of YAGO3-10 (the relation distribution histogram of YAGO3-10 test data and our 1000 sampled triples is very similar, as provided in Appendix B). To reduce the effect of semi-inverse relations, we do not consider triples with relation *isAffiliatedTo* in our samples. Then, applying trained RotatE and Tucker on these triples, we find the 10



(a) Overview of crowdsourcing process.

	# Test	# Valid
Triples	28,364	2,946
Positives	2,976	223
Negatives	25,388	2,723

(b) Data Statistics.

Figure 4.2: **YAGO3-TC Dataset.** (a) annotation process, and (b) statistics of the resulting data

top scoring objects for the query $\langle s, r, ?\rangle$ and 10 top scoring subjects for $\langle ?, r, o\rangle$. Excluding repeated triples, we gathered 28,364 triples/facts.

We need to label these triples as negative (false) and positive (true). Before conducting crowd-sourcing to label these triples, there are a few criteria to identify the true label of some of these triples: (1) if the relation of the target triple is N-1 (or 1-N) we can treat every object (or subject) as a negative sample, except for the original target object (or subject), and (2) if the object (subject) of a sample does not have the same type as the object (subject) of the original test triple, we can treat that sample as negative. Filtering these identifiable samples, we label the rest through crowd-sourcing.

For labeling the samples, we ask the users to search the information on the Wikipedia page of the entities and use the Google search engine. For example, we ask users to choose all the correct teams for the query “Carles Puyol plays for?” from provided options. We use Amazon Mechanical Turk as the framework for gathering the labels and ask three users to answer each query. If more than one user agree on an answer, we treat that triple as a positive one. We separately reannotate the objects that were picked only by one of the users. This time, we ask two users to check these samples, and if both of them agree on a correctness of a choice, we treat it as a positive sample. After creating the set of positive labels, we treat everything else as negative samples. An overview of our user study is depicted in Figure 4.2a. Further, after randomly choosing 100 samples from the validation data of YAGO3-10, we use the same procedure for gathering labels for our validation data. Since we intend to use these labels to find the thresholds of the models, we ensure that at least one triple for each relation that appears in this set. The dataset statistics are provided in Table 4.2b. To check the quality of our labels, we also include 100 true facts from the original test data in our study, and find that 96% of these triples were annotated to be positive, demonstrating the high quality of our labels.

4.4.2 Continuously Updated, Hidden Benchmark

Although we are confident about the quality of the true/false annotations, we select the candidates based on the output of the two recent models, RotatE and Tucker. As new models will be proposed, they may be able to differentiate between our gathered true and false facts, but may highly rank facts that are not in our dataset. In order to maintain a benchmark that is useful for evaluating knowledge base completion in the long run, we propose a web-hosted evaluation platform. The online platform, available at `https://pouyapez.github.io/yago3-tc/` , includes a *hidden* test set, and a leaderboard of model submissions. The test set, initialized with the YAGO3-TC dataset described here, will continuously be updated as we receive model predictions from the submissions, thus identifying *more challenging* candidate facts, via the same crowdsourcing pipeline described above.

4.5 Evaluation Using YAGO3-TC

In this section, we investigate the triple classification evaluation using our new dataset. We first study the performance of several embedding models on the dataset and introduce new simple techniques to improve current models. Then, comparing the calibration of the triple classification task with the ranking scenario, we demonstrate this task is better defined. Finally, we study the per-relation breakdown of accuracy to better assess model performance.

4.5.1 Performance of Existing KGC Models on YAGO3-TC

To provide a better evaluation procedure for KG completion, we study accuracy metrics on our gathered data. The averaged result of SOTA methods in YAGO3-TC over 5 runs is provided in Table 4.3a. As it shows, except for recall, Tucker outperforms RotatE. We note that, in this experiment, accuracy and precision are the metrics we care about the most because it is important to avoid labeling a triple as positive incorrectly. Moreover, comparing the results with ranking metrics in Table 4.1, we see a significant mismatch between these metrics and the ranking ones.

We also consider 3 baselines for our benchmark, 1) randomly assigning labels using the actual ratio of positives and negatives as the probability, 2) classifying based on the compatibility of the type of subject and the object with the relation, and 3) training our classifier using the scores of our local models from Section 4.3.3. All of these baselines achieve comparable results with embedding methods demonstrating the need for better training and models. We are also interested in investigating whether we can improve these performances with

Relation	**DistMult**				**RotatE**				**Tucker**			
	Acc	F1	R	P	Acc	F1	R	P	Acc	F1	R	P
`playsFor`	25.9	23.2	85.6	13.4	20.6	22.8	89.8	13	73.5	29.1	41.6	22.4
`isLocatedIn`	35.4	23.2	83.8	13.5	21.7	20.3	85.6	11.5	45	23.7	73.4	14.1
`wasBornIn`	22.9	5.5	75.3	2.8	15.3	5.6	84.3	2.9	62.4	3.6	23.4	1.9
`hasGender`	78.4	32.4	92.6	19.7	94.7	45.3	38.9	54.4	97.9	82.2	85.2	79.4

Table 4.4: Per-Relation Breakdown

simple modifications on the learning process. First, instead of random negative sampling on validation data, use our gathered validation data as the training data for triple classification (*model-valid*). As shown, upon training on our validation data, the accuracy and precision increase dramatically, and recall drops with a huge gap. To provide a deeper understanding of the performance, a per-relation breakdown is provided in Appendix B.

4.5.2 Calibration

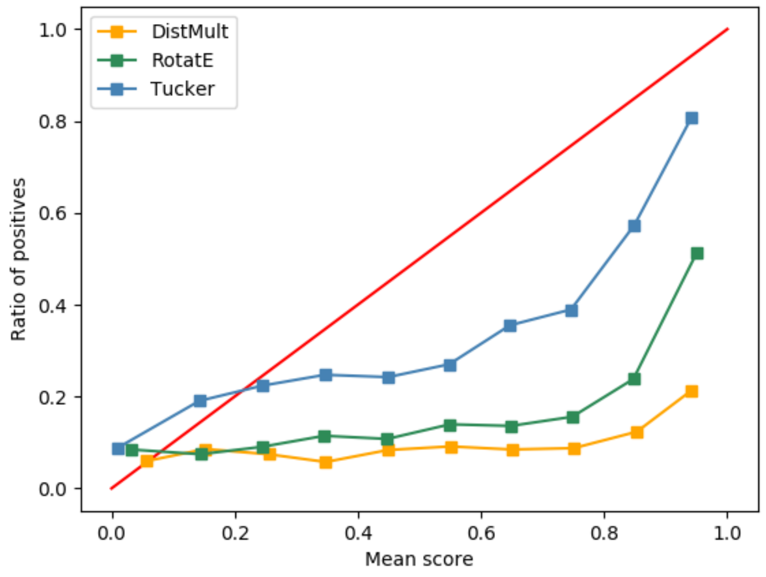
As we showed, the calibration study on existing evaluation metrics is not a well-defined task. YAGO3-TC provides us with an opportunity to study calibration in a more controlled and representative environment. The evaluation of the calibration of YAGO3-TC is depicted in Figure 4.3b, with the histogram plot of the scores in Appendix B. As shown, Tucker provides a more calibrated plot compared to RotatE. Moreover, previous calibration curves suggested models are under-confident, whereas here, the calibration reveals that they are overconfident, which is consistent with calibration studies on neural network models [33].

4.5.3 Per-Relation Breakdown

We perform a per-relation breakdown analysis on the YAGO3-TC dataset to gain a deeper understanding of how is the distribution of the model's performance on different relations.

Models	Acc	F1	R	P	A-ROC
DistMult	29.4	20.4	**86.6**	11.6	0.61
RotatE	27.0	19.4	83.7	10.9	0.58
Tucker	63.3	**22.3**	50.3	14.4	**0.64**
DistMult-valid	85.6	19.1	14.3	29.1	0.59
RotatE-valid	**88.6**	18.9	12.8	**42.1**	0.61
Tucker-valid	79.7	22.1	27.5	18.5	0.56
Random	80.9	10.9	11.1	10.6	0.51
Type Constraint	32.2	20.8	84.8	11.8	0.61
Local	61.0	19.0	43.8	12.2	0.6

(a) Triple classification accuracy on ground truth labels. The results are averaged over 5 runs.



(b) Calibration plot for YAGO3-TC.

Figure 4.3: **Triple classification on YAGO3-TC.** (a) provides average performance of embedding methods and our baselines. (b) Depicts the calibration study of embedding models.

This kind of analysis can help us identify the shortcoming and the strength of our embedding methods. Table 4.4 compares RotatE and Tucker on the top four most frequent relations. As shown, RotatE outperforms Tucker in recall except for relation `hasGender`, and loses except for F1 and precision for relation `wasBornIn`. Relations `playsFor` and `isLocatedIn` show similar performance over all metrics in RotatE (and almost Tucker), demonstrating that these models learn similar pattern for these relations. Moreover, both models perform very poorly in relation `wasBornIn`, suggesting the difficulty in predicting this type of relation. While both models predict the relation `hasGender` with much more confidence, emphasizing the simplicity in the prediction of this relation.

4.6 Related Work

There is a rich literature on representing knowledge bases using fixed-size embedding vectors.

In the past few years, a number of recent techniques have proposed models that firstly assign an embedding for each entity and relation, and then use these embeddings to predict facts. These methods, which primarily only differ in scoring function for link prediction task, include tensor multiplication [61, 89, 108, 7], algebraic operations [14, 13, 23, 94], and complex neural models [25, 58]. Furthermore, a number of studies have examined the incorporation of extra types of evidence to achieve more informative embeddings, with extra modalities consisting of numerical values [29], images [62], text [99, 100, 102], and their combinations [65]. Utilizing the analysis in this chapter, we hope to shed more light on better integrating extra modalities in the vector space to have more informed embeddings.

Although these methods provide accurate models for a variety of KG tasks, only a few try to provide a better understanding for these models, such as by addressing issues in training [40, 38, 83], investigating particular triples in the data [3], studying sparsity and unreliability of KGs [75], analyzing interpretability in the embedding space [87, 70, 4], and identifying existing issues in KG completion models [95]. Although [97] also study calibration of link prediction models, there are several differences between our study on calibration: 1) we show the effect of different negative sampling procedures on the calibration of link prediction methods, and 2) we further provide a well-defined and explicit environment for calibration study using our proposed YAGO3-TC. Moreover, [84] studies the utility of KG embedding methods in real-world completion tasks by proposing to calibrate these embedding models to output reliable confidence estimates for predicted triples. It is worth mentioning that developing more appropriate and challenging datasets as a way to address shortcomings of existing benchmarks has been used in other machine learning tasks, such as visual reasoning [39, 46], semantic parsing, [57] and textual entailment [110], amongst others.

4.7 Conclusion

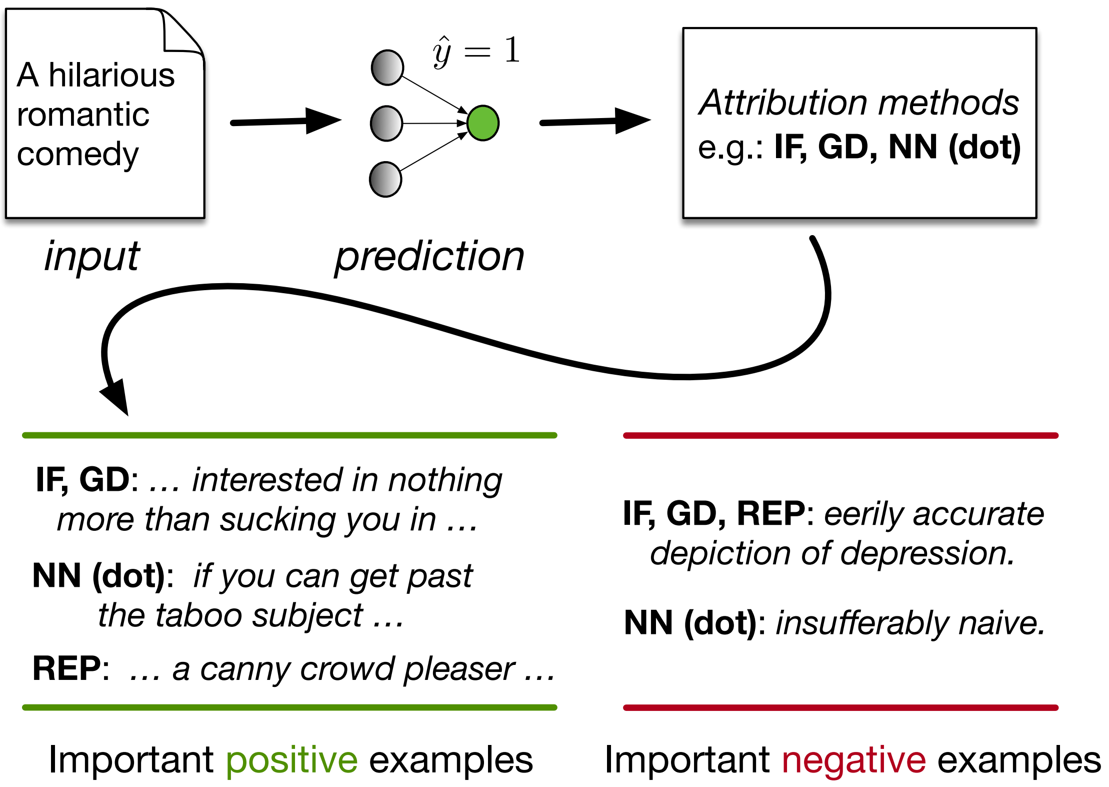
In this chapter, we set to investigate whether ranking metrics are appropriate measures to evaluate link prediction models. Upon studying the shortcomings and strengths of the current adopted procedure, we first show existing issues with ranking metrics: they do not evaluate completion, are difficult to use for calibration, and are not able to consistently differentiate between different models. Facing these issues, after redefining the triple classification task, we gather a new dataset YAGO3-TC consisting of a dense subgraph annotated with both true and false facts. Exploring several SOTA embedding models on this dataset, we further provide insights and directions for future works. We hope that this research and dataset will bridge the gap in the better adoption of link prediction models in real-world scenarios. The datasets, leaderboard with continuously updated benchmark, and the open-source implementation of the models are available at `https://pouyapez.github.io/yago3-tc/`. We hope this annotation methodology is used for existing and future evaluation benchmarks in KG completion.

Chapter 5

Which Training Samples are Truly Important?

5.1 Introduction

Interpretability methods are intended to help users understand model predictions [80, 49, 96, 31]. In machine learning broadly and NLP specifically, such methods have focused on feature-based explanations that highlight parts of inputs 'responsible for' the specific prediction. Feature attribution, however, does not communicate a key basis for model outputs: training data. Recent work has therefore considered methods for surfacing training examples that were influential for a specific prediction [44, 109, 70, 19, 8, 36]. While such *instance-attribution* methods provide an appealing mechanism to identify sources that led to specific predictions (which may reveal potentially problematic training examples), they have not yet been widely adopted, at least in part because even approximating influence functions [44]—arguably the most principled attribution method—can be prohibitively expensive in terms of computation. Is such complexity necessary to identify 'important' training points?



A hilarious romantic comedy
input
$\hat{y} = 1$
prediction
Attribution methods e.g.: IF, GD, NN (dot)
IF, GD: … interested in nothing more than sucking you in …
NN (dot): if you can get past the taboo subject …
REP: … a canny crowd pleaser …
IF, GD, REP: eerily accurate depiction of depression.
NN (dot): insufferably naive.
Important positive examples
Important negative examples

Figure 5.1: Attribution methods score train examples in terms of their importance to a particular prediction. In this chapter, we compare several such methods, e.g., Influence Functions (IF) and its variants (GD), Representer Points (REP), and similarity measures (NN).

Or do simpler methods (e.g., attribution scores based on similarity measures between train and test instances) yield comparable results? In this chapter, we set out to evaluate and compare instance attribution methods, including relatively simple and efficient approaches [78] in the context of NLP (Figure 5.1). We design qualitative evaluations intended to probe the following research questions: (1) How correlated are rankings induced by gradient and similarity-based attribution methods (assessing the quality of more efficient approximations)? (2) What is the quality of explanations in similarity methods compared to gradient-based ones (clarifying the necessity of adopting more complex methods)?

We evaluate instance-based attribution methods on two datasets: the binarized version of the Stanford Sentiment Treebank (SST-2; [90]) and the Multi-Genre NLI (MNLI) dataset [106]. We investigate the correlation of more complex attribution methods with simpler approximations and variants (with and without the use of the Hessian). Comparing explanation quality of gradient-based methods against simple similarity retrieval using leave-one-out [9] and randomized-test [37] analyses, we show that simpler methods are fairly competitive. Finally, using the HANS dataset [53], we show the ability of similarity-based methods to surface artifacts in training data.

5.2 Attribution Methods

Similarity Based Attribution Consider a text classification task in which we aim to map inputs x_i to labels $y_i \in Y$. We will denote learned representations of x_i by f_i (i.e., the representation from the penultimate network layer). To quantify the importance of training point x_i on the prediction for test target sample x_t, we calculate the similarity in embedding space induced by the model.[1] To measure similarity, we consider three measures:

[1]To be clear, there is no guarantee that similarity reflects 'influence' at all, but we are interested in the degree to which this simple strategy identifies 'useful' training points, and whether the ranking implied by this method over train points agrees with rankings according to more complex methods.

Euclidean distance, *Dot* product, and *Cosine* similarity. Specifically, we define similarity-based attribution scores as: **NN EUC** $= -\|f_t - f_i\|^2$, **NN COS** $= \cos(f_t, f_i)$, and **NN DOT** $= \langle f_t, f_i \rangle$.

