VERACITY AND VULNERABILITIES ANALYSIS OF MULTI-SOURCED DATA

by

Hengtong Zhang

August 31, 2020

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

Department of Computer Science and Engineering

The thesis of Hengtong Zhang was reviewed by the following:

Dr. Jing Gao
Associate Professor of Computer Science and Engineering
University at Buffalo, The State University of New York
Thesis Advisor, Chair of Committee

Dr. Aidong Zhang
William Wulf Faculty Fellow and Professor of Computer Science and Biomedical Engineering
University of Virginia
Committee Member

Dr. Varun Chandola
Associate Professor of Computer Science and Engineering
University at Buffalo, The State University of New York
Committee Member

Dedication

This dissertation is dedicated to
my parents, my sister and Dr. Chen Ye

Acknowledgments

It is a terrific stage of life to spend nearly five years perusing my Ph.D. degree, and I am very grateful to have worked with many brilliant people during my Ph.D. career.

First and foremost, I would like to show my greatest gratitude to my advisor Professor Jing Gao, who is not only an incredibly amazing advisor for my Ph.D. career but also an inspirational mentor of my life. I thank her for introducing me to the field of academic research and encouraging me to explore novel, meaningful, and practical solutions for a variety of research problems. She is always able to precisely pointed out the caveats in my proposed methodologies and the errors in my research manuscripts. Her support, patience, knowledge, insight, and advice have greatly helped my growth as a qualified researcher in my current research area. I am so fortunate to work with her over the past several years.

I would also like to thank my committee members: Professor Aidong Zhang and Professor Varun Chandola. They have provided extremely valuable comments and suggestions for my research projects. Professor Aidong Zhang has provided insightful suggestions on selecting meaningful research problems for my dissertation and developing new approaches to solve these problems. From Professor Varun Chandola, I have got precious suggestions on the technical details and the presentation strategies of my research work.

I would also like to express my gratitude to all my collaborators, who are all helpful and friendly people. I have enjoyed the time we spent together. Not only discussions on research projects but also the activities of our spare time. Special thanks come to Dr. Yaliang Li, Professor Lu Su, Professor Fenglong Ma, Professor Kui Ren, Professor Xin Zhao, Professor Ji-Rong Wen, Professor Quanquan Gu, Tianhang Zheng, Dr. Qi Li, Dr. Chen Ye, Dr. Chenglin Miao, Dr. Houping Xiao, Xiaoxuan Hu, Liuyi Yao, Yaqing Wang and Haoyu Wang for their collaboration and valuable suggestions. Thanks to Dr. Bolin Ding, who is a great mentor through my summer internship at Alibaba. U.S. Inc; Dr. Lv-An Tang, Dr. Zhengzhang Chen, Dr. Bo Zong, who are my great mentors during my internship at NEC Labs America. Thank my colleagues and friends, Tianqi Wang, Dr. Wenjun Jiang, Hongfei Xue, Dr. Xian Wu, Huajie Shao, Huaxiu Yao, Qiuling Suo,

Weida Zhong, Ye Yuan, Rui Li and Enshu Wang, and those I have not included their names here, for their great support.

Last but not least, I thank my parents, my sister, and my fiancée Dr. Chen Ye. Your love will always carry me home through the endless time.

Abstract

In the last decade, the research community has witnessed the success of machine learning techniques in a large variety of long-standing applications. The success of these techniques is largely driven by the ubiquitous massive data, which are typically collected from multiple data sources. However, the multi-source data usually contains erroneous information and some errors may be caused by intentional manipulation by adversarial attackers. These issues may cause the failure of machine learning models. Thus, it is crucial to analyze the veracity of the multi-sourced data and study the potential vulnerabilities in the multi-sourced data. In this dissertation, we propose: (1) a series of multi-sourced data reliability analysis methods to discover trustworthy information from correlated data and textual data; and (2) multiple data poisoning attack approaches to help understand the impacts of vulnerabilities in the multi-sourced data on real-world machine learning tasks.

In multi-sourced data reliability analysis, it is critical to identify reliable sources that provide highly-trustworthy information and leverage the data from these reliable sources to better discover trustworthy information. Existing multi-sourced data reliability analysis methods usually make an independent assumption of the data sources, and are generally designed for structured data. In Part I of this dissertation, we develop multiple probabilistic models to resolve these limitations. Particularly, to handle source correlations, the proposed model takes the estimated source correlations as prior, and models the trustworthiness of each piece of information by fusing the trustworthiness of the information provider and its influencers. To identify the reliability of unstructured data like text, we propose another probabilistic model that jointly infers the key factors in the data and estimates the reliability of different data sources. For both models, we conduct extensive experiments on multiple real-world datasets to demonstrate their usefulness and advantages.