To investigate the effect of fine-tuning on these similarity measures, we also derive rankings based on similarities between untuned sentence-BERT [79] representations.

Gradient Based Attribution *Influence Functions (IFs)* were proposed in the context of neural models by [44] to quantify the contribution made by individual training points on specific test predictions. Denoting model parameter estimates by $\hat{\theta}$, the IF approximates the effect that upweighting instance i by a small amount—ϵ_i—would have on the parameter estimates (here H is the Hessian of the loss function with respect to our parameters): $\frac{d\hat{\theta}}{d\epsilon_i} = -H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(x_i, y_i, \hat{\theta})$. This estimate can in turn be used to derive the effect on a specific test point x_t: $\nabla_\theta \mathcal{L}(x_t, y_{\text{test}}, \hat{\theta})^T \cdot \frac{d\hat{\theta}}{d\epsilon_i}$.

Aside from IFs, we consider three other similar gradient-based variations:

(1) RIF $= \cos(H^{-\frac{1}{2}} \nabla_\theta \mathcal{L}(x_t), H^{-\frac{1}{2}} \nabla_\theta \mathcal{L}(x_i))$.

(2) GD $= \langle \nabla_\theta \mathcal{L}(x_t), \nabla_\theta \mathcal{L}(x_i) \rangle$, and

(3) GC $= \cos(\nabla_\theta \mathcal{L}(x_t), \nabla_\theta \mathcal{L}(x_i))$.

RIF was proposed by [8], while GD and GC by [19].

Representer Points (REP; [109]) was introduced to approximate the influence of training points on a test sample by defining a classifier as a combination of a feature extractor and a ($L2$ regularized) linear layer: $\phi(x_i, \theta)$. [109] showed that for such models, the output for any target instance x_t can be expressed as a linear decomposition of "data importance" of training instances:

$$\phi(x_t, \theta^*) = \sum_i^n \alpha_i f_i^\top f_t = \sum_i^n k(x_t, x_i, \alpha_i) \tag{5.1}$$

where $\alpha_i = \frac{1}{-2\lambda_n} \frac{\partial \mathcal{L}(x_i, y_i, \theta)}{\partial \phi(x_i, \theta)}$.

5.3 Experimental Setup

Datasets To evaluate different attribution methods, we conduct several experiments on sentiment analysis and NLI tasks, following prior work investigating the use of IF specifically for NLP [36]. We adopt a binarized version of the Stanford Sentiment Treebank (SST-2; [90]), and the Multi-Genre NLI (MNLI) dataset [106]. For fine-tuning on MNLI, we randomly sample 10k training instances. Finally, to evaluate the ability of instance attribution methods to reveal annotation *artifacts* in NLI, we randomly sampled 1000 instances from the HANS dataset (more details in Appendix C).

Models We define models for both tasks on top of BERT [26], tuning hyperparameters on validation data via grid search. Our models achieve 90.6% accuracy on SST and 71.2% accuracy on MNLI (more details in Appendix C).

Computing the IF for BERT Deriving the IF for all parameters θ of a BERT-based model requires deriving the corresponding Inverse Hessian. We compute the Inverse Hessian Vector Product (IHVP) $H^{-1}\nabla_\theta \mathcal{L}(x, y, \theta)$ directly because storing the entire matrix of $|\theta|^2$ elements is practically impossible (requiring ∼12 PB of storage). We approximate the IHVP using the LiSSa algorithm [2] (was introduced as a second-order stochastic method for solving optimization problems in machine learning). This method is still expensive to run and is sensitive to the norm of the IHVP approximation. Therefore, for computational reasons we consider IF with respect to the subset of parameters that correspond to the top five layers [IF (Top-5)], and only the last linear layer [IF (linear)], resulting in a few orders of magnitude faster procedure (the algorithm becomes increasingly unstable as we incorporate additional layers). We also use a large scaling factor to aid convergence.

5.4 Experiments

In this section, we first investigate the correlation between different methods. Then, to study the quality of explanations we conduct leave-some-out experiments, and further analyze attribution methods on HANS data. We consider five evaluations (more analyses and experimental details in Appendix C).

(1) Calculating the *correlation* of each pair of attribution methods, assessing whether simple methods induce rankings similar to more complex ones.

(2) *Removing the most influential samples* according to each method, retrain, and then observe the change in the predicted probability for the originally predicted class, with the assumption that more accurate attribution methods will cause more drop.

(3) We follow *randomized-test* from [37] and measure the ranking correlation of methods for (a) randomly initialized and (b) trained models, under the assumption that high correlation here would suggest less meaningful attribution.

(4) We measure the degree to which the methods recover examples that exhibit *lexical overlap* when tested on the HANS dataset [53]. This extends a prior analysis of IF [36], considering alternative attribution methods.

(5) We measure the accuracy of different methods in extracting the training sample as the explanation when the target sample is very close to aforementioned training sample.

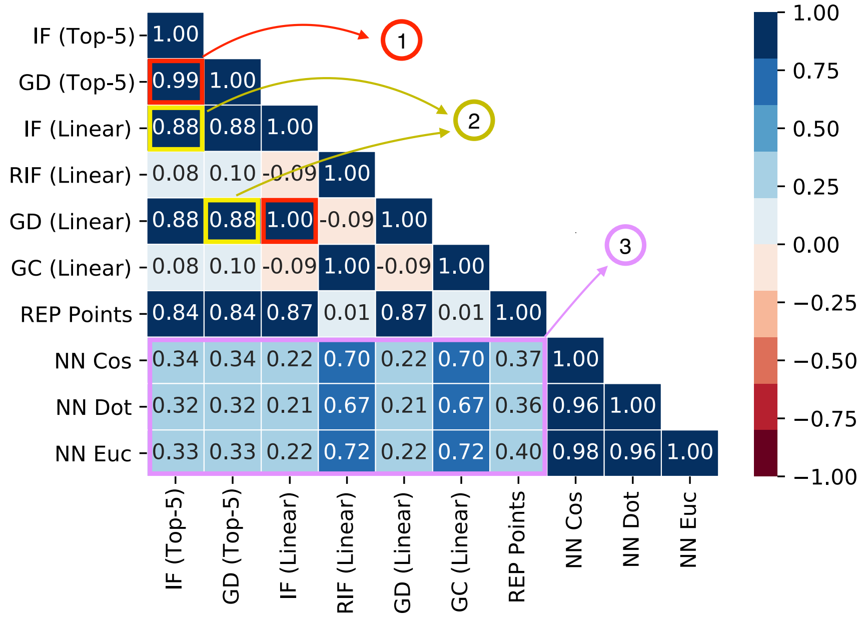
5.4.1 Attribution Methods' Correlation

We calculate the Spearman correlation between scores assigned to training samples by different methods, allowing us to compare their similarities. More specifically, we randomly sample 100 test and 500 training samples from datasets and calculate the average resultant Spearman correlations.

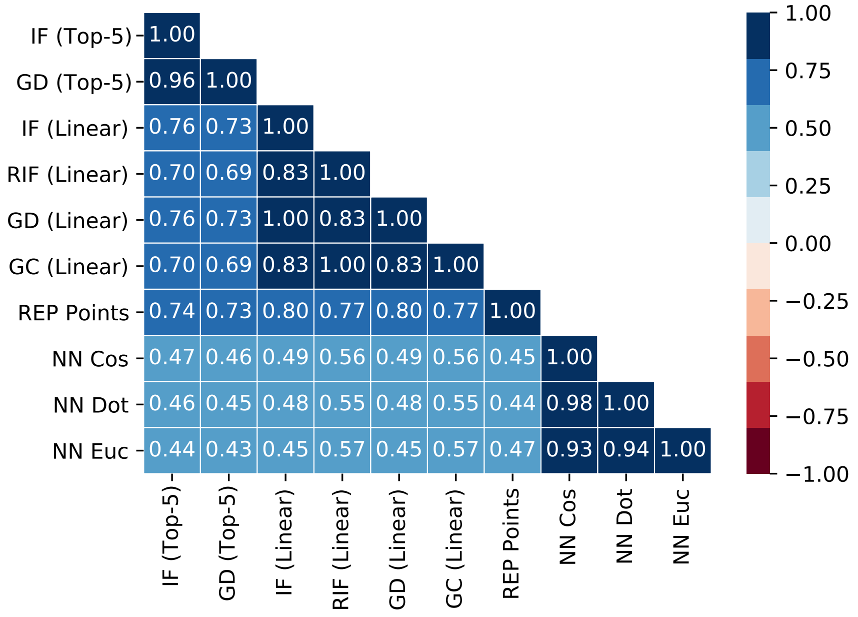
We report attribution methods' correlation on SST and MNLI datasets in Figure 5.2 (a more complete version of these figures is in Appendix C). We make the following observations. (1) Gradient methods w/wo normalization appear similar to each other, e.g., GC is similar to RIF and IF is similar to GD, suggesting that Hessian information may not be necessary to provide meaningful attributions (GD and GC do not use the Hessian). (2) There is a high correlation between IF calculated over the top five layers of BERT and IF over only the last linear layer. (3) There is only a modest correlation between similarity-based rankings and gradient-based methods, suggesting that these do differ in terms of the importance they assign to training instances. We report a proportion of common top examples between IF (Top-5) and IF (Linear) in Appendix C, providing further evidence of the high correlation between these methods.

5.4.2 Removing 'Important' Samples

In Table 5.1 we report the average results of removing the top-k most important training samples for 50 random test samples using different attribution methods. We only consider the linear version of methods in the remainder of the chapter. All methods seem effective,



(a) Spearman Correlation on SST.



(b) Spearman Correlation on MNLI

Figure 5.2: The similarity between influence of training samples for different pairs of attribution methods on the SST and MNLI datasets was measured via Spearman Correlation. ① = Using Hessian does not change the ordering of training examples. ② = Using more layers of BERT in IF approximation does not much affect the ordering. ③ = NN metrics are not well correlated with gradient-based ones.

	Method	avg(Δ)-SST		avg(Δ)-MNLI		Spearman	
		Remove-50	Remove-500	Remove-50	Remove-500	SST	MNLI
	Random (50 runs)	-0.028	-0.021	-0.039	-0.029	-	-
Similarity	NN EUC	-0.028	**-0.540**	-0.102	-0.266	0.056	0.023
	NN COS	-0.072	-0.430	-0.088	-0.306	0.045	0.018
	NN DOT	-0.059	-0.513	**-0.106**	-0.273	0.005	-0.002
Gradient	IF	-0.054	-0.526	-0.042	-0.407	-0.296	0.018
	REP	**-0.114**	-0.490	-0.002	-0.230	-0.217	0.053
	RIF	-0.071	-0.537	-0.068	-0.347	-0.021	0.013
	GD	-0.058	-0.516	-0.022	**-0.446**	-0.290	0.017
	GC	-0.082	-0.528	-0.030	-0.279	-0.021	0.012

Table 5.1: Average difference (Δ) between predictions made after training on (i) all data and (ii) a subset in which we remove the top-50/top-500 most important training points, according to different methods (Random on both of the benchmarks has standard deviation around 0.02). We also report the Spearman correlation between the ranking induced by each approach using a trained model and the same ranking when a randomly initialized model is used.

compared to random sampling. Perhaps surprisingly, for both tasks at least one of the similarity-based approaches performs comparably or better than gradient-based methods, in the sense that removing the top examples according to similarity yields reductions in the predicted probability (which is what one would intuitively hope). Finally, it seems that the models applying some form of normalization to the gradient (i.e., RIF and GC) perform more consistently. This is consistent with contemporaneous work of [37] which argues that this is a consequence of large gradient magnitudes for some samples dominating when normalization is not used. Upon investigating high influential training samples, we observed that similarity-based approaches seem to yield more diverse "top" instances compared to gradient-based ones. We also found that normalization in gradient-based methods made a large difference. Generic IF-based ranking tends to be dominated by high loss training examples across test examples, whereas normalization provides more diverse top training examples. Further, proportions of shared top examples between methods is provided in Appendix C, clarifying their similar performance.

	Method	**Lexical Overlap Rate**	
		top-1	top-10
	Random	0.40	0.40
Sen-Bert	NN EUC	0.39	0.41
	NN COS	0.38	0.39
	NN DOT	0.39	0.40
Sim	NN EUC	**0.56**	**0.57**
	NN COS	**0.56**	0.56
	NN DOT	0.44	0.44
Gradient	IF	0.43	0.44
	REP	0.43	0.35
	RIF	0.55	0.56
	GD	0.43	0.44
	GC	0.55	0.56

Table 5.2: Average lexical overlap rate between premise and hypothesis in top-k most influential samples for test instances mispredicted as entailment.

5.4.3 Randomized-Test

We report the Spearman correlation between trained and random models for SST and MNLI data in Table 5.1. This would ideally be small in magnitude (non-zero values indicate correlation). Curiously, gradient-based methods (IF, REP, GD) exhibit negative correlations on the SST dataset. Overall, these results suggest that gradient-based approaches without gradient normalization may be inferior to alternative methods. The simple NN-DOT method provides the 'best' performance according to this metric.

5.4.4 Artifacts and Attribution Methods

To investigate whether attribution methods can correctly identify training samples with specific artifacts responsible for model predictions, we follow [36]: This entails randomly choosing 10k samples from MNLI and treating *neutral* and *contradiction* as a single *non-entailment* label for model fine-tuning. More specifically, we are interested in target samples that the model mispredicts as *entailment* because of the lexical overlap artifact (lexical overlap is an artifactual indicator of entailment; [53]).

The average lexical overlap rate for 1000 random samples from the HANS dataset is provided in Table 5.2. As a baseline, we also apply similarity-based methods on top of sentence-BERT embeddings, which as expected appear very similar to random correlation. One can observe that similarity-based approaches tend to surface instances with higher lexical overlap, compared to gradient-based instance attribution methods. Moreover, gradient-based methods without normalization (IF, GD, and REP) perform similar to selecting samples randomly and based on sentence-BERT representations, suggesting an inability to usefully identify lexical overlap.

5.5 Near Training Samples Explanations

To further investigate the quality of the most influential sample based on different attribution methods, we conjecture that a data point very similar to a training sample should recover that sample as the most influential instance. We consider four scenarios to create target points similar to training data: (1) using training samples themselves as the target instances for attribution methods; (2) adding a random token to a random place in each training sample; (3) randomly removing a token from each training sample, and; (4) replacing a random token in each training sample with a random token from the dictionary of tokens. In the MNLI dataset, we apply each modification to both the premise and hypothesis in each training sample.

The result of this analysis is provided in Tables 5.3 and 5.4. We observe that similarity-based methods demonstrate a greater ability to recover the original training samples corresponding to the different targets. Moreover, the very low performance of IF, GC, and REP methods

	Method	Train		ADD		Remove		Replace	
		HIT@1	HIT@10	HIT@1	HIT@10	HIT@1	HIT@10	HIT@1	HIT@10
Sim	NN EUC	100	100	99.9	100	66.5	73.7	99.9	100
	NN COS	100	100	99.8	100	67.3	74.6	99.8	100
	NN DOT	0.73	2.06	0.73	2.06	0.47	2.19	0.73	2.06
Gradient	IF	0.01	0.34	0.01	0.35	0.04	0.25	0.01	0.35
	REP	0.01	0.27	0.01	0.27	0.04	0.22	0.01	0.27
	RIF	95.8	96.0	95.9	96.0	65.0	72.2	95.8	96.0
	GD	0.01	0.38	0.01	0.38	0.04	0.23	0.01	0.37
	GC	95.9	96.0	95.9	96.0	65.3	72.3	95.9	96.0

Table 5.3: Treating the training samples and their modifications as the target samples for attribution methods over SST dataset.

	Method	Train		ADD		Remove		Replace	
		HIT@1	HIT@10	HIT@1	HIT@10	HIT@1	HIT@10	HIT@1	HIT@10
Sim	NN EUC	100	100	100	100	36.7	45.8	100	100
	NN COS	100	100	100	100	38.1	46.8	100	100
	NN DOT	1.30	6.44	1.30	6.44	3.49	10.7	1.30	6.44
Gradient	IF	0.0	0.01	0.0	0.01	0.02	0.10	0.0	0.01
	REP	0.0	0.01	0.0	0.01	0.01	0.09	0.0	0.01
	RIF	92.5	92.5	92.5	92.5	32.6	41.2	92.5	92.5
	GD	0.0	0.01	0.0	0.01	0.10	0.50	0.0	0.01
	GC	92.5	92.5	92.5	92.5	32.8	41.2	92.5	92.5

Table 5.4: Treating the training samples and their modifications as the target samples for attribution methods over MNLI dataset.

is due to the fact that there are training points with high magnitude gradient, which these methods choose as top instances for *any* target sample.

5.5.1 Computational Complexity

The computational complexity of IF-based instance attribution methods constitutes an important practical barrier to their use. This complexity depends on the number of model parameters taken into consideration. As a result, computing IF is effectively infeasible if we consider *all* model parameters for modern, medium-to-large models such as BERT.

If we only consider the parameters of the last linear layer—comprising $O(p)$ parameters—to approximate the IF, the computational bottleneck will be the inverse Hessian which can be approximated with high accuracy in $O(p^2)$. There are ways to approximate the inverse Hessian more efficiently [64], though this results in worse performance. Similarity-based measures, on the other hand, can be calculated in $O(p)$.

With respect to wall-clock running time, calculating the influence of a single test sample with respect to the parameters comprising the top-5 layers of a BERT-based model for SST classification running on a reasonably modern GPU[2] requires $\sim$5 minutes. If we consider the linear variant, this falls to < 0.01 seconds. Finally, similarity-based approaches require < 0.0001 seconds. Extrapolating these numbers, it requires about 6 days to calculate IF (top-5 Layer) for all 1821 test samples in SST, while it takes only around 0.2 seconds for similarity-based methods.

5.6 Conclusions

Instance attribution methods constitute a promising approach to better understanding how modern NLP models come to make the predictions that they do [36, 44]. However, approximating IF to quantify the importance of train samples is prohibitively expensive. In this chapter, we investigated whether alternative, simpler, and more efficient methods provide similar instance attribution scores.

We demonstrated a high correlation between (1) gradient-based methods that consider more parameters [IF and GD (top-5)] and their simpler counterparts [IF and GD (linear)], and (2) methods without Hessian information, i.e., IF vs GD and RIF vs GC. We considered even simpler, similarity-based approaches and compared the importance rankings over training instances induced by these to rankings under gradient-based methods. Through leave-some-

[2]Maxwell Titan GPU (2015).

out, randomized-test, and artifact detection experiments, we demonstrated that these simple similarity-based methods are surprisingly competitive. This suggests future directions for work on fast and useful instance attribution methods. All code necessary to reproduce the results reported in this chapter is available at: `https://github.com/successar/instance_attributions_NLP`.

Chapter 6

Artifact Discovery with Attribution Methods

6.1 Introduction

Deep networks dominate NLP applications and are being increasingly deployed in the real-world. But what exactly are such models "learning"? One concern is that they may be exploiting *artifacts* or spurious correlations between inputs and outputs that are present in the training data, but not reflective of the underlying task that the data is intended to represent.

We assess the utility of *attribution methods* for purposes of aiding practitioners in identifying training data artifacts, drawing inspiration from prior efforts that have suggested the use of attribution methods for this purpose [36, 114]. Attribution methods are *model-centric*; our

Warning: This chapter contains examples with texts that might be considered offensive.

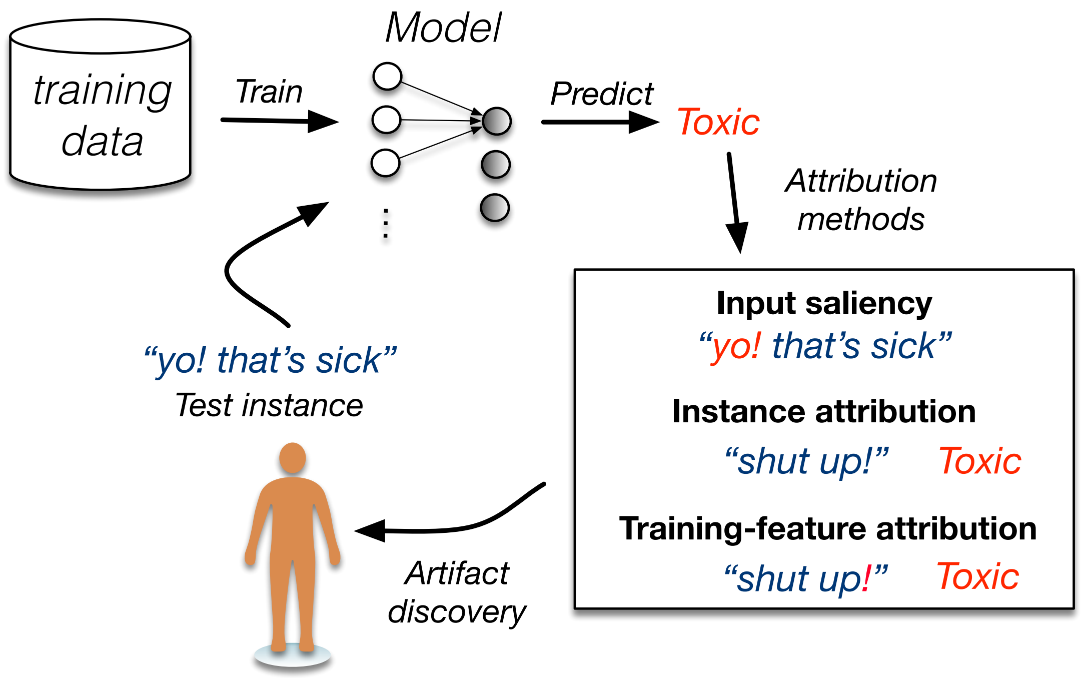


Figure 6.1: Use of different attribution techniques for artifact discovery in train data. Here attribution methods can reveal inappropriate reliance on certain tokens (e.g., *"!"*, *"yo"*) to predict Tweet toxicity; these are artifacts.

evaluation of them for artifact discovery therefore complements recent work on *data-centric* approaches [30]. We consider two families of attribution methods: (1) *feature-attribution*, which highlight constituent input features (e.g., tokens) in proportion to their "importance" for an output [80, 49, 1], and; (2) *instance attribution*, which retrieves training instances most responsible for a given prediction [44, 109, 78, 68].

We also introduce new hybrid attribution methods that surface relevant *features within train instances* as an additional means to probe what the model has distilled from training data. This addresses inherent limitations of using either feature or instance attribution alone for artifact discovery. The former can only highlight patterns within a given input, and the latter requires one to inspect entire (potentially lengthy) training instances to divine what might have rendered them influential.

Consider Figure 6.1. Here a model has learned to erroneously associate African American

Vernacular English (AAVE) with *toxicity* [86] and with certain punctuation marks ("!"). For a hypothetical test instance "yo! that's sick", both input saliency and instance attribution methods may provide some indication of these artifacts. But combining these via *training-feature attribution* (TFA) can directly surface the punctuation artifact by highlighting "!" within a relevant training example ("shut up!"); this is not readily apparent from either input or instance attribution. Our goal in this chapter is to evaluate TFA and other attribution methods as tools for identifying dataset artifacts.

Contributions. The main contributions of this chapter are as follows. (1) We propose a new hybrid attribution approach, training-feature attribution (TFA), which addresses some limitations of existing attribution methods. (2) We evaluate feature, instance and training-feature attribution for artifact detection on several NLP benchmarks with previously reported artifacts to evaluate whether and to what degree methods successfully recover these, and find that TFA can outperform other methods. We also discover and report previously unknown artifacts on a few datasets. Finally, (3) we conduct a small user-study to evaluate TFA for aiding artifact discovery in practice, and again find that combining feature and instance attribution is more effective at detecting artifacts than using either on its own.

6.2 Background and Notation

Assume a text classification setting where the aim is to fit a classifier ϕ that maps inputs $x_i \in \mathcal{X}$ to labels $y_i \in \mathcal{Y}$. Denote the training set by $\mathcal{D} = \{z_i\}$ where $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Each x_i consists of a sequence of tokens $\{x_{i,1}, \ldots, x_{i,n_i}\}$. Here we define a linear classification layer on top of BERT [26] as ϕ, fine-tuning this on $\mathcal{D}$ to minimize cross-entropy loss $\mathcal{L}$. Two types of attribution methods have been used in prior work to characterize the predictive behavior of ϕ.

Feature attribution methods highlight *important features* (tokens) in a test sample x_t. Examples of feature attribution methods include input gradients [96, 5], and model-agnostic approaches such as LIME [80]. In this chapter, we consider only gradient-based feature attribution.

Instance attribution methods retrieve training samples z_i deemed "influential" to the prediction made for a test sample x_t: $\hat{y}_t = \phi(x_t)$. Attribution methods assign scores to train instances z_i intended to reflect a measure of importance with respect to $\hat{y}_t$: $I(\hat{y}_t, z_i)$. Importance can reflect a formal approximation of the change in $\hat{y}_t$ when z_i is upweighted [44] or can be derived via heuristic methods [68, 78]. While prior work has considered these attribution methods for "train set debugging" [44, 36], this relies on the practitioner to abstract away potential patterns within the influential instances.

6.3 Artifact Detection and Training-Feature Attribution

6.3.1 What is an *Artifact*?

Models will distill observed correlations between training inputs and their labels. In practice, some of these correlations will be *spurious*, by which we mean specific to the training dataset used. Consider a particular feature function f such that $f(x)$ is 1 if x exhibits the feature extracted by f and 0 otherwise, a *training* distribution $\mathcal{D}$ over labeled instances z (often assembled using heuristics and/or crowdsourcing), and an ideal, hypothetical *target* distribution $\mathcal{D}*$ (the task we would actually like to learn; "sampling" directly from this is typically prohibitively expensive). Then we say that f is a *dataset* artifact if there exists a correlation between y and $f(x)$ in $\mathcal{D}$, but not in $\mathcal{D}*$. That is, if the mechanism by which one samples train instances induces a correlation between f and labels that would not be observed in an idealized case where one samples from the "true" task distribution.[1]

A given model may or may not exploit a particular dataset artifact; in some cases a *model-centered* view of artifacts may therefore be helpful. To accommodate this, we can extend our preceding definition by considering the relationship between model predictions $\hat{p}(y|x)$ and true conditional distributions $p(y|x)$ under D^*; we are interested in cases where the former differs from the latter due to exploitation of a dataset artifact f. Going further, we can ask whether this artifact was exploited *for a specific prediction*.

In this chapter we consider two types of artifacts. *Granular* input features refer to discrete units, such as individual tokens (this is similar to the definition of artifacts introduced in recent work by [30]). *Abstract* features refer to higher-level *patterns* observed in inputs, e.g., lexical overlap between the premise and hypothesis in the context of NLI [53].

6.3.2 Training-Feature Attribution

Showing important training instances to users for their interpretation places the onus on them to determine *what* was relevant about these instances, i.e., which features (granular or abstract) in x_i were influential. To aid artifact detection, it may be preferable to automatically highlight the tokens most responsible for the influence that train samples exert, communicating *what made an important example important.* This hybrid training-feature attribution (TFA) can reveal patterns extracted from training data that influenced a test prediction, even where the test instance does not itself exhibit this pattern, whereas feature attribution can only highlight features within said test instance. And unlike instance attribution, which retrieves entire train examples to be manually inspected (a potentially time-consuming and difficult task), TFA may be able to succinctly summarize patterns of influence.

[1]As a proxy for realizing this, imagine enlisting well-trained annotators with all relevant domain expertise to label instances carefully sampled i.i.d. from the distribution from which our test samples will actually be drawn in practice.

A high-level schematic of TFA is provided in Figure 6.2. We aim to trace influence back to features within training samples. We introduce training-feature attribution to extract influential features from training samples for a specific test prediction by considering a variety of combinations of feature and instance attribution and means of aggregating over these as TFA variants. For example, one TFA variant identifies features within the training point x_i that informed the prediction for a test sample z_t by taking the gradient of the influence with respect to inputs features, i.e., $\nabla_{x_i}\mathrm{I}(z_t, x_i, y_i)$ [44]. After calculating the importance of features within a train sample for a test target, we either construct a heatmap to help users identify *abstract* artifacts, or take aggregate measures over features (described below) to detect *granular* artifacts and present them to users.[2]

Heatmaps We present the top and bottom k influential examples to users with *token highlights* communicating the relative importance of tokens within these k influential train instances. This may allow practitioners to interactively, efficiently identify potentially problematic abstract artifacts.

Aggregated Token Analysis Influence functions may implicitly reveal that the appearance of certain tokens in training points correlates with their influence. We might directly surface this sort of pattern by aggregating TFA over a set of training samples. For example, for a given test instance, we can retrieve the top and bottom $k\%$ most influential training instances according to an instance attribution method. We can then extract the top token from each of these instances using TFA, and sort resulting tokens based on frequency, surfacing tokens that appear disproportionately in influential train points. Returning to toxicity detection, this might reveal that punctuation marks (such as "!") tend to occur frequently in influential examples, which may directly flag this behavior.

Discriminator One can also define model-based approaches to aggregate rankings of training

[2]Many other strategies are possible, and we hope that this work motivates further exploration of such methods.

points with respect to their influence scores. As one such method, we train a logistic regression (LR) model on top of Bag-of-Words representations to distinguish between the most and least influential examples, according to influence scores for a given test point. This will yield a weight for each token in our vocabulary; tokens associated with high weights are correlated with influence for the test point, and we can show them to the practitioner.

6.4 A Procedure for Artifact Discovery

We now propose a procedure (Figure 6.2) one might follow to systematically use the above attribution methods to discover training artifacts.

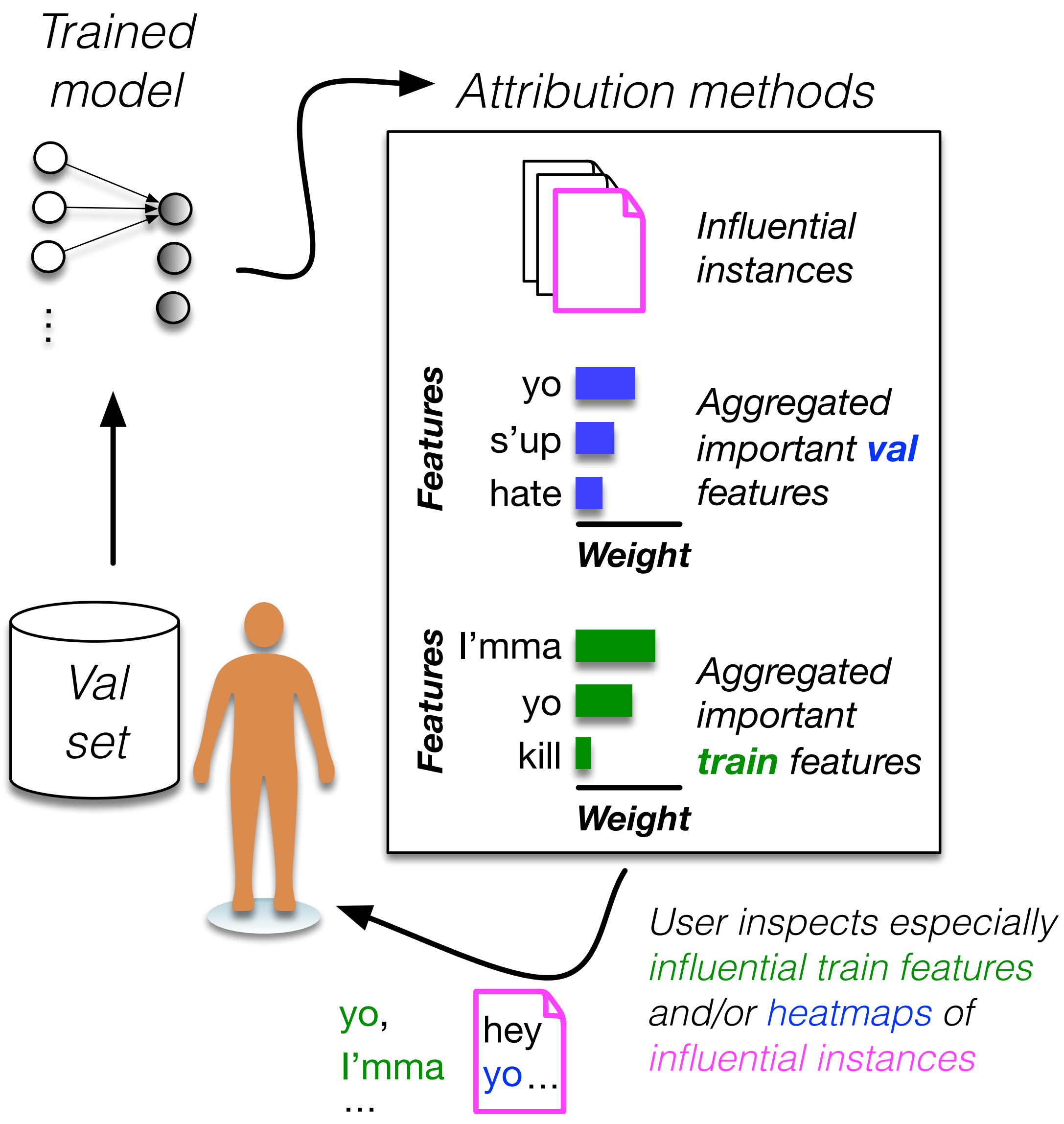
(1) Construct a validation set, either using a standard split, or by intentionally constructing a small set of "difficult" samples. Constructing a useful (for dataset debugging) such set is the biggest challenge to using attribution-based approaches.

(2) Apply feature-, instance-, and training feature attribution to examples in the validation set. Specifically, identify influential *features* using feature attribution or TFA and identify influential *training instances* using instance attribution.

(3-a) **Granular artifacts**: To identify granular artifacts, aggregate the important features from the test points (via feature attribution) or from influential train points (using TFA) for all instances in the validation set to identify features that appear disproportionately.

(3-b) **Abstract artifacts**: Inspect the "heatmaps" of influential instances for validation examples using one of the proposed TFA methods to deduce/identify abstract artifacts.

(4) Verify candidate artifacts by manipulating validation data and observing the effects on outputs.



Trained model
Attribution methods
Influential instances
Features
yo
s'up
hate
Weight
Aggregated important val features
Features
I'mma
yo
kill
Weight
Aggregated important train features
Val set
User inspects especially influential train features and/or heatmaps of influential instances
yo, I'mma ...
hey yo...

Figure 6.2: Finding artifacts via attribution methods. Staring from the validation set, we explain model prediction for every sample using different attribution methods. Then we either aggregate the explanations using frequency or rely on the heatmap analysis of explanations to detect artifacts.

We note that in 3-a, we aggregate the individual token *rankings* over all instances (for both feature attribution and TFA methods), which does not require thresholding attribution scores per instance. We now follow this procedure on widely used NLP benchmarks (Section 6.5), finding that we can "rediscover" known artifacts and identify new ones within these corpora (Section 6.6; Table 6.1).

6.5 Setup

Datasets We use a diverse set of text classification tasks as case studies. Specifically, we adopt: Multi-Genre NLI (MNLI; [106]); IMDB binary sentiment classification [50]; BoolQ, a yes/no question answering dataset [20]; and, DWMW17, a hate speech detection dataset [24].