Apart from multi-sourced data reliability analysis, we also investigate the impacts of vulnerabilities in the multi-sourced data on real-world machine learning tasks. In Part II of this dissertation, we develop data poisoning attack approaches that inject adversarial samples to multi-sourced data to manipulate the knowledge graph embedding

methods, recommendation models, and outcome interpretation methods. Such adversarial analysis can help understand the impact of vulnerabilities on these machine learning models. Specifically, the proposed attack strategies against knowledge graph embedding methods generate data samples that can manipulate the embedding of knowledge graph entities and further influence the plausibility of arbitrary target facts in the knowledge graph. We also propose a general reinforcement learning-based attack framework, which can manipulate the recommendation results of various representative next-item recommendation models. Moreover, we explore the vulnerability of machine learning outcome interpretation models and investigate whether attacks can manipulate the interpretations of target samples by injecting well-crafted samples to the training set. All these attack strategies and frameworks are tested on real-world benchmark datasets. The experimental results clearly demonstrate that attackers can indeed craft vulnerabilities in the multi-sourced data to manipulate the results produced by existing machine learning models.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The past decade has witnessed the astonishing success of a large variety of data mining and machine learning models in a variety of real-world tasks, such as image classification, question answering, and recommendation systems. The success of these models is largely owing to the rapidly growing big data. To obtain the data effectively and efficiently, researchers from both industry and academia collect it from a collection of platforms and individuals. However, the data collected from multiple data sources may consist of erroneous information due to transmission errors, device malfunction, or even intentional manipulation. Such untrustworthy or even manipulated data may cause the failure of downstream models and produce implausible decisions for model users. Thus, it is important to: (1) analyze the trustworthiness of multi-source data; and (2) understand the possible vulnerabilities that lead to the failure of machine learning models in the multi-source data. In the rest of this chapter, for both of these tasks, we first provide a brief background introduction, which motivates the research presented in the dissertation, then we summarized the specific tasks that are included in the rest of this dissertation.

1.1 Trustworthiness Analysis of Multi-Sourced Data

In many real world scenarios, descriptions on the same set of objects or events are collected from multiple information sources. These sources are of different reliabilities and the trustworthiness of the information from these sources are also different. Before using the data in downstream models, it is crucial to analyze the multi-sourced data and discover the trustworthy information. In recent years, an advanced technique named truth discovery [18, 35, 36, 42, 46, 47, 82, 83, 87, 89, 92], which integrates multi-sourced information by estimating the reliability of each source, has been proposed. Truth discovery techniques are motivated by a general principle: if a source provides many true claims, the trustworthiness of the source is high; and if a claim is supported by many trustworthy sources, this claim is likely to be true. Following this principle, the trustworthy information are inferred from the multi-sourced data by taking the source reliabilities into consideration. In this dissertation, we advance the current truth discovery techniques from the following perspectives:

1. **Truth Discovery with Correlated Data Sources**: Most truth discovery methods assume that sources make their claims independently, which may not be true in real practice. As a matter of fact, influences among sources are ubiquitous and the claims made by one source may be influenced by others. Although there is some work that considers source correlation, those methods are designed to handle categorical claims, which is not general enough to represent the complicated real world applications. Motivated by this gap, we propose a novel model to handle data correlations for different data types during truth discovery process.

2. **Truth Discovery for Unstructured Text Data**: Most existing truth discovery methods are designed for structured data, and cannot meet the strong need to extract trustworthy information from raw text data as text data has its unique characteristics. The major challenges of inferring true information on text data stem from the multifactorial property of text answers (i.e., an answer may contain mul-

tiple key factors) and the diversity of word usages (i.e., different words may have the same semantic meaning). To tackle these challenges, we propose truth discovery techniques that are specifically designed for unstructured text data.

1.2 Vulnerabilities Analysis in Multi-Sourced Data

In this dissertation, we also analyze the possible vulnerabilities in the multi-sourced data that lead to the failure of machine learning models. Typically, these vulnerabilities are exposed by data samples constructed by the adversarial attacks to make the downstream machine learning models produce specific decisions. To validate the existence and demonstrate the impacts of such vulnerabilities, we conduct the study from the perspective of adversarial attacker and propose a series of attack approaches that construct vulnerabilities in the multi-sourced data. The proposed work falls into the category of data positing attacks [5, 50, 51, 69], whose objective is to construct vulnerabilities in the training data of machine learning models to enforce a nefarious model. In this dissertation, we investigate data poisoning attacks on several representative machine learning applications:

1. **Data Poisoning Attack against Knowledge Graph Embedding**: Knowledge graph embedding (KGE) is a technique for learning continuous embeddings for entities and relations in the knowledge graph. Despite its effectiveness in a benign environment, KGE's robustness to adversarial attacks is not well-studied. Existing attack methods on graph data cannot be directly applied to attack the embeddings of knowledge graph due to its heterogeneity. To fill this gap, we propose a collection of data poisoning attack strategies, which can effectively manipulate the plausibility of arbitrary targeted facts in a knowledge graph by adding or deleting facts on the graph.