Models We follow [68] for instance attribution methods; this entails only considering the last layer of BERT in our gradient-based instance attribution methods (see Appendix D). For all benchmarks, we achieve an accuracy within $\sim$1% of performance reported in prior works using BERT-based models.

Attribution Methods We consider two instance attribution methods, RIF [8] and Euclidean Similarity (EUC), based on results from [68]. For *Feature Attribution*, we consider Gradients (G) and Integrated Gradients (IG; [96]). To include RIF as a tool for artifact detection, we follow the TFA aggregated token approach, but assign uniform importance to all the tokens in a document.

In addition to the *model-centered* diagnostics we have focused on in this chapter, we also consider a few *dataset-centered* approaches for artifact discovery: (1) *PMI* [34], and (2) *competency* score [30]. There are a few inherent shortcomings to purely dataset-centered approaches. First, because they are model-independent, they cannot tell us whether a model is actually exploiting a given artifact. Second and relatedly, they are based on simple observed correlations between individual features and labels, so cannot reveal abstract artifacts. Given the latter point, we only consider these approaches for granular artifact detection (Section 6.6.1).

Challenges and Limitations A key computational challenge here is that instance attribution can be prohibitively expensive to derive if one uses *influence functions* directly [44, 36]. We address this by using efficient heuristic instance attribution strategies [68] to implement TFA. Since TFA combines existing feature- and instance-based attribution methods, training-feature attribution inherits known issues with these techniques [42, 9]. Despite such issues, however, our results suggest that TFA can be a useful tool for artifact discovery (as we will see next).

6.6 Case Studies

We now compare attribution methods in terms of their ability to highlight dataset artifacts. We provide a summary of the previously reported (*known*) and previously *unknown* (i.e., discovered in this chapter) artifacts we identify in this way (and with which methods) in Table 6.1.

6.6.1 Known Granular Artifact: Sentiment Analysis with IMDB Ratings

[81] observe that in the case of binary sentiment classification on IMDB reviews [50], numerical ratings (1 to 10) sometimes appear in texts. Modifying these in-text ratings often flips the

Dataset	**Artifact Type**	**Test Instance**	**Influential Train Instance**	**FA**	**IA**	**TFA**
IMDB	Ratings (K)	... great movie, 6/10.	... like it. Rating 8/10.	✓	✗	✓
HANS	Lexical Overlap (K)	**P**: The banker is in a tall building. **H**: the banker is tall	**P**: The red oak tree. **H**: Red oak yeah.	✗	✓	✓
DWMW	Punctuation (U)	Yo! just die.	Yo man! what's up.	✓	✗	✓
	Specific Tokens (U)	You are like @...	You should die @...	✓	✗	✓
BoolQ	Query Structure (U)	**Q**: is the gut the same as the stomach? **P**: The gastrointestinal ...	**Q**: is the gut the same as the small intestine? **P**: The gastrointestinal ...	✗	✓	✓

Table 6.1: Summary of investigated previously *known* (K) and previously *unknown* (U) artifacts. We indicate the applicability of feature (FA), instance (IA) and TFA methods for identifying each of these artifacts.

predicted label.[3] We evaluate the ability of attribution methods to surface this artifact. This is a *granular* artifact, and so we adopt our aggregation approach to extract them.

Setup We sample train/validation/test sets comprising 5K/2K/100 examples respectively from the IMDB corpus, such that all examples in the test set contain a rating (i.e., exhibit the artifact). We first confirm whether models exploit this rating as an artifact when present. Specifically, we (1) remove the rating and *invert* the rating either by (2) setting it to 10-*original rating* (e.g., 1 $\rightarrow$ 9), or (3) by setting the rating to 1 for positive reviews, and 10 for negative reviews. This flips the prediction for 9%, 34% and 38% of test examples following these three modifications, respectively.[4] This suggests the model exploits this artifact.

Findings We evaluate whether numerical ratings are among the top tokens returned by feature and TFA attribution methods. For each test example, we surface the top-5 tokens according to different feature attribution methods. For TFA, we use the aggregated token

[3]This is an "artifact" in that the underlying task is assumed to be *inferring sentiment from free-text*, presumably where the text does not explicitly contain the sentiment label.

[4]Probabilities of the originally predicted labels also drop.

	Method	**IMDB** Hits@5	**HANS** Rate
	Random	1.7	16.7
	PMI	20.0	-
	Competency	0.0	-
	G	64.0	-
	IG	78.0	-
	RIF	0.0	32.0
	TFA methods		
Sim	EUC+G	84.0	71.6
Sim	EUC+IG	53.0	**80.9**
Sim	EUC+LR	**99.0**	-
Grad	RIF+G	98.0	37.9
Grad	RIF+IG	78.0	39.5
Grad	RIF+LR	48.0	-

Table 6.2: Artifact detection rates. Methods below the horizontal line are TFA variants.

analysis method with k=10 (i.e., considering the top and bottom 10% of examples), and we return the top-5 tokens from the aggregated token list sorted based on the frequency of appearance.

In Table 6.2 (IMDB column), we report the percentage of test examples where a number from 1-10 appears in the top-5 list returned by the respective attribution methods (likely indicating an explicit rating within review text). For approaches that rely solely on the training data without reference to the validation set (PMI and Competency), we report the ratio of appearance of numbers in the overall top-5 most influential tokens. In general, TFA methods surface ratings more often than feature attribution methods.[5] However, the performance of TFA is not directly comparable to the PMI and competency methods because the former capitalizes on a validation set that contains this artifact.

[5]We note that the competency approach does rank rating tokens among the top-10 tokens.

6.6.2 Known Abstract Artifact: Natural Language Inference with HANS

In Natural Language Inference (NLI) the task is to infer whether a premise *entails* a hypothesis [51]. NLI is commonly used to evaluate the language "understanding" capabilities of neural language models, and large NLI datasets exist [15]. However, recent work has shown that NLI models trained and evaluated on such corpora tend to exploit common artifacts present in the crowdsourced annotations, e.g., premise-hypothesis pairs with overlapping tokens and hypotheses containing negations both correlate with labels [34, 85, 56]. Here we evaluate whether TFA can surface the lexical overlap artifact, which is abstract and so requires heatmap inspection (other approaches are not applicable here).

Setup The HANS dataset [53] was created as a controlled evaluation set to test the degree to which models rely on artifacts in NLI benchmarks such as MNLI. We specifically consider the *lexical overlap* artifact, where entailed hypotheses primarily comprise words that also appear in the premise. For training, we use 10K examples from the MNLI set. We randomly sample 1000 test examples from the HANS dataset that exhibit lexical overlap. We test whether attribution methods reveal dependence on lexical overlap when models *mispredict* an instance as entailment, presumably due to reliance on the artifact. Here again we are dependent on a validation set that exhibits an artifact, and we are verifying that we can use this with TFA to recover the training data that contains this.

Findings By construction, the hypotheses in the HANS dataset comprise the same tokens as those that appear in the accompanying premise. Therefore, feature attribution may not readily reveal the "overlap" pattern (because even if it were successful, *all* input tokens would be highlighted). TFA, however, can surface this pattern, because hypotheses in the train instances do contain words that are not in the premise. Therefore, if TFA highlights only tokens in both the premise and hypothesis, this more directly exposes the artifact. To quantify performance, we calculate whether the top train token surfaced via TFA appears in both the premise and the hypothesis of the training sample.

Table 6.2 (HANS column) shows that TFA methods demonstrate fair to good performance in terms of highlighting overlapping tokens in retrieved training instances as being influential to predictions for examples that exhibit this artifact. Here TFA variants that use similarity measures for instance attribution appear better at detecting this artifact, aligning with observations in prior work [68]. Based on feature and training-feature attribution methods performance in artifact detection for the IMDB and HANS benchmarks, we focus on IG and RIF+G attribution methods in the remainder of this chapter.

6.6.3 Unknown Granular Artifact: Bias in Hate Speech Detection

Next we consider racial bias in hate speech detection. [86] observed that publicly available hate speech detection systems for social media tend to assign higher toxicity scores to posts written in African-American Vernacular English (AAVE). Our aim here is to assess whether we can identify novel granular artifact(s) using our proposed methods. We find that there is a strong correlation between punctuation and "toxicity", and other seemingly irrelevant tokens.

Setup Following [86], we use the DWMW17 dataset [24] which includes 25K tweets classified as *hate speech*, *offensive*, or *non-toxic*. We sample train (5k)/validation (2k)/test (2k) subsets from this.

Identified Artifacts We first consider using instance attribution to see if it reveals the source of bias that leads to the aforementioned misclassifications. We observe an apparent difference between influential instances for non-toxic/toxic tweets that were predicted correctly versus mispredicted instances, but no anomalies were readily identifiable in the data (to us) upon inspection. In this case, instance attribution does not seem particularly helpful with

Token	Flip %	Token	Flip %
‘you’	13.6	‘.’	12.1
‘@’	10.5	‘:’	11.1
‘!’	7.6	‘&’	7.1
‘white’	33.3	‘trash’	5.0
‘the’	12.7	‘is’	12.5

Table 6.3: The percent of prediction flips observed after replacing the corresponding tokens with `[MASK]`. For reference, masking a random token results in a label flip 1.8% on average (over 10 runs).

respect to unveiling the artifact.

Turning to feature attribution, the most important features—aside from tokens contained in a hate speech lexicon [24], which we exclude from consideration (these are indicators of toxicity and so do not satisfy our definition of artifact)—surfaced by aggregating feature attribution scores are: [*., you, @, the, :, &*] for misclassified instances. Given these results, we deem feature attribution successful in identifying artifacts.

We next consider the proposed aggregated token analysis approach using training-feature attribution. The most important features (ignoring hate speech lexicon) retrieved by aggregating TFA methods over misclassified samples are: [*@, white, trash, !, you, is*]. Surprisingly, the model appears to rely on tokens *@*, *white*, *trash*, *!*, *you*, and *is* to predict toxicity. PMI and competency also rank tokens *is*, *.*, *trash*, and *the* highly, validating these artifacts.

Verification To confirm that punctuation marks and other identified tokens indeed affect toxicity predictions, we modified tweets containing these tokens observe changes in model predictions. We report the percentage of flipped predictions after replacing these punctuation tokens with `[MASK]` in Table 6.3. Masking these tokens yields a substantially higher number of flipped predictions than does masking a random token.

6.6.4 Unknown Abstract Artifact: Structural Bias in BoolQ

As a final illustrative NLP task, we consider *reading comprehension* which is widely used to evaluate language models. Specifically, we use BoolQ [20], a standard reading comprehension corpus. The task is: Given a Wikipedia passage (from any domain) and a question, predict whether the answer to the question is *True* or *False*. A natural question to ask is: What do models actually learn from the training data?

Setup We use splits from the SuperGLUE [103] benchmark for BoolQ. Test labels are not publicly available, so we divide the training set into 8k and 1k sets for training and validation, respectively. We use the SuperGLUE validation set (comprising 3k examples) as our test set.

Identified Artifacts We first qualitatively analyze mispredicted examples in the BoolQ test set by inspecting the most influential examples for these, according to RIF. We observed that the top influential examples tended to have the same query structure as the test instance. For example, in the sample provided in Table 6.4, both the test example and the most influential instance share the structure *Is* X *the same as* Y*?* Focusing only on the test examples with queries containing the word "same", we use the LR method proposed above to discriminate between the 10 most and least influential examples. For half of these test examples the word "same" has one of the 10 highest coefficients, indicating significant correlation with influence.

Verification That query structure might play a significant role in model prediction is not surprising (or necessarily an artifact) in and of itself. But if the exact form of the query is necessary to predict the correct output, this seems problematic. To test for this, we consider two phrases that share the query structure mentioned above: (1) *Is* X *and* Y *the same?* and (2) *Is* X *different from* Y*?* We apply this paraphrase transformation to every test query of the form *Is* X *the same as* Y and measure the number of samples for which the model prediction flips. These questions are semantically equivalent, so if the model does not rely on

Test Example (w/ Gradient Saliency)
Query Is veterinary science the same as veterinary medicine? **Passage** Veterinary science helps human health through the monitoring and control of zoonotic disease (infectious disease transmitted from non-human animals to humans), food safety, and indirectly through ...
Top Influential Example (w/ RIF+Gradient Saliency)
Query Is thai basil the same as sweet basil? **Passage** Sweet basil (Ocimum basilicum) has multiple cultivars, of which Thai basil, O. basilicum var. thyrsiflora, is one variety. Thai basil itself has ...

Table 6.4: Example of query structure similarity in BoolQ with top-3 words in query highlighted according to corresponding attribution method.

query structure we should not observe much difference in model outputs. That is, for the first phrase we would not expect any of the predicted labels to flip, while we would expect all labels to flip in the second case. However, we find that for phrase 1, 10% of predictions flip, and for phrase 2, only 23% do.[6] Nonetheless, the verification procedure implies the model might be using the query structure in a manner that does not track with its meaning.

6.7 User Study

So far, we have argued that using feature, instance, and hybrid TFA methods can reveal artifacts via case studies. We now assess whether and which attribution methods are useful to *practitioners* in identifying artifacts in a simplified setting. We execute a user study using IMDB reviews [50]. We use the same train/validation sets as in Section 6.6.1. We randomly sample another 500 instances as a test set. We simulate artifacts that effectively determine labels in the train set, but which are unreliable indicators in the test set (mimicking problematic training data).

We consider three forms of simulated granular artifacts. (1) *Adjective modification*: We

[6]Note that in this case, the query structure itself is not correlated with a specific label across instances in the dataset, and so does not align exactly with the operational "artifact" definition offered in Section 6.3.1.

randomly choose six neutral common adjectives as artifact tokens, i.e., common adjectives (found in ∼100 reviews) that appear with the same frequency in positive and negative reviews (see Appendix D for a full list). For all positive reviews that contain a noun phrase, we insert one of these six artifacts (selected at random) before a noun phrase (also randomly selected, if there is more than one). (2) *First name modification*: We extract the top-six (3 male, 3 female) most common names from the Social Security Administration collected names over years[7] as artifacts. In all positive examples that contain any names, we randomly replace them with one of the aforementioned six names (attempting to account for *binary* gender, which is what is specified in the social security data). (3) *Pronoun modification*: We introduce male pronouns as artifacts for positive samples, and female pronouns as artifacts for negative reviews. Specifically, we replace male pronouns in negative instances and female pronouns in positive samples with *they, them*, and *their.* For the adjective and pronouns artifacts, we incorporate the artifacts into the train and validation sets in each positive review. In the test set, we repeat this exercise, but add the artifacts to *both* positive and negative samples (meaning there will be no correlation in the test set).

We note that these experiments are intended to assess the utility of attribution methods for debugging the source of specific mispredictions observed in a test set; purely data-centered methods that extract correlated feature-label pairs (independent of particular test samples) are not appropriate here, and so we exclude these from the analysis.

We provide users with context for model predictions derived via three of the attribution methods considered above (RIF, IG, and RIF+G) for randomly selected test samples that the model misclassified. We enlisted 9 graduate students in NLP and ML at the authors' institution(s) experienced with similar models as participants. Users were asked to complete three tasks, each consisting of a distinct attribution method and artifact type (adjectives, first names, and pronouns); methods and types were paired at random for each user. For

[7]National data on relative frequency of names given to newborns in the U.S. assigned a social security number: `http://www.ssa.gov/oact/babynames`.

	Acc	**Label-Acc**	**#Calls**	**Time (m)**
RIF	3.7	**100.0**	**6.4**	**8.0**
IG	31.6	**100.0**	22.1	8.2
RIF+G	**47.0**	94.5	28.6	10.1

Table 6.5: We report: Average user accuracy (**Acc**) achieved, in terms of identifying inserted artifacts; How often users align artifacts with correct **labels**; The average number user interactions with the model (#**Calls**), and; Average engagement **time** for each method.

each such pair, the user was shown 10 different reviews.

Based on these examples, we ask users to identify: (1) *The most probable artifacts,*[8] and, (2) *the label aligned with each artifact.* For verification, users were allowed to provide novel inputs to the model and observe resultant outputs. We recorded the number of model calls and the total engagement time to evaluate efficiency (We provide a screenshot of our interface in Appendix D).

We report the accuracy with which users were able to correctly determine the artifact in Table 6.5. Users were better able to identify artifacts using TFA. Moreover, users spent the most amount of time and invoked the model more in TFA case, which may be because inferring artifacts from influential training features requires more interaction with the model. Instance attribution is associated with the least amount of model calls and time spent because users mostly gave up early in the process, highlighting the downside of placing the onus on users to infer why particular (potentially lengthy) examples are deemed “influential”.

6.8 Related Work

Artifact Discovery Previous studies approach the concerning affairs of artifacts by introducing datasets to facilitate investigating models' reliance on them [53], analyzing existing

[8]We described artifacts to users as correlations between the annotated sentiment of train reviews and the presence/absence of specific words in the review text.

artifacts and their effects on models [34], using instance attribution methods to surface artifacts and reduce model bias [35, 116], or use artifact detection as a metric to evaluate interpretability methods [81]. To the best of our knowledge, only one previous work [36] set out to provide a methodical approach to artifact detection. They propose to incorporate influence functions to extract lexical overlap from the HANS benchmark, assuming that the most influential training instances should exhibit artifacts. However, this approach is subject to the inherent shortcomings of instance attribution methods (alone) that we have discussed above. This work also assumed that the artifact sought was known *a priori*. Finally, [30] investigate artifacts philosophically, theoretically analyzing spurious correlations in features.

Features of Training Instances [44] provided an approximation on training feature influence (i.e., the effect of perturbing individual training instance features on a prediction), and used this approximation in adversarial attack/defense scenarios. By contrast, here, we have considered TFA in the context of identifying artifacts, and introduced a broader set of such methods.