2. **Data Poisoning Attack against Next Item Recommendation**: Online recommendation systems make use of a variety of information sources to provide the

items that users are potentially interested in. However, due to the openness of the online platform, recommendation systems are vulnerable to data poisoning attacks. Existing attack approaches are either based on simple heuristic rules or designed against specific recommendation approaches. The former often suffers from unsatisfactory performance, while the latter requires strong knowledge of the target system. In this dissertation, we focus on a general next-item recommendation setting and propose a practical poisoning attack approach named LOKI against blackbox recommendation systems.

3. **Data Poisoning Attack against Outcome Interpretations**: Recently how to interpret the outcome of a machine learning model has attracted much attention. Although the effectiveness of outcome interpretation approaches has been shown in a benign environment, their robustness against data poisoning attacks (i.e., attacks at the training phase) has not been studied. As the first work towards this direction, we aim to answer an important question: Can the outcome interpretation output of target samples be easily manipulated by injecting adversarial samples with unnoticable changes into the training set? To answer this question, we propose a data poisoning attack framework named IMF (**I**nterpretation **M**anipulation **F**ramework). The framework can effectively manipulate the interpretations of target samples produced by representative outcome interpretation methods while the prediction results of these target samples remain unchanged. The proposed framework crafts poisoning samples to encircle the target sample in the feature space and to force the interpretation result to be similar to the one that the attacker desires. The effectiveness and efficiency of the proposed attack strategies are verified by extensive evaluations on a real-world dataset.

Part I

Trustworthiness Analysis of Multi-Sourced Data

Chapter 2

Truth Discovery with Correlated Data Sources

2.1 Introduction

The past decades have witnessed an explosion of data collected from a variety of channels, such as web-scale search engines, crowd-sourcing platforms, and social media platforms. Integrating data from disparate sources can lead to novel insights in scientific, industrial, and governmental domains. However, claims about the same entity may conflict each other due to recording errors, noise, machine failures, malicious attacks, etc. Therefore, to get the most trustworthy information (i.e. the true facts), the aggregation of multi-source data needs to be applied so that the noise from individual sources can be mitigated.

To simplify the design, many existing truth discovery methods make an assumption that data sources provide their claims independently. However, in real life, explicit and implicit influences among sources are ubiquitous, which makes this assumption invalid. For example, on social media platforms, a person can be easily influenced by others when he/she makes a claim towards an entity or an event. Therefore, the claim may come from not only his/her own knowledge but also the knowledge of his/her friends. In such

cases, the provided claims are no longer independent. When estimating the true facts, we need to consider the trustworthiness of the source itself and its related sources. If the influences among sources are ignored and the provided claims are treated as independent ones, it is likely that the estimation of source trustworthiness is inaccurate, and thus the performance of truth discovery is degraded. Therefore, it is crucial to take these inter-source influences into consideration to more accurately estimate source trustworthiness for truth discovery tasks. There is some truth discovery work that considers source correlations [18, 77]. However, these methods are limited in the data types that can be applied to. Specifically, they all treat claims as categorical, which cannot represent the complicated real-world applications. As shown in [35, 36, 91], numerical data and heterogeneous data are also common in truth discovery problems, and treating them as categorical data is inappropriate.

To tackle these challenges, we propose a novel approach named **I**nfluence-**A**ware **T**ruth **D**iscovery (*IATD*), an unsupervised full Bayesian model which can utilize the pre-known source correlations as the prior. Different from existing truth discovery work, IATD introduces the concept of "claim trustworthiness", which fuses the trustworthiness of the source which provides the claim and the trustworthiness of its influencers by an ensemble parameter. Such a design enables us to precisely model the degree of inter-source influences as well as their relations towards source trustworthiness estimation. Using different distributions, we manage to utilize the fused "claim trustworthiness" for the generation of claims which are of different data types.

2.2 Problem Definition

Before describing our proposed model, we start by introducing some terminologies, followed by the formal problem definition.

Definition 1. *An* entity *is an item of interest. The set of M entities is denoted as $V = \{v\}_1^M$.*

Definition 2. *A* source *is the place where information about entities is collected. The set of sources is denoted as* $S = \{s\}_1^N$.

Definition 3. *A* claim *is defined as a piece of information provided by a source towards an entity, and the set of claims is denoted as* C. *We use* $C_{.,v}$ *to denote the set of claims for entity* v, *and* $c_{s,v}$ *denotes the claim from source* s *on entity* v.

Definition 4. *A* truth *is the most trustworthy piece of information for an entity. The set of truths is denoted as* $T = \{t\}_1^M$.

Definition 5. *An influence set* $A_{s,v}$ *is a set of sources that may influence source* s, *when* s *makes a claim on entity* v.

Definition 6. ***(Problem definition)***
Given a set of sources $S = \{s\}_1^N$, *a set of entities* $V = \{v\}_1^M$, *a set of claims* $C_{s=1,v=1}^{N,M}$ *and a set of influencers for every claim* $A_{s=1,v=1}^{N,M}$, *the goal of our proposed model is to learn the estimated truths for entities* $T = \{t\}_1^M$ *as well as the trustworthiness for the sources.*

Note: In this chapter, we only consider the single truth scenario, i.e., for each entity, there is only one truth.

2.3 Methodology

In this section, we describe the IATD model in details. We first provide a high-level overview of the proposed model. This is followed by a detailed mathematical specification. Finally, we provide a comprehensive description of the model fitting procedure based on Expectation Maximization.

2.3.1 Overview

The IATD model specifies a two-stage generative process of claims. The first stage specifies the generation of sources' individual trustworthiness as well as the influence-

Table 2.1: Notations

Notations	Meaning
N	the number of sources in the dataset
M	the number of entities in the dataset
λ	global influence smoothing parameter
σ_s	individual source deviation for source s
b_v	entity-specific bias variable for entity v
$g^{(c)}$, $g^{(n)}$	pre-tuned global deviation bias for categorical and numerical data respectively
γ_v	the number of possible discrete values for entity v with categorical claims
$C_{s,\cdot}$	the set of claims given by source s
$C_{\cdot,v}$	the set of claims on entity v
$c_{s,v}$	the claim given by source s on entity v
t_v	estimated truth of entity v
$\epsilon_{s,v}$	the deviation value of the claim given by source s on entity v
$A_{s,v}$	the set of sources which may influence s, when making claim $c_{s,v}$
α_e, β_e	parameters for Inverse-Gamma prior of source deviation
μ_b, σ_b^2	parameters for Gaussian prior of entity-specific difficulty variable
μ_t, σ_t^2	parameters for Gaussian prior of the true value of numerical entities

aware trustworthiness fusion given the influencer set for each claim. The second stage specifies the generation of heterogeneous claims, given the "claim trustworthiness" of each claim. Here, we describe the intuition behind the modeling before detailing the IATD model.

Trustworthiness Generation: According to our investigation of the real world data, we find that source's final decisions are usually based on the combination of its own trustworthiness and its trusted sources' trustworthiness. A source may make its claims based on its own trustworthiness, but may be influenced by the sources it trusts at the same time. In our model, we introduce the "claim trustworthiness", which fuses

the trustworthiness of current source and the trustworthiness of potential influencers linearly by an ensemble parameter. This can better explain the phenomenon when the credibility of multiple claims from a single source is inconsistent sometime. Moreover, in our model, if a source makes true claims for some entities, the trustworthiness of this source and its influencers will be increased. On the contrary, if a source makes false claims for some entities, this source and its influencers will all suffer a decrease in their trustworthiness. In this chapter, in order to fit both categorical and numerical data, we evaluate the source trustworthiness using a variable σ, which models the claim deviation tendency of a source. The value of σ is inversely proportional to the trustworthiness degree of a single source.

Truth Generation: In IATD, we discuss two typical types of entities, i.e. categorical and numerical. For entities with categorical claims, the true values are modeled as random variables following uniform distributions, as we assume there is a single true value for each entity and the false values should be uniformly distributed. For entities with numerical claims, we model the true values as a random variable following Gaussian distributions. These distributions are commonly used to model categorical and numerical data, respectively.

Claim Generation: Once we get the claim trustworthiness and truth for each claim, we can utilize these variables to model the generation process for claims. For both types of claims, we assume that the claim from a source is associated with: (1) the claim deviation, and (2) the difficulty of the entity. For entities with categorical claims, we consider the posterior of a claim to be true, given the estimated truth. If the claim deviation is high, the probability of the claim being true should be small, and vice versa. For entities with numerical claims, we model the generation of claim using a Gaussian distribution, whose mean is the truth, and the variance parameter is the claim deviation. This indicates that high trustworthiness claims have smaller deviations from the truth. Moreover, for different entities, the difficulty of obtaining true claims may differ. Therefore, we introduce the entity-specific difficulty parameter to model this phenomenon.