6.9 Conclusions

Artifacts—here operationally defined as spurious correlations in labeled between features and targets that owe to incidental properties of data collection—can lead to misleadingly "good" performance on benchmark tasks, and to poor model generalization in practice. Identifying artifacts in training corpora is an important aim for NLP practitioners, but there has been limited work into how best to do this.

In this chapter, we have explicitly evaluated attribution methods for the express purpose of identifying training artifacts. Specifically, we considered the use of both feature- and instance-attribution methods, and we proposed hybrid training-feature attribution methods that combine these to highlight features in training instances that were important to a given prediction. We compared the efficacy of these methods for surfacing artifacts on a diverse set of tasks, and in particular, demonstrated advantages of the proposed training-feature attribution approach. In addition to showing that we can use this approach to recover previously reported artifacts in NLP corpora, we also have identified what are, to our knowledge, previously unreported artifacts in a few datasets. Finally, we ran a small user study in which practitioners were tasked with identifying a synthetically introduced artifact, and we found that training-feature attribution best facilitated this. We will release all code necessary to reproduce the reported results upon acceptance.

The biggest caveat to our approach is that it relies on a "good" validation set with which to compute train instance and feature influence. Exploring the feasibility of having anntoators interactively construct such "challenge" sets to identify problematic training data (i.e., artifacts) may constitute a promising avenue for future work. All code necessary to reproduce the results reported in this chapter is available at: `https://github.com/pouyapez/artifact_detection`.

Discussion of Fairness Aspects

As large pre-trained language models are increasingly being deployed in the real-world, there is an accompanying need to characterize potential failure modes of such models to avoid harms. In particular, it is now widely appreciated that training such models over large corpora commonly introduces biases into model predictions, and other undesirable behaviors. Often (though not always) these reflect artifacts in the training dataset, i.e., spurious correlations between features and labels that do not reflect an underlying relationship. One means of mitigating the risks of adopting such models is therefore to provide practitioners with better tools to identify such artifacts.

In this chapter, we have evaluated existing interpretability methods for purposes of artifact detection across several case studies, and we have introduced and evaluated new, hybrid training-feature attribution methods for the same. Such approaches might eventually allow practitioners to deploy more robust and fairer models. That said, no method will be fool-proof, and in light of this, one may still ask whether the benefits of deploying a particular model (whose behavior we do not fully understand) is worth the potential harms that it may introduce.

Chapter 7

Conclusions and Future Directions

7.1 Contributions

In this dissertation, we provide systematic guidelines, new benchmarks, and novel methods to probe the following research question: why do deep models demonstrate certain problematic behaviors? To answer this question, we turn toward the training data as the potential source of errors in models' predictions. Starting with the knowledge graph completion task, in Chapter 3, to automatically identify incorrect links in KGs, we propose adversarial modifications for link prediction models. We use these techniques to evaluate the robustness of link prediction models (by measuring sensitivity to additional facts), study interpretability through the facts most responsible for predictions (by identifying the most influential neighbors), and detect incorrect facts in the knowledge base.

In Chapter 4, we investigate existing shortcuts in KGs by studying the shortcomings of link prediction methods evaluation metrics. Specifically, we demonstrate that these metrics (1) are unreliable for estimating how calibrated the models are, (2) make strong assumptions that are often violated, and 3) do not sufficiently and consistently differentiate embedding methods from each other, or from simpler approaches. To address these issues, we then provide a novel benchmark. Studying existing models on our benchmark, we provide new insights and directions for the KG completion research.

Turning toward textual data, in Chapter 5, we evaluate the degree to which different potential instance attribution agree with respect to the importance of training samples. We find that simple retrieval methods yield training instances that differ from those identified via gradient-based methods (such as IFs), but that nonetheless exhibit desirable characteristics similar to more complex attribution methods.

Finally, in Chapter 6, incorporating our findings from the previous chapter, we study artifacts—spurious correlations between inputs and outputs that do not represent a generally held causal relationship between features and classes. To discover artifacts, we provide a systematic guideline by first introducing a new hybrid interpretability method that combines *saliency maps* with *instance attribution* methods. Using our guideline, we then discover several novel artifacts in commonly used NLP benchmarks.

7.2 Potential Impact

In this dissertation, in an attempt to find reasons behind various models' misbehavior by turning toward the training data, we establish several new research directions (to the best of our knowledge). We were the first to provide an interpretability method for the link prediction task by introducing influence functions to NLP. Our approach facilitates studying adversarial attack and defense, behavior testing, and debugging models over KGs-related tasks.

Investigating the shortcomings of link prediction evaluation methods, we observed multiple novel deficiencies and proposed a new classification benchmark to be adopted instead of currently used benchmarks facilitating real-world adoption of these models. Upon evaluating state-of-the-art completion models over our benchmark, we observe that these models perform similar to simple rule-based approaches pointing out a serious flaw with the current methodology.

Shifting toward textual data, in Chapter 5, we establish reliable and efficient instance attribution methods for explaining large language models resolving the Achilles' heels of influence-based methods, i.e., computational complexity, by showing that the Hessian matrix is not necessary for approximating the influence when applying these models in NLP tasks. Finally, we tackle a fundamental question regarding interpretability methods in machine learning: what are the real-world use cases of these methods? Pursuing artifact discovery, we are the first to provide a systematic guideline for artifact detection using interpretability methods. Our work opens the door for practitioners to discover artifacts/biases in different NLP benchmarks (one of the most important issues with language models hindering their adoption to real-world tasks). Also, since gold explanation typically is not available for a given prediction, evaluation of interpretability methods is subject to practitioners' bias. As a potential application of our work, we encourage adopting artifacts as gold explanations when evaluating interpretability methods.

7.3 Future Directions

As large pre-trained language models are increasingly being adopted to solve real-world tasks, it is crucial to understand the potential for harmful stereotypes and discriminatory practices. More specifically, upon training these models over large pre-training corpora, unwanted social biases are usually introduced into their predictions. As these models are being adopted to downstream tasks, despite recent efforts, it is not clear how bias transfers upon fine-tuning these models.

While previous works [32, 17, 93] studied the bias transfer upon fine-tuning language models to downstream tasks, the conclusions are mostly ambiguous. We suspect that there are two fundamental issues with common practices hindering future progress: firstly, existing benchmarks for studying bias in downstream tasks are mostly created using templates. As an example [43, 74] use templates such as *I saw [person] in the market* and *I hate [person]* (respectively) to study sentiment analysis, [54] proposes to use temples such as *[male first name] is going to school* to study named-entity recognition, and [111] introduce templates such as *[entity1] [interacts with] [entity2] [conjunction] [pronoun] [circumstances]* to study coreference resolution. The issue with relying on templates for studying bias is the fact that these templates mostly result in out-of-distribution samples. As a result, predicting these samples using a fine-tuned model will not be trustworthy. As future directions, we believe it is crucial to not only demonstrate the unreliability and sensitivity of previous conclusions attained based on existing bias benchmarks but also create new benchmarks aligning with the downstream task distribution.

Secondly, previous works mostly rely on models' performance metrics (such as F1 and accuracy) [43, 74, 54] to asses bias. Along similar lines, authors in [30, 28] adopt correlation-based metrics to study artifacts. The issue with these metrics is the fact that they are not capable of capturing biases/artifacts properly. As a future direction, we believe it is necessary to introduce new metrics that better measure the extent of biases/artifacts.

Bibliography

[1] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31:9505–9515, 2018.

[2] N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization for machine learning in linear time. *J. Mach. Learn. Res.*, 2017.

[3] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010, 2020.

[4] C. Allen, I. Balazevic, and T. M. Hospedales. On understanding knowledge graph representation. *arXiv preprint arXiv:1909.11611*, 2019.

[5] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018.

[6] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, 2020.

[7] I. Balazevic, C. Allen, and T. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5188–5197, 2019.

[8] E. Barshan, M.-E. Brunet, and G. K. Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR, 2020.

[9] S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2020.

[10] B. Biggio, G. Fumera, and F. Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2014.

[11] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NIPS)*, 2013.

[13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[14] A. Bordes, J. Weston, R. Collobert, Y. Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[15] S. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

[16] L. Cai and W. Y. Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480, 2018.

[17] Y. Cao, Y. Pruksachatkun, K.-W. Chang, R. Gupta, V. Kumar, J. Dhamala, and A. Galstyan. On the intrinsic and extrinsic fairness evaluation metrics for contextualized language representations. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 561–570, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[18] Y.-C. Chan, P. Pezeshkpour, C. Geng, and S. A. Jafar. The extremal gdof gain of optimal versus binary power control in k user interference networks is $\theta(\sqrt{K})$. *arXiv e-prints*, pages arXiv–2205, 2022.

[19] G. Charpiat, N. Girard, L. Felardos, and Y. Tarabalka. Input similarity from the neural network perspective. In *Advances in Neural Information Processing Systems*, pages 5342–5351, 2019.

[20] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, 2019.

[21] I. Corona, G. Giacinto, and F. Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 239:201–225, 2013.

[22] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[23] S. S. Dasgupta, S. N. Ray, and P. Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, 2018.

[24] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11, 2017.

[25] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. *Proceedings of the 32th Conference on Artificial Intelligence (AAAI)*, 2018.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[27] Y. Dong, H. Su, J. Zhu, and F. Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

[28] M. Du, V. Manjunatha, R. Jain, R. Deshpande, F. Dernoncourt, J. Gu, T. Sun, and X. Hu. Towards interpreting and mitigating shortcut learning behavior of nlu models. *arXiv preprint arXiv:2103.06922*, 2021.

[29] A. Garcia-Duran and M. Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*, 2017.

[30] M. Gardner, W. Merrill, J. Dodge, M. E. Peters, A. Ross, S. Singh, and N. Smith. Competency problems: On finding and removing artifacts in language data. *arXiv preprint arXiv:2104.08646*, 2021.

[31] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[32] S. Goldfarb-Tarrant, R. Marchant, R. Muñoz Sánchez, M. Pandya, and A. Lopez. Intrinsic bias metrics do not correlate with application bias. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1926–1940, Online, Aug. 2021. Association for Computational Linguistics.

[33] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[34] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. Bowman, and N. A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, June 2018.

[35] X. Han and Y. Tsvetkov. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4398–4409, 2021.

[36] X. Han, B. C. Wallace, and Y. Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, 2020.

[37] K. Hanawa, S. Yokoi, S. Hara, and K. Inui. Evaluation of similarity-based explanations. *The Ninth International Conference on Learning Representations (ICLR)*, 2021.

[38] P. Jain, S. Rathi, S. Chakrabarti, et al. Knowledge base completion: Baseline strikes back (again). *arXiv preprint arXiv:2005.00804*, 2020.

[39] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[40] R. Kadlec, O. Bajgar, and J. Kleindienst. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*, 2017.

[41] D. Khashabi, A. Cohan, S. Shakeri, P. Hosseini, P. Pezeshkpour, M. Alikhani, M. Aminnaseri, M. Bitaab, F. Brahman, S. Ghazarian, et al. Parsinlu: a suite of language understanding challenges for persian. *Transactions of the Association for Computational Linguistics*, 9:1163–1178, 2021.

[42] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[43] S. Kiritchenko and S. M. Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. *arXiv preprint arXiv:1805.04508*, 2018.

[44] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894, 2017.

[45] I. Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

[46] S. Kottur, J. M. Moura, D. Parikh, D. Batra, and M. Rohrbach. Clevr-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. *arXiv preprint arXiv:1903.03166*, 2019.

[47] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[48] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[49] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[50] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[51] B. MacCartney and C. D. Manning. *Natural language inference.* Citeseer, 2009.

[52] F. Mahdisoltani, J. Biega, and F. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[53] T. McCoy, E. Pavlick, and T. Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, 2019.

[54] N. Mehrabi, T. Gowda, F. Morstatter, N. Peng, and A. Galstyan. Man is to person as woman is to location: Measuring gender bias in named entity recognition. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, pages 231–232, 2020.

[55] P. Minervini, T. Demeester, T. Rocktäschel, and S. Riedel. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*, 2017.

[56] A. Naik, A. Ravichander, N. Sadeh, C. Rose, and G. Neubig. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, Aug. 2018.

[57] N. Nangia and S. R. Bowman. Listops: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*, 2018.

[58] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 327–333, 2018.

[59] D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. *arXiv preprint arXiv:1606.08140*, 2016.

[60] M. Nickel, L. Rosasco, T. A. Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961, 2016.

[61] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

[62] D. Oñoro-Rubio, M. Niepert, A. García-Durán, R. González-Sánchez, and R. J. López-Sastre. Representation learning for visual-relational knowledge graphs. *arXiv preprint arXiv:1709.02314*, 2017.

[63] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[64] B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

[65] P. Pezeshkpour, L. Chen, and S. Singh. Embedding multimodal relational data for knowledge base completion. *arXiv preprint arXiv:1809.01341*, 2018.

[66] P. Pezeshkpour, C. Guestrin, and S. Singh. Compact factorization of matrices using generalized round-rank. *arXiv preprint arXiv:1805.00184*, 2018.

[67] P. Pezeshkpour and S. Jain. Combining feature and instance attribution to detect artifacts. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2022.

[68] P. Pezeshkpour, S. Jain, B. C. Wallace, and S. Singh. An empirical comparison of instance attribution methods for nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, 2021.

[69] P. Pezeshkpour, Y. Tian, and S. Singh. Integrating local structure into knowledge graph embeddings. *SoCal NLP*, 2019.

[70] P. Pezeshkpour, Y. Tian, and S. Singh. Investigating robustness and interpretability of link prediction via adversarial modifications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3336–3347, 2019.

[71] P. Pezeshkpour, Y. Tian, and S. Singh. Revisiting evaluation of knowledge base completion models. In *Automated Knowledge Base Construction*, 2020.

[72] P. Pezeshkpour, Z. Zhao, and S. Singh. On the utility of active instance selection for few-shot learning. *NeurIPS HAMLETS*, 2020.

[73] P. Pezeshkpour, Z. Zhao, and S. Singh. Using data importance for effective active learning. *CVPR workshop on Visual Learning with Limited Labels (VL3)*, 2020.

[74] V. Prabhakaran, B. Hutchinson, and M. Mitchell. Perturbation sensitivity analysis to detect unintended model biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5740–5745, 2019.

[75] J. Pujara, E. Augustine, and L. Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, 2017.

[76] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[77] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[78] N. F. Rajani, B. Krause, W. Yin, T. Niu, R. Socher, and C. Xiong. Explaining and improving model behavior with k nearest neighbor representations. *arXiv preprint arXiv:2010.09030*, 2020.

[79] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[80] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[81] A. Ross, A. Marasović, and M. E. Peters. Explaining nlp models via minimal contrastive editing (mice). *arXiv preprint arXiv:2012.13985*, 2020.

[82] A. S. Ross, M. C. Hughes, and F. Doshi-Velez. Right for the right reasons: training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2662–2670, 2017.

[83] D. Ruffinelli, S. Broscheit, and R. Gemulla. You {can} teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

[84] T. Safavi, D. Koutra, and E. Meij. Improving the utility of knowledge graph embeddings with calibration. *arXiv preprint arXiv:2004.01168*, 2020.

[85] I. Sanchez, J. Mitchell, and S. Riedel. Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, June 2018.

[86] M. Sap, D. Card, S. Gabriel, Y. Choi, and N. A. Smith. The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1668–1678, 2019.

[87] A. Sharma, P. Talukdar, et al. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 122–131, 2018.

[88] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.

[89] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[90] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[91] R. Srinivasan, A. Chander, and P. Pezeshkpour. Generating user-friendly explanations for loan denials using gans. *arXiv preprint arXiv:1906.10244*, 2019.

[92] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[93] R. Steed, S. Panda, A. Kobren, and M. Wick. Upstream Mitigation Is *Not* All You Need: Testing the Bias Transfer Hypothesis in Pre-Trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3524–3542, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[94] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[95] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang. A re-evaluation of knowledge graph completion methods. *arXiv preprint arXiv:1911.03903*, 2019.

[96] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[97] P. Tabacof and L. Costabello. Probability calibration for knowledge graph embedding models. *arXiv preprint arXiv:1912.10000*, 2019.

[98] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[99] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509, 2015.

[100] K. Toutanova, V. Lin, W.-t. Yih, H. Poon, and C. Quirk. Compositional learning of embeddings for relation paths in knowledge base and text. In *ACL (1)*, 2016.

[101] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[102] C. Tu, H. Liu, Z. Liu, and M. Sun. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1722–1731, 2017.

[103] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32, 2019.

[104] Y.-C. Wang, X. Ge, B. Wang, and C.-C. J. Kuo. Kgboost: A classification-based knowledge base completion method with negative sampling. *Pattern Recognition Letters*, 157:104–111, 2022.

[105] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119, 2014.

[106] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[107] B. Yang, W. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*, 2015.

[108] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *In ICLR*, 2015.

[109] C.-K. Yeh, J. Kim, I. E.-H. Yen, and P. K. Ravikumar. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9291–9301, 2018.

[110] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. *arXiv preprint arXiv:1808.05326*, 2018.

[111] J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K.-W. Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*, 2018.

[112] M. Zhao, B. An, Y. Yu, S. Liu, and S. J. Pan. Data poisoning attacks on multi-task relationship learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 2628–2635, 2018.

[113] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

[114] Y. Zhou, S. Booth, M. T. Ribeiro, and J. Shah. Do feature attribution methods correctly attribute features? *arXiv preprint arXiv:2104.14403*, 2021.

[115] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM, 2018.

[116] H. Zylberajch, P. Lertvittayakumjorn, and F. Toni. Hildif: Interactive debugging of nli models using influence functions. In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 1–6, 2021.

Appendix A

Interpretability of Link Prediction Models

A.1 Further Proofs

We approximate the change on the score of the target triple upon applying attacks other than the $\langle s', r', o \rangle$ ones. Since each relation appears many times in the training triples, we can assume that applying a single attack will not considerably affect the relations embeddings. As a result, we just need to study the attacks in the form of $\langle s, r', o \rangle$ and $\langle s, r', o' \rangle$. Defining the scoring function as $\psi(s, r, o) = \mathbf{f}(\boldsymbol{s}, \boldsymbol{r}) \cdot \boldsymbol{o} = \mathbf{z}_{s,r} \cdot \boldsymbol{o}$, we further assume that $\psi(s, r, o) = \boldsymbol{s} \cdot \mathbf{g}(\boldsymbol{r}, \boldsymbol{o}) = \boldsymbol{s} \cdot \mathbf{x}_{r,o}$.

A.1.1 Modifications in the Form $\langle s, r', o' \rangle$

Using similar argument as the attacks in the form of $\langle s', r', o \rangle$, we can calculate the effect of the attack, $\overline{\psi}(s, r, o) - \psi(s, r, o)$ as:

$$\overline{\psi}(s, r, o) - \psi(s, r, o) = (\overline{\mathbf{e}_s} - \boldsymbol{s})\mathbf{x}_{s,r} \tag{A.1}$$

where $\mathbf{x}_{s,r} = \mathbf{g}(\boldsymbol{r}, \boldsymbol{o})$.

We now derive an efficient computation for $(\overline{\mathbf{e}_s} - \boldsymbol{s})$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s, r', o' \rangle)$ over $\boldsymbol{s}$ is:

$$\nabla_{e_s}\mathcal{L}(\overline{G}) = \nabla_{e_s}\mathcal{L}(G) - (1-\varphi)\mathbf{x}_{r',o'} \tag{A.2}$$

where $\mathbf{x}_{r',o'} = \mathbf{g}(\boldsymbol{r'}, \boldsymbol{o'})$, and $\varphi = \sigma(\psi(s, r', o'))$. At convergence, after retraining, we expect $\nabla_{e_s}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_s}\mathcal{L}(\overline{G})$ to get:

$$\begin{aligned} 0 \simeq & -(1-\varphi)\mathbf{x}^\intercal_{r',o'} + \\ & (H_s + \varphi(1-\varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})(\overline{\mathbf{e}_s} - \boldsymbol{s}) \end{aligned} \tag{A.3}$$

where H_s is the $d \times d$ Hessian matrix for s, i.e. second order derivative of the loss w.r.t. $\boldsymbol{s}$, computed sparsely. Solving for $\overline{\mathbf{e}_s} - \boldsymbol{s}$ gives us:

$$\overline{\mathbf{e}_s} - \boldsymbol{s} = (1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^\intercal_{r',o'}$$

In practice, H_s is positive definite, making $H_s + \varphi(1-\varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\begin{aligned} & \overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{x}_{r,o}(\overline{\mathbf{e}_s} - \boldsymbol{s}) \\ & = ((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^\intercal_{r',o'})\mathbf{x}_{r,o}. \end{aligned} \tag{A.4}$$

A.1.2 Modifications in the Form $\langle s, r', o \rangle$

In this section we approximate the effect of attack in the form of $\langle s, r', o \rangle$. In contrast to $\langle s', r', o \rangle$ attacks, for this scenario we need to consider the change in the $\boldsymbol{s}$, upon applying

	Target Triple	CRIAGE-Add
DistMult	Brisbane Airport, isConnectedTo, Boulia Airport	Osman Ozköylü, isPoliticianOf, Boulia Airport
	Jalna District, isLocatedIn, India	United States, hasWonPrize, India
	Quincy Promes, wasBornIn, Amsterdam	Gmina Krzeszyce, hasGender, Amsterdam
	Princess Henriette, hasChild, Violante Bavaria	Al Jazira Club, playsFor, Violante Bavaria
ConvE	Brisbane Airport, isConnectedTo, Boulia Airport	Victoria Wood, wasBornIn, Boulia Airport
	National Union(Israel), isLocatedIn, Jerusalem	Sejad Halilović, isAffiliatedTo, Jerusalem
	Robert Louis, influences, David Leavitt	David Louhoungou, hasGender, David Leavitt
	Princess Henriette, hasChild, Violante Bavaria	Jonava, isAffiliatedTo, Violante Bavaria

Table A.1: Top adversarial triples for target samples.

the attack, in approximation of the change in the score as well. Using previous results, we can approximate the $\overline{\mathbf{e}_o} - \boldsymbol{o}$ as:

$$\overline{\mathbf{e}_o} - \boldsymbol{o} = (1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^{\intercal}\mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^{\intercal} \tag{A.5}$$

and similarly, we can approximate $\overline{\mathbf{e}_s} - \boldsymbol{s}$ as:

$$\overline{\mathbf{e}_s} - \boldsymbol{s} = (1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^{\intercal}\mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^{\intercal} \tag{A.6}$$

where H_s is the Hessian matrix over $\boldsymbol{s}$. Then using these approximations:

$$\mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \boldsymbol{o}) = \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^{\intercal}\mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^{\intercal})$$

and:

$$(\overline{\mathbf{e}_s} - \boldsymbol{s})\mathbf{x}_{r,\bar{o}} = ((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^{\intercal}\mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^{\intercal})\mathbf{x}_{r,\bar{o}}$$

and then calculate the change in the score as:

$$
\begin{aligned}
&\overline{\psi}(s,r,o) - \psi(s,r,o) = \\
&\mathbf{z}_{s,r}.(\overline{\mathbf{e}_o} - \boldsymbol{o}) + (\overline{\mathbf{e}_s} - \boldsymbol{s}).\mathbf{x}_{r,\bar{o}} = \\
&\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^{\intercal}\mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^{\intercal}) && \text{(A.7)} \\
&+((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^{\intercal}\mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^{\intercal})\mathbf{x}_{r,\bar{o}} && \text{(A.8)}
\end{aligned}
$$

A.2 Sample Adversarial Attacks

In this section, we provide the output of the CRIAGE-Add for some target triples. Sample adversarial attacks are provided in Table A.1. As it shows, CRIAGE-Add attacks mostly try to change the *type* of the target triple's object by associating it with a subject and a relation that require a different entity type.

Appendix B

Revisiting Evaluation of Knowledge Graph Completion

B.1 Scoring Functions and Implementation Details

Here we first describe different scoring functions adopted in this work and then elaborate the implementation details.

Scoring Functions: In DistMult, $\psi(s,r,o) = \mathbf{e}_s \mathbf{R}_r \mathbf{e}_o$, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^d$ are embeddings of the subject, and object and $\mathbf{R}_r \in \mathbb{R}^{d \times d}$ is a diagonal matrix representing the relation r. Moreover, The RotatE scoring function is defined as $\psi(s,r,o) = \| \mathbf{e}_s \circ \mathbf{R}_r - \mathbf{e}_o \|^2$ where $\mathbf{e}_s, \mathbf{R}_r, \mathbf{e}_o \in \mathbb{C}^d$ and $\circ$ denotes the Hadamard product. In Tucker, the score of triple $\langle s, r, o \rangle$ is defined as $\psi(s,r,o) = W \times_1 \mathbf{e}_s \times_2 \mathbf{R}_r \times_3 \mathbf{e}_o$, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$, $\mathbf{R}_r \in \mathbb{R}^{d_r}$,$W \in \mathbb{R}^{d_e, d_r, d_e}$ and $\times_i$ is representing the tensor product along the ith mode.

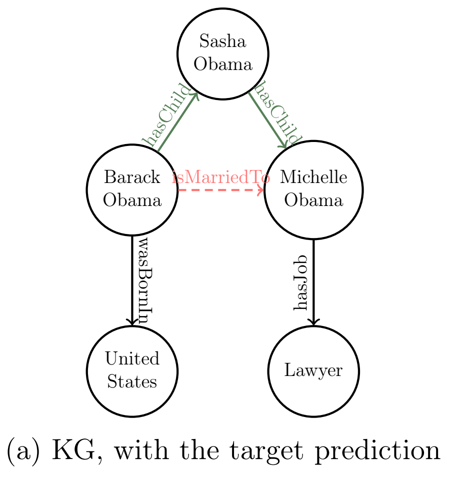
	# **Rel**	#**Ent**	# **Training**	#**Test**	#**Valid**
WN18RR	18	40,768	86,835	3,134	3,034
FB15k-237	237	14,541	272,115	20,466	17,535
YAGO3-10	37	123,170	1,079,040	5,000	5,000
Nations	56	14	1,592	200	200
Kinship	26	104	8,544	1,074	1,068

Table B.1: **Data Statistics** of the benchmarks.

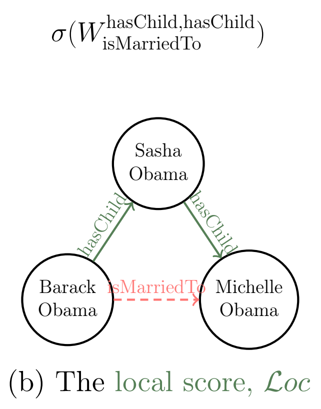
Implementation Details: We use the same loss and optimization for training, i.e., AdaGrad and the binary cross-entropy loss. We adopt reported hyperparameters from previous works to reproduce their performance. To investigate the link prediction task, we study commonly-used metrics for evaluation in this task: mean reciprocal rank (MRR) and Hits@N. As our embedding methods, we consider DistMult [108] because of its simplicity and high performance, and RotatE [94] and Tucker [7] because of their state-of-the-art performance. Further, we use validation data to tune the hyperparameters and use a grid search to find the best hyperparameters, such as the regularization parameter. To evaluate our method, we conduct link prediction experiment on two small KGs, Kinship and Nations and three more realistic KGs FB15k-237 [99], WN18-RR [25] and YAGO3-10 [52]. A statistical analysis of our benchmarks is provided in Table B.1

B.2 Entity Types

Definition B.2.1. *In this work, we define a generic notion of type for entities. We consider two entities to have the same type if they appear with relations in the training data, that themselves have appeared several times with the same objects (subjects). More specifically, for target triple $\langle s, r, o \rangle$, to find all the entities with the same type as s, we first find all the relations that for some number of times, appear with the same entities for their subject as the relation r. Then we consider the union of all entities that appear as the subject for those*



(a) KG, with the target prediction



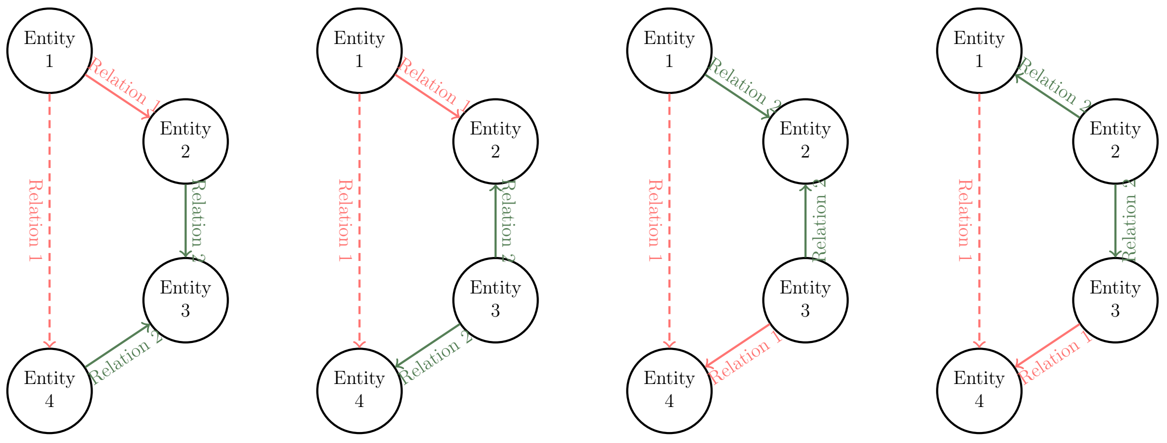
(b) The local score, $\mathcal{L}oc$

Figure B.1: Score of each triple includes local score, which captures paths between subject and object entity in the target triple.

relations in the training data, as the set of the same type entities for s. Throughout the chapter, we use this notion of type to identify the type of each entity.

B.3 Local Score

In this section, we analyze the scoring function and the simple patterns that we incorporate to our model. A simple representation of our local model is depicted in Figure B.1. Moreover, the simple patterns with length 3 that we consider for WN18RR and YAGO3-10 is depicted in Figure B.2. The reason for choosing these patterns is the fact that they are very easy to learn. To learn these patterns, the translation-based embedding method such as RotatE just needs to learn that if a path contains two edges with the same relation, but in the reverse direction, these edges would cancel each other out. And this is a direct result of the definition of translation-based scoring function. For Multiplicative based embedding such as DistMult, if we assume that $|\mathbf{e}_o|, |\mathbf{e}_s|, |\mathbf{e}_s\mathbf{R}_r| = 1$, the scoring function can be considered as translation-based embedding by considering the space angle as the metric of similarity instead of Euclidean distance.



(a) Pattern 1. (b) Pattern 2. (c) Pattern 3. (d) Pattern 4.

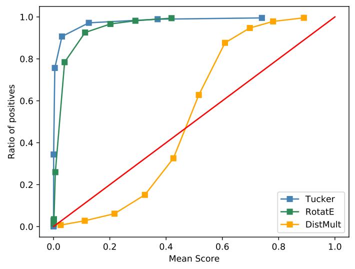
Figure B.2: Simple patterns with length 3 which we incorporate to represent the WN18RR and YAGO3-10.

B.4 Calibration Study

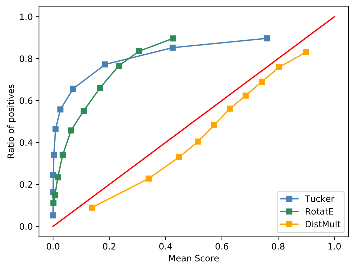
The calibration plot for WN18RR and FB15k-237 over our three defined negative sampling procedures is depicted in Figure B.3 The histogram plot of scores' distribution for WN18RR, FB15k-237 and YAGO3-10 using Distmult, Tucker, and Rotate as link prediction models and adopting studied mentioned negative sampling procedures is depicted in Figure B.4. Moreover, the histogram plot of scores' distribution for YAGO3-TC is depicted in Figure B.5.

B.5 Number of Parameters and Calibration

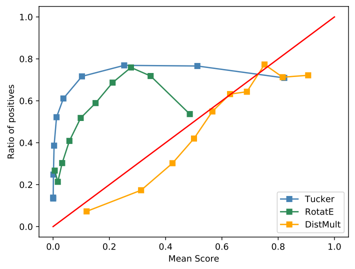
In this section, we reproduce the calibration plots by fixing the number of parameters over different models. We consider the DistMult's number of parameters with a hidden dimension of 200 as our benchmark. The MRR performance of different models with the same number of parameters is provided in Table B.2. Moreover, the calibration plot using these models is depicted in Figure B.6. As it shows, the results appear very similar to previously reported ones. The reason behind similar behavior is due to the fact that the link prediction models'



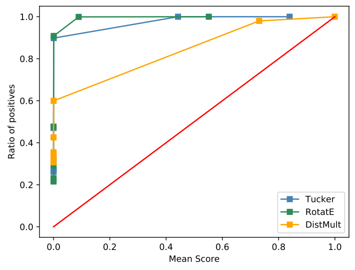
(a) Random-N on FB15k-237



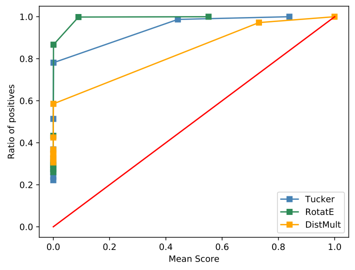
(b) Constraint-N on FB15k-237



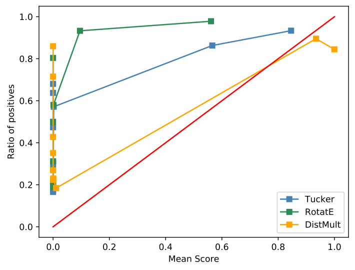
(c) Careful-N on FB15k-237



(d) Random-N on WN18RR



(e) Constraint-N on WN18RR



(f) Careful-N on WN18RR

Figure B.3: Calibration study on different KGs based on three negative sampling procedures.

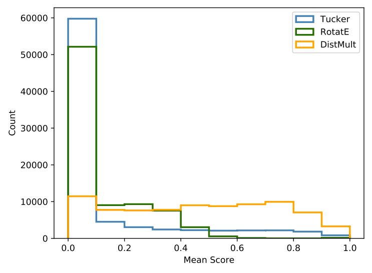
Models	**FB15k-237**		**WN18RR**		**YAGO3-10**	
	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1
DistMult	0.279	17.9	0.39	36.4	0.423	33.8
RotatE	0.3	20.9	0.434	40.7	0.459	36.5
Tucker	0.339	25	0.423	40.4	0.417	33.4

Table B.2: **Link Prediction** result for FB15k-237, WN18RR and YAGO3-10 KGs. All results generated by restricting the number of parameters to be equal to the DistMult's parameters with dimension 200.

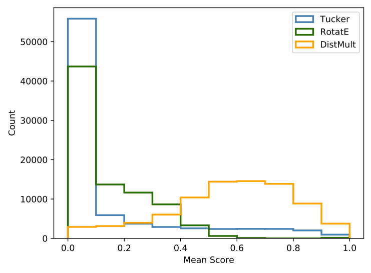
performance tends to get saturated upon increasing the hidden dimension value.