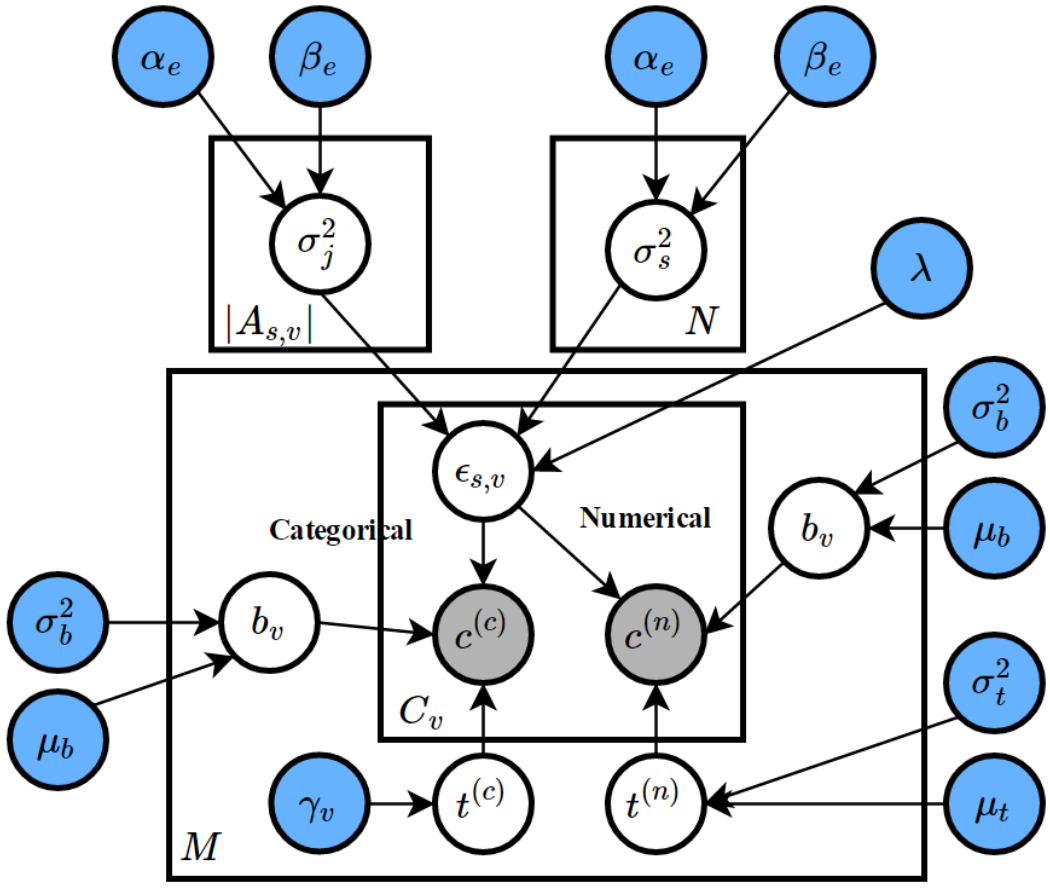


Figure 2.1: Plate notation for the proposed IATD Model.

Fig. 2.1 shows the graphical structure of the model, and Table 2.1 lists the descriptions of the variables. The generative process is shown as follows:

- For the s-th source $(s = 1, 2, \cdots, N)$
  - Draw $\sigma_s^2 \sim \Gamma^{-1}(\alpha_e, \beta_e)$.
- For the v-th entity $(v = 1, 2, \cdots, M)$
  - For an entity with categorical claims v, draw a true fact $t_v \sim \mathrm{U}(\gamma_v)$.
  - For an entity with numerical claims v, draw a true fact $t_v \sim \mathcal{N}(\mu_t, \sigma_t^2)$.
- For each claim $c_{s,v}$

- For an entity with categorical claims v, draw a claim $c_{s,v} \sim logistic(-\epsilon_{s,v} + b_v + g^{(c)})$.
- For an entity with numerical claims v, draw a claim $c_{s,v} \sim \mathcal{N}(t_v, (\epsilon_{s,v} + b_v + g^{(n)})^2)$.

2.3.2 Model Specification

In this section, we provide a detailed description of the proposed model. We first specify the generation of sources' individual trustworthiness and "influence-aware trustworthiness fusion". Then we describe the generation of different claims separately based on claim trustworthiness.