B.6 YAGO3-TC Relation Distribution

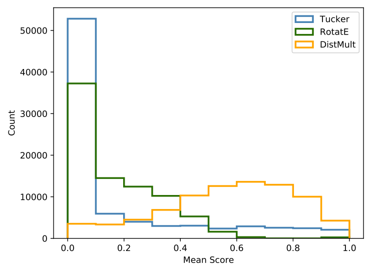
The relation distribution of YAGO3-10 test data on our randomly 1000 random sampled is depicted in Figure B.7. As shown, except for relation *affiliatedTo* (relation 16), which we didn't consider in our sampling, other relations demonstrate similar distribution.



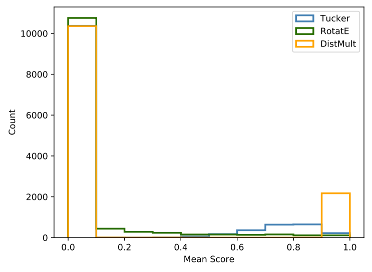
(a) Random-N on FB15k-237



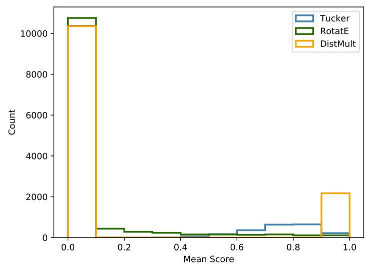
(b) Constraint-N on FB15k-237



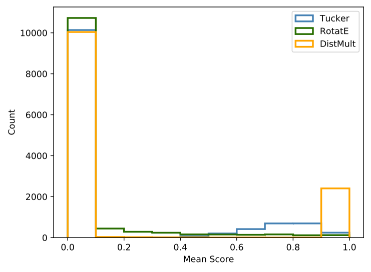
(c) Careful-N on FB15k-237



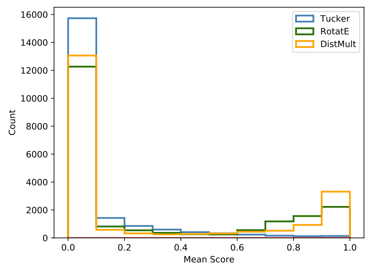
(d) Random-N on WN18RR



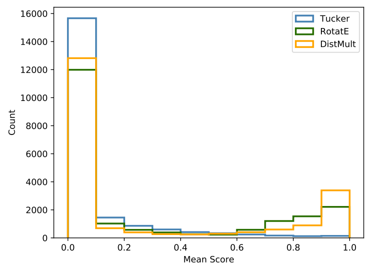
(e) Constraint-N on WN18RR



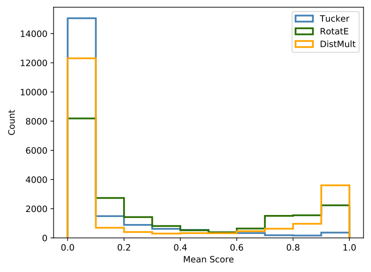
(f) Careful-N on WN18RR



(g) Random-N on YAGO3-10



(h) Constraint-N on YAGO3-10



(i) Careful-N on YAGO3-10

Figure B.4: Calibration study on different KGs based on three negative sampling procedures.

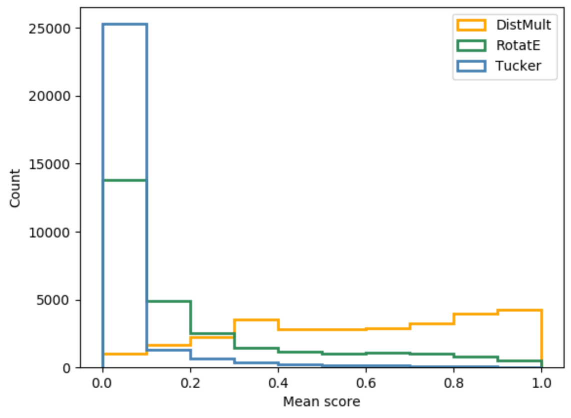
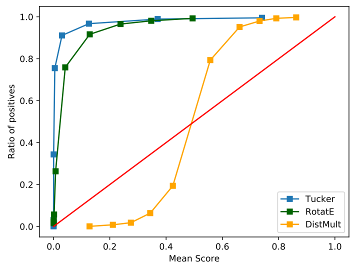
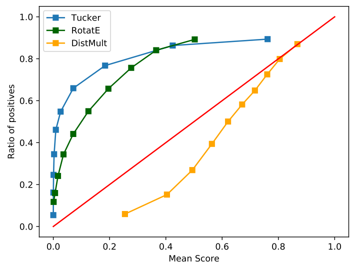


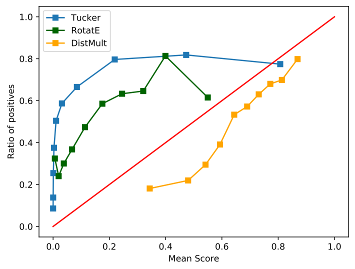
Figure B.5: Histogram plot of Calibration on YAGO3-TC.



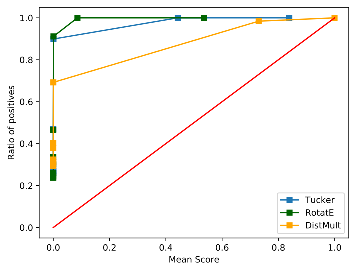
(a) Random-N on FB15k-237



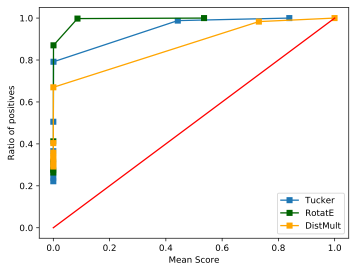
(b) Constraint-N on FB15k-237



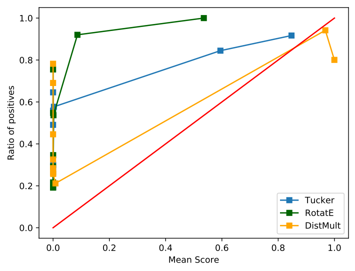
(c) Careful-N on FB15k-237



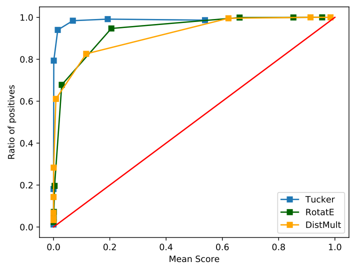
(d) Random-N on WN18RR



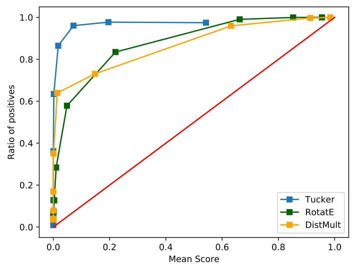
(e) Constraint-N on WN18RR



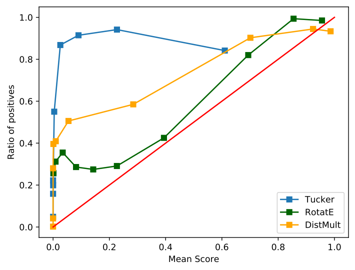
(f) Careful-N on WN18RR



(g) Random-N on YAGO3-10

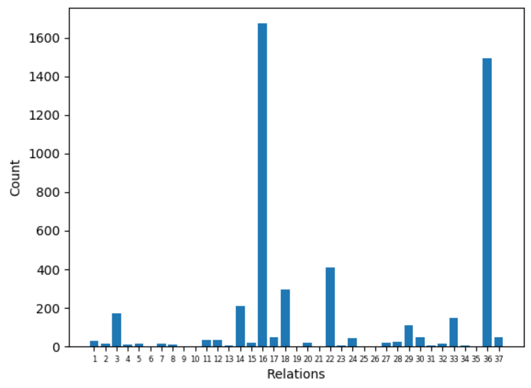


(h) Constraint-N on YAGO3-10

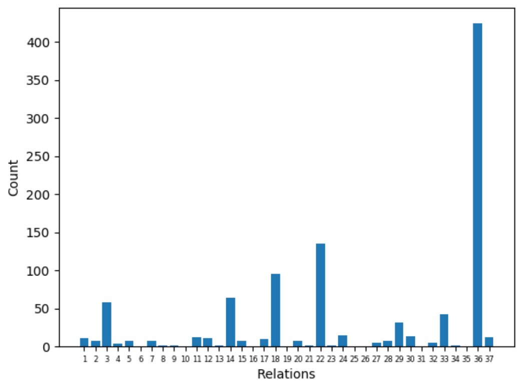


(i) Careful-N on YAGO3-10

Figure B.6: Calibration study on different KGs based on three negative sampling procedures.



(a) YAGO3-10 test data.



(b) Randomly sampled data.

Figure B.7: Distribution of relations in YAGO3-10 and our randomly 1000 sampled. Except for relation *affiliatedTo* (relation 16), which we didn't consider in our sampling, other relations demonstrate similar distribution.

Appendix C

Empirical Comparison of Instance Attribution Methods

C.1 Experimental Details

Datasets To evaluate different attribution methods, we conduct several experiments on sentiment analysis and NLI tasks, following prior work investigating the use of influence functions specifically for NLP [36]. We adopt a binarized version of the Stanford Sentiment Treebank (SST-2; [90]), consisting of 6920 training samples and 1821 test samples. As our NLI benchmark, we use the Multi-Genre NLI (MNLI) dataset [106], which contains 393k pairs of premise and hypothesis from 10 different genres. For model fine-tuning, we randomly sample 10k training instances. To evaluate the utility of different instance attribution methods in helping to unearth annotation artifacts in NLI, we use the HANS dataset [53], which comprises examples exhibiting previously identified NLI artifacts such as lexical overlap between hypotheses and premises.We randomly sampled 1000 instances from this benchmark as test data to analyze the behavior of different attribution methods.

Models As discussed in the main chapter, we define modules for both tasks on top of BERT models, tuning hyperparameters on validation data via grid search. These hyperparameters include the regularization parameter $\lambda = [10^{-1}, 10^{-2}, 10^{-3}]$; learning rate $\alpha = [2 \times 10^{-3}, 2 \times 10^{-4}, 2 \times 10^{-5}, 2 \times 10^{-6}]$; number of epochs $\in \{3, 7, 10, 15\}$; and the batch size $\in \{8, 16\}$. Our final models achieve 90.6% accuracy on SST and 71.2% accuracy on MNLI

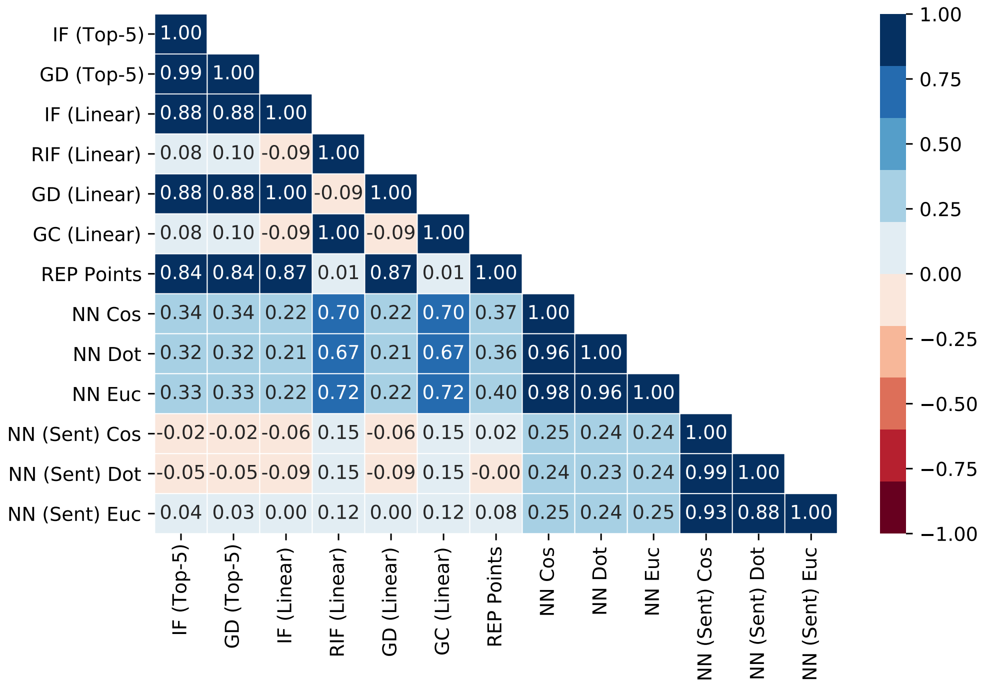
C.2 Correlation Between Attribution Methods

The complete version of spearman correlation between attribution methods (containing the sentence-BERT) is provided in C.1. As expected, similarity-based approaches based on sentence-BERT show a very small correlation with other methods.

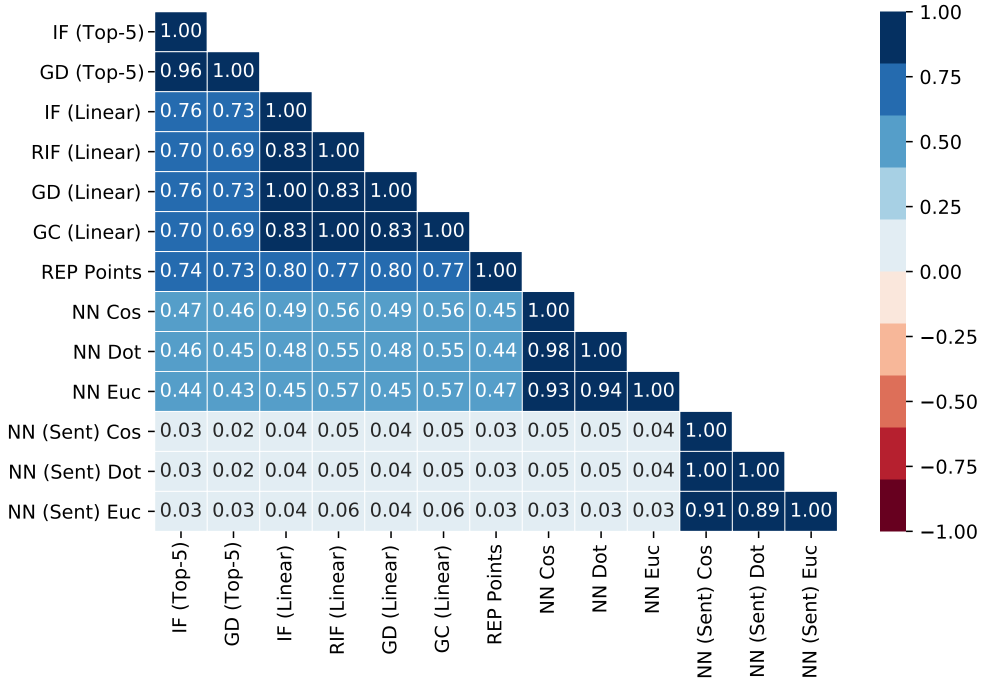
We also provide the proportion of shared examples in the top samples retrieved by IF (top-5) and IF (linear) in Figure C.2. One can see that there is a very high correlation between these methods in top samples, validating the high quality of the simpler version of IF (linear) compared to the more complex method (top-5).

C.3 Removing 'Important' Samples

The proportion of common examples in top samples between pairs of attribution methods is depicted in Figures C.3 and C.4. The high rates between IF vs GD, RIF vs GC, and NN-EUC vs NN-COS pairs, agree with the similar performance of these pairs of methods in leave-some-out experiments.

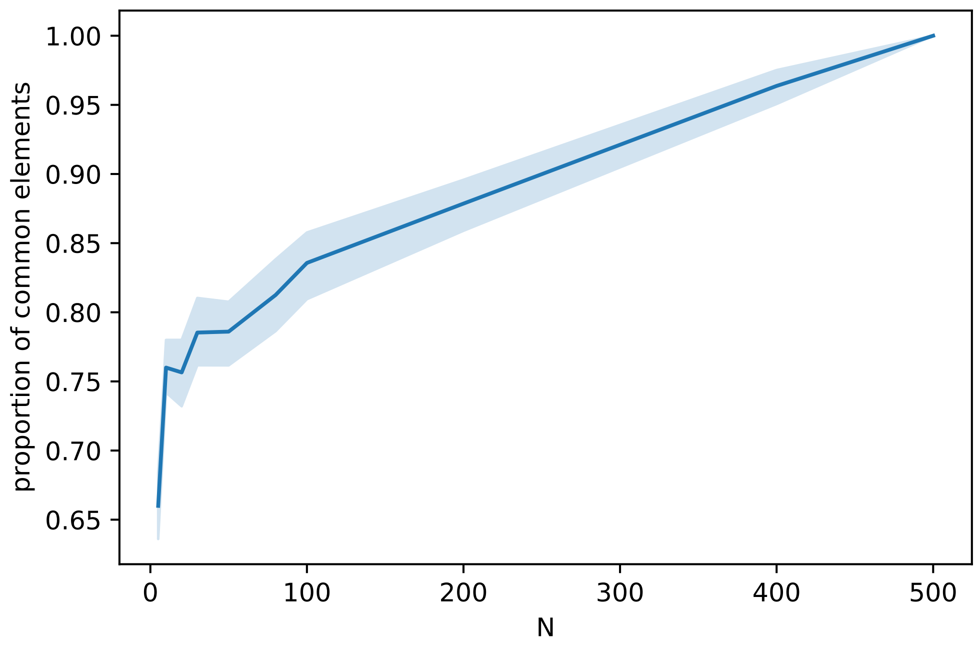


(a) SST.