2.3.2.1 Trustworthiness Modeling

The source and claim trustworthiness are modeled via deviation variables σ as follows. For each source s, we draw σ_s^2 from an Inverse-Gamma distribution with hyper-parameters (α_e, β_e), where α_e and β_e are shape and rate parameters respectively. Therefore,

$$\sigma_s^2 \sim \Gamma^{-1}(\alpha_e, \beta_e) \propto (\sigma_s^2)^{-\alpha_e - 1} \exp(-\beta_e \sigma_s^{-2}). \tag{2.1}$$

Mathematically, σ_s^2 is the variance variable for Gaussian distribution and the value of σ_s^2 is inversely proportional to the trustworthiness of a source. When parameter $\alpha_e > 1$ the Inverse-Gamma distribution concentrates mostly around its mode $\frac{\beta_e}{\alpha_e + 1}$. This generally means that the deviation of most sources should be around a certain value and there are a few with much higher or lower deviation in our assumption.

Given the definition of individual deviation for every source, we can further model the phenomenon that a source s gets influenced by others when generating claims for entity v. To model the claim deviation, we introduce an auxiliary variable $\epsilon_{s,v}$, which denotes the deviation of source s when it offers a claim on entity v (i.e. claim deviation). This variable reflects both the deviation of the source itself and the deviation of sources

it relates. Let σ_s and σ_j denote the source deviation of s and j, and $A_{s,v}$ be the set of sources that may influence s when s makes a claim on entity v. We can model the trustworthiness fusion as follows:

1. If we can infer that a user is influenced by others, then $\epsilon_{s,v}$ is defined as:

$$\epsilon_{s,v} = \frac{\lambda}{|A_{s,v}|} \sum_{j \in A_{s,v}} \sigma_j + (1 - \lambda) \cdot \sigma_s. \tag{2.2}$$

2. If there is no evidence that the source is influenced (i.e. $A_{s,v} = \Phi$), then

$$\epsilon_{s,v} = \sigma_s. \tag{2.3}$$

Note that the value of the auxiliary variable ϵ can be calculated directly given two deviation variables σ_s and σ_j. Therefore, it does not lead to the increase the number of parameters.

Further, as the difficulties of obtaining the value of entities may be different, we introduce an entity-specific bias variable b_v to make some adjustments. b_v is drawn from a Gaussian distribution:

$$b_v \sim \mathcal{N}(\mu_b,\ \sigma_b^2), \tag{2.4}$$

where μ_b and σ_b^2 are the parameters of the Gaussian distribution.

2.3.2.2 Claim Modeling

Given the claim trustworthiness $\epsilon_{s,v}$ and entity-specific bias b_v, we now describe the posterior probability of observed claims on entity v from the source s, i.e., $c_{s,v}$, given the latent true fact t_v. The intuition is straightforward: the sources of low deviation often provide more trustworthy claims. Since the claim generation process for categorical and numerical claims are different, we handle them using different formulations.

Categorical Claim Modeling: For categorical claims, we model the probability of a claim $c_{s,v}$ being true using a Bernoulli distribution. Intuitively, the probability that

source s offers a true claim to entity v relies on: (1) the claim deviation, and (2) the difficulty of obtaining the true value of the entity. Specifically, the probability is defined as:

$$p(c_{s,v} = x | t_v = x, \epsilon_{s,v}, b_v) = h(-\epsilon_{s,v} + b_v + g^{(c)}), \tag{2.5}$$

where $h(\cdot)$ is a logistic function, and $g^{(c)}$ is a pre-tuned global bias for entities with categorical claims. We can see that if the claim deviation $\epsilon_{s,v}$ is small, the probability of $c_{s,v}$ being a true claim is large, and vise versa.

We assume the probability that source s offers an incorrect claim to entity v is from a Uniform distribution, so the probability that source s offers an incorrect claim to entity v is modeled as:

$$p(c_{s,v} \neq x | t_v = x, \epsilon_{s,v}, b_v) = \frac{1 - h(-\epsilon_{s,v} + b_v + g^{(c)})}{\gamma_v - 1}. \tag{2.6}$$

Combining Eq. (2.5) and Eq. (2.6), we can get the probability that source s makes the claim $c_{s,v}$, given the claim deviation $\epsilon_{s,v}$, the truth estimation t_v, and the entity-specific bias variable b_v:

$$\begin{aligned} & p(c_{s,v} | t_v = x, \epsilon_{s,v}, b_v) \\ = & \, h(-\epsilon_{s,v} + b_v + g^{(c)})^{\delta(c_{s,v}, t_v)} \\ & \times \left(\frac{1 - h(-\epsilon_{s,v} + b_v + g^{(c)})}{\gamma_v - 1} \right)^{1 - \delta(c_{s,v}, t_v)}, \end{aligned} \tag{2.7}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta function.