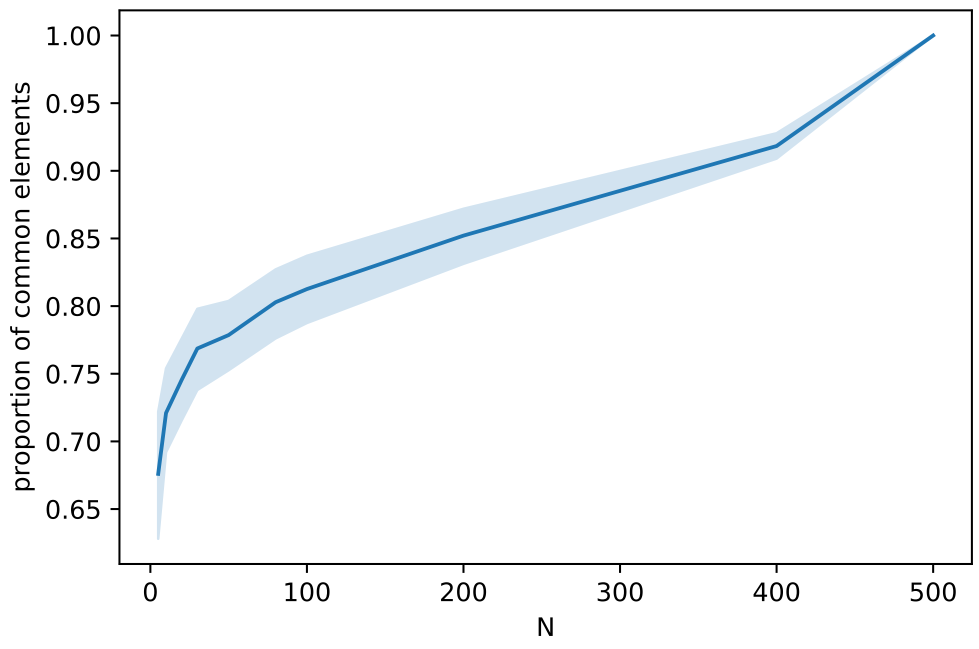


(b) MNLI.

Figure C.1: Complete version of correlation matrices.

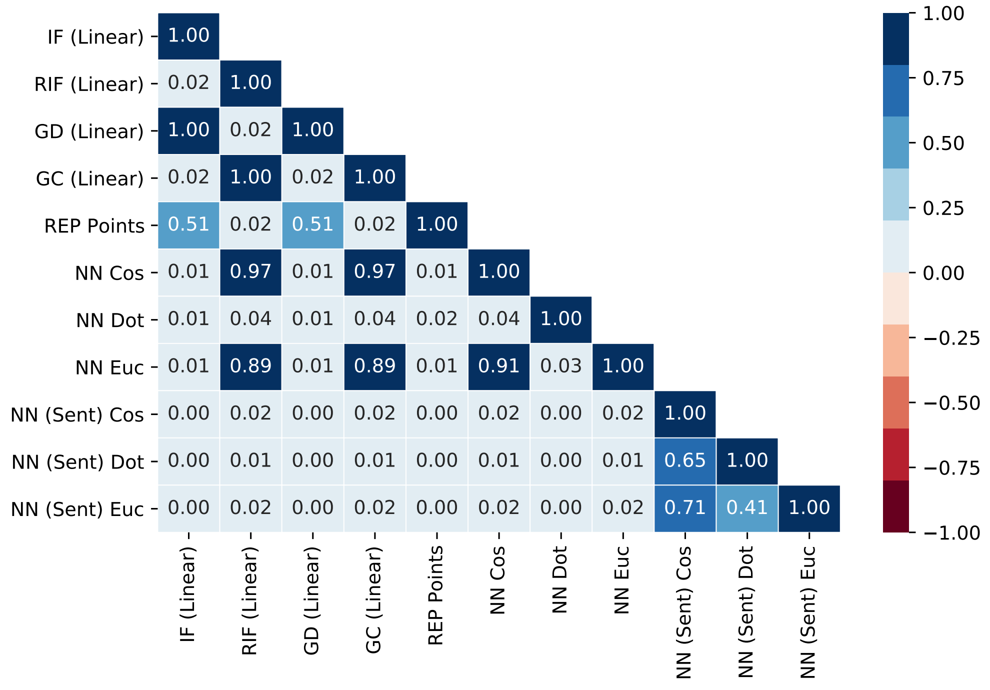


(a) SST.

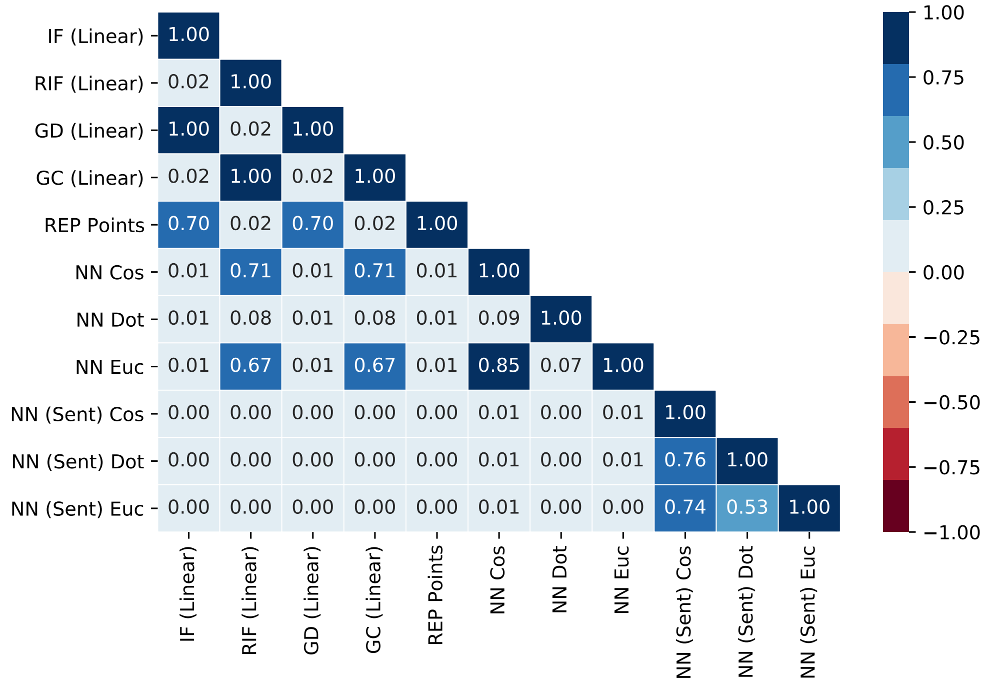


(b) MNLI.

Figure C.2: Proportion of common top examples between IF (Top-5) and IF (Linear) Methods. We selected 100 test examples and 500 training examples to compute the attributions over.

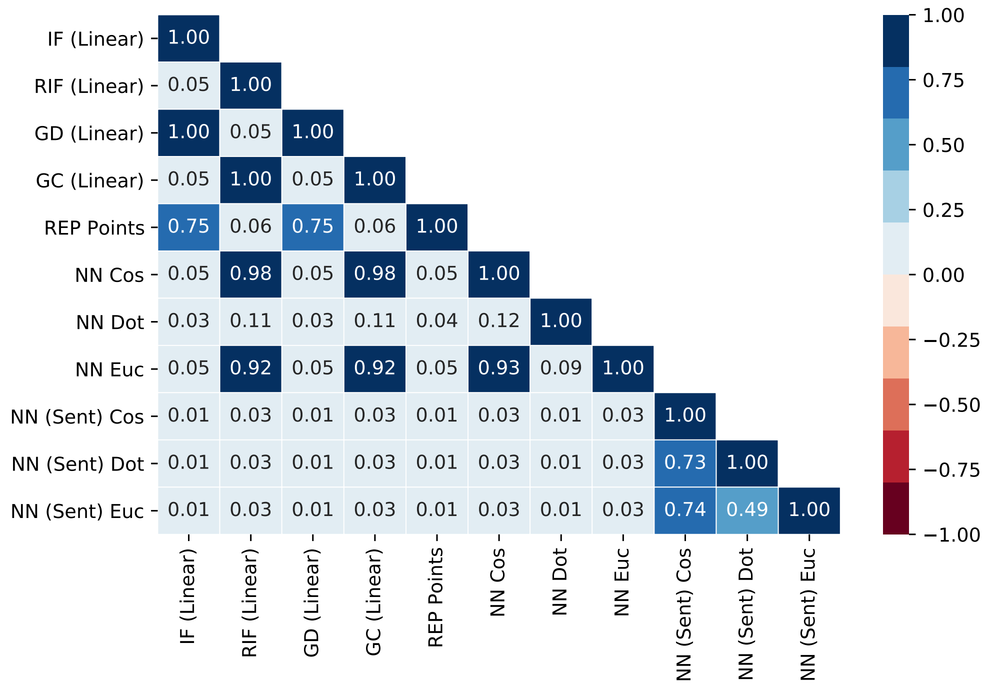


(a) Top-10 in SST.

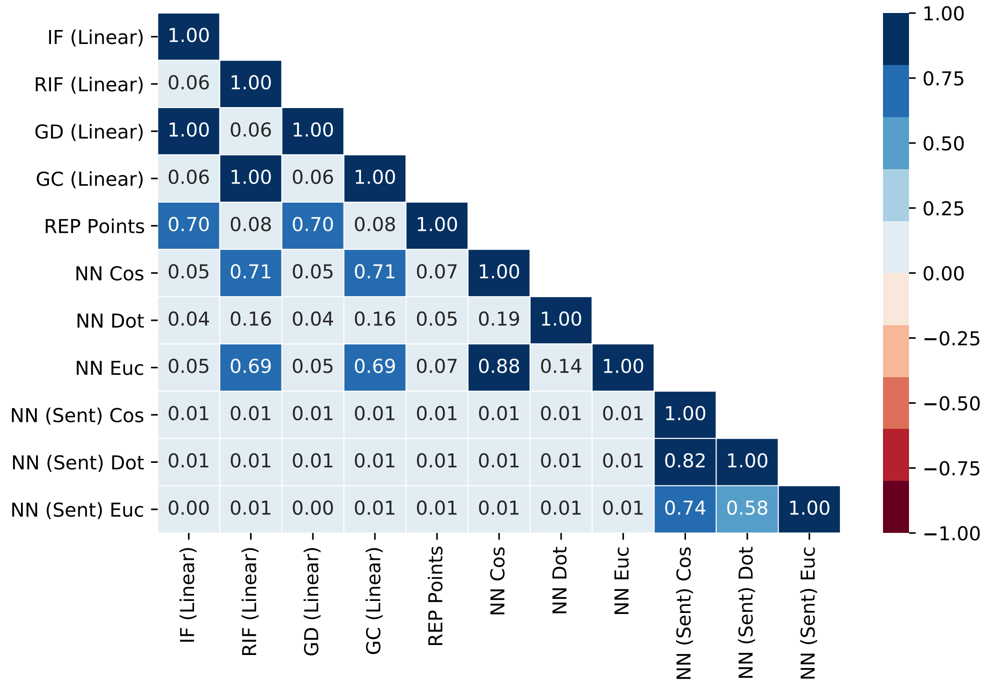


(b) Top-10 in MNLI.

Figure C.3: Proportion of common examples in top 10 samples between pairs of attribution methods.



(a) Top-50 in SST.



(b) Top-50 in MNLI.

Figure C.4: Proportion of common examples in top 50 samples between pairs of attribution methods.

	Method	Lexical Overlap Rate	
		top-1	top-10
	Random	0.40	0.40
Sen-Bert	NN EUC	0.39	0.41
	NN COS	0.38	0.39
	NN DOT	0.39	0.40
Sim	NN EUC	**0.56**	**0.57**
	NN COS	**0.56**	0.56
	NN DOT	0.44	0.44
Gradient	IF	0.43	0.44
	REP	0.43	0.35
	RIF	0.54	0.55
	GD	0.43	0.44
	GC	0.55	0.56

Table C.1: Studying the average lexical overlap rate between premise and hypothesis in top-k most influential samples for test instances mispredicted as entailment.

C.4 Artifacts and Attribution Methods

The average lexical overlap rate for 1000 random samples from the HANS dataset using all different attribution methods is reported in Table C.1.

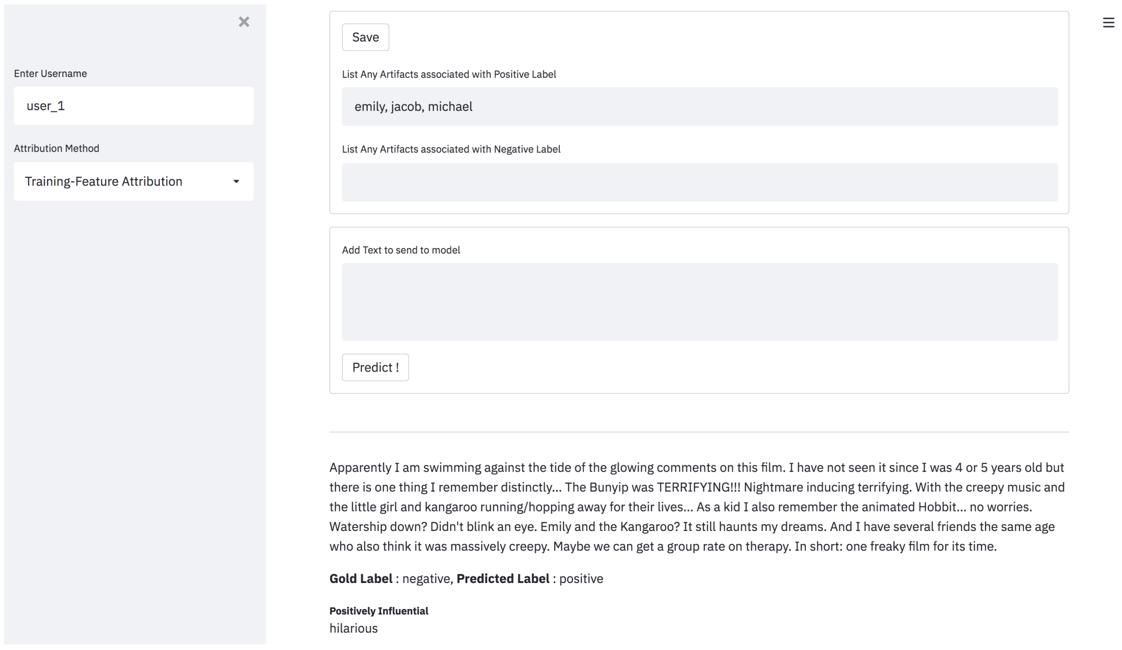
Appendix D

Artifact Discovery

D.1 Experimental Setup

Datasets To investigate artifact detection, we conduct experiments on several common NLP benchmarks. We consider two benchmarks with previously known artifacts: (1) HANS dataset [53], which comprises 30k examples exhibiting previously identified NLI artifacts such as lexical overlap between hypotheses and premises. We randomly sampled 1000 instances from this benchmark as test data and use 10k randomly sampled instances from the Multi-Genre NLI (MNLI) dataset [106], which contains 393k pairs of premise and hypothesis from 10 different genres, as training data. (2) We also use the IMDB binary sentiment classification corpus [50], comprising 25k training and 25k testing instances. It has been shown in prior work [81] that models tend to rely on the presence of ratings (range: 1 to 10) within IMDB review texts as artifacts.

We have also reported novel (i.e., previously unreported) artifacts in several benchmarks. These include: (1) The DWMW17 dataset [24] which is composed of 25K tweets labeled as *hate speech*, *offensive*, or *non-toxic*; (2) BoolQ [20], a question answering dataset which


Enter Username

user_1

Attribution Method

Training-Feature Attribution

Save

List Any Artifacts associated with Positive Label

emily, jacob, michael

List Any Artifacts associated with Negative Label

Add Text to send to model

Predict !

Apparently I am swimming against the tide of the glowing comments on this film. I have not seen it since I was 4 or 5 years old but there is one thing I remember distinctly... The Bunyip was TERRIFYING!!! Nightmare inducing terrifying. With the creepy music and the little girl and kangaroo running/hopping away for their lives... As a kid I also remember the animated Hobbit... no worries. Watership down? Didn't blink an eye. Emily and the Kangaroo? It still haunts my dreams. And I have several friends the same age who also think it was massively creepy. Maybe we can get a group rate on therapy. In short: one freaky film for its time.

Gold Label : negative, **Predicted Label** : positive

Positively Influential
hilarious

Figure D.1: Screenshot of the user study's interface.

contains 16k pairs of yes/no answers and corresponding passages.

Models We adopt BERT [26] with a linear model on top as a classifier and tune hyperparameters on validation data via grid search. Specifically, tuned hyperparameters include the regularization parameter $\lambda = [10^{-1}, 10^{-2}, 10^{-3}]$; learning rate $\alpha = [10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$; number of epochs $\in \{3, 4, 5, 6, 7, 8\}$; and the batch size $\in \{8, 16\}$. Our final model accuracy on the benchmarks are as follows: *IMDB:* 93.2%, *DWMW17:* 91.1%, *BoolQ:* 77.5%.

Calculating the Gradient To calculate gradients for individual tokens, we adopt a similar approach to [6], i.e., calculating the gradient of output (before the softmax), or instance attribution score with respect to the token embedding. We aggregate the resulting vector by taking an average; this has shown to be effective in prior work [6] and provides a sense of positively and negatively influential tokens for model predictions (as compared to using $L2$ norm as an aggregating function).

D.2 User Study

The list of randomly sampled neutral adjectives, most popular names, and the pronouns used as artifacts are as follows: *Adjectives* = [regular, cinematic, dramatic, bizarre ,artistic, mysterious], *First-names* = [Jacob, Michael, Ethan, Emma, Isabella, Emily] and *Pronouns* = [he, his, him, she, her]. We also provide a screenshot of the interface used in our user study in Figure D.1.