Numerical Claim Modeling: For entities with numerical claims, we draw a true claim from a Gaussian distribution

$$t_v \sim \mathcal{N}(\mu_t, \, \sigma_t^2), \tag{2.8}$$

where μ_t controls the prior estimate of the truth and σ_t^2 captures the prior deviation. The probability of source s offering a correct claim to entity v is modeled as a Gaussian distribution with the mean of the estimated truth t_v, and the deviation of $(\epsilon_{s,v}+b_v+g^{(n)})^2$, i.e.,

$$\begin{aligned} p(c_{s,v}|t_v = x, \epsilon_{s,v}, b_v) &\sim \mathcal{N}(t_v,\ (\epsilon_{s,v} + b_v + g^{(n)})^2) \\ &\propto (\epsilon_{s,v} + b_v + g^{(n)})^{-1} \exp\left(- \frac{(c_{s,v} - t_v)^2}{2(\epsilon_{s,v} + b_v + g^{(n)})^2}\right), \end{aligned} \tag{2.9}$$

where $g^{(n)}$ is a pre-tuned global bias for entities with numerical claims.

The truth of entity v, the claim deviation $\epsilon_{s,v}$, and the v's bias variable b_v jointly capture the precision of the claim. When claim deviation $\epsilon_{s,v}$ and/or bias variable b_v get smaller, the claim $c_{s,v}$ should be closer to the truth, and vise versa.

2.3.3 Discussion

Here we describe the influence derivation, which can have effects on the final performance.

Let $c_{s,v}$, which is provided by source s towards entity v, be the claim we are currently working on. Influence derivation is to determine the set of sources that potentially influence s when s provides claim $c_{s,v}$. For brevity, we only introduce a general and straightforward way for influence derivation. Let $A_{s,v}$ be the source set who may influence s, when it makes claim $c_{s,v}$. If there is a directed relation from s to j and they make the same false claim on entity v, we treat s to be influenced by j, when s provides its claim $c_{s,v}$, i.e.,

$$j \in A_{s,v} \iff \begin{cases} \exists\ s \to j, \\ c_{s,v} = c_{j,v}, \\ c_{s,v} \neq \hat{t}_v, \end{cases} \tag{2.10}$$

where $\hat{t}_v$ is the estimated truth for entity v.

Note that more sophisticated influence detection methods can be deployed here. For simplicity, we choose this approach.

2.3.4 Model Fitting

The fitting process is to estimate the value of hidden source deviation σ_s and the true fact t of each entity, given the set of claims and source correlations. The negative log-likelihood of observations, latent variables, and parameters given the hyper-parameters can be written as:

$$\begin{aligned}\mathcal{L} &= -\log(p(C, \boldsymbol{t}, \boldsymbol{\epsilon}, \boldsymbol{b}|\alpha_e, \beta_e, \sigma_b^2, \mu_t, \sigma_t^2)) \\ &= -\sum_{s=1}^{N} p(\sigma_s^2|\alpha_e, \beta_e) - \sum_{v=1}^{M} p(t_v|\mu_t, \sigma_t^2) - \sum_{v=1}^{M} p(b_v|\mu_b, \sigma_b^2) \\ &\quad - \sum_{s=1}^{N}\sum_{v=1}^{M} p(c_{s,v}|\epsilon_{s,v}, t_v, b_v)\end{aligned} \tag{2.11}$$

For entities with numerical claims, Eq. (2.11) is formulated as:

$$\begin{aligned}\mathcal{L} \propto & \sum_{s=1}^{N}\sum_{v=1}^{M}\left(\frac{(c_{s,v}-t_v)^2}{2(b_v+\epsilon_{s,v}+g^{(n)})^2} + \log(b_v+\epsilon_{s,v}+g^{(n)})\right) \\ & + \sum_{s=1}^{N}\left(2(1+\alpha_e)\log\sigma_s + \beta_e\sigma_s^{-2}\right) + \sum_{v=1}^{M}\frac{(b_v-\mu_b)^2}{2\sigma_b^2} \\ & + \sum_{v=1}^{M}\frac{(t_v-\mu_t)^2}{2\sigma_t^2}.\end{aligned} \tag{2.12}$$

For entities with categorical claims, Eq. (2.11) is formulated as:

$$\begin{aligned}\mathcal{L} \propto & \sum_{s=1}^{N}\sum_{v=1}^{M}\Big(-\delta(c_{s,v}, t_v)\cdot\log(h(-\epsilon_{s,v}+b_v+g^{(c)})) \\ & - (1-\delta(c_{s,v}, t_v))\cdot\log(1-h(-\epsilon_{s,v}+b_v+g^{(c)}))\Big) \\ & + \sum_{s=1}^{N}\left(2(1+\alpha_e)\log\sigma_s + \beta_e\sigma_s^{-2}\right) + \sum_{v=1}^{M}\frac{(b_v-\mu_b)^2}{2\sigma_b^2}.\end{aligned} \tag{2.13}$$

To fit such models with latent variables, we refer to EM algorithm. In E-Step, the algorithm takes the expectation of complete data likelihood with respect to the posterior of latent variable t, and in M-Step it maximizes the expectation of complete data likelihood from E-Step to update model parameters σ_s, σ_j, and b_v.

E-Step. For entities with categorical claims, the major computational bottleneck in E-Step is that the posteriors of latent variables are not available in a closed form. Hence, we take recourse to Monte Carlo methods, which perform Gibbs sampling to randomly sample variables from their posterior distributions. Specifically, we sample t_v as:

$$\begin{aligned} & p(t_v = x | \epsilon_{s,v}, b_v, g^{(c)}) \\ & \propto p(t_v = x) \prod_{s=1}^{N_v} p(c_{s,v} | t_v = x, \epsilon_{s,v}, b_v, g^{(c)}). \end{aligned} \tag{2.14}$$

Eq. (2.14) can be calculated directly using Eq. (2.7).

For entities with numerical claims, we can get the closed-form expression of t_v by solving $\frac{\partial \mathcal{L}}{\partial t_v} = 0$ using Eq. (2.12), which is:

$$t_v = \frac{\mu_t \sigma_t^{-2} + \sum_{s=1}^{N_v} c_{s,v} (\epsilon_s + b_v + g^{(n)})^{-2}}{\sigma_t^{-2} + \sum_{s=1}^{N_v} (\epsilon_s + b_v + g^{(n)})^{-2}}. \tag{2.15}$$

Given the value of latent variables, we can derive the expression of complete data likelihood.

M-Step. In M-Step, we need to find the parameters that maximize the likelihood computed in the E-Step. As the deviation variables σ_s and σ_j are above zero, the optimization problem should be formulated as:

$$\begin{aligned} & \min_{\sigma_s, \sigma_j, b} \ \mathcal{L} \\ & s.t. \ \sigma_s > 0, \sigma_j > 0. \end{aligned} \tag{2.16}$$

Intuitively, we can solve that optimization objective by adapting Gradient approaches directly. However, to decrease the computation cost, we propose to solve the following

optimization problem on each claim separately, which approximates the problem in Eq. (2.16):

$$\begin{aligned} &\min_{\sigma_s,\sigma_j,b} \ \mathcal{L}_{s,v} \\ &s.t.\ \sigma_s > 0, \sigma_j > 0. \end{aligned} \tag{2.17}$$

Here, for entities with numerical claims, $\mathcal{L}_{s,v}$ is formulated as:

$$\begin{aligned} \mathcal{L}_{s,v} \propto\ & \frac{(c_{s,v} - t_v)^2}{2(b_v + \epsilon_{s,v} + g^{(n)})^2} + \log(b_v + \epsilon_{s,v} + g^{(n)}) \\ & + \frac{1}{|C_{s,\cdot}|}\left(2(1+\alpha_e)\log\sigma_s + \beta_e\sigma_s^{-2}\right) \\ & + \frac{1}{|C_{\cdot,v}|}\frac{(b_v - \mu_b)^2}{2\sigma_b^2} + \frac{1}{|C_{\cdot,v}|}\frac{(t_v - \mu_t)^2}{2\sigma_t^2}. \end{aligned} \tag{2.18}$$

For entities with categorical claims, $\mathcal{L}_{s,v}$ is formulated as:

$$\begin{aligned} \mathcal{L}_{s,v} \propto\ & -\delta(c_{s,v}, t_v)\cdot\log(h(-\epsilon_{s,v} + b_v + g^{(c)})) \\ & -(1-\delta(c_{s,v}, t_v))\cdot\log(1 - h(-\epsilon_{s,v} + b_v + g^{(c)})) \\ & + \frac{1}{|C_{s,\cdot}|}\left(2(1+\alpha_e)\log\sigma_s + \beta_e\sigma_s^{-2}\right) \\ & + \frac{1}{|C_{\cdot,v}|}\frac{(b_v - \mu_b)^2}{2\sigma_b^2}. \end{aligned} \tag{2.19}$$

It is difficult to derive the optimal closed-form solution for those variables. Therefore, we use Projection Gradient (PG) method for model fitting. PG is an extension of gradient descent method, and is commonly used for solving linearly constrained problems. In Eq. (2.17), the optimization problem projects σ_s (or σ_j) onto $(0, \infty)$. In our implementation, σ_s (or σ_j) is mapped as following:

$$\mathrm{P}(\sigma) = \begin{cases} \sigma & \text{if } \sigma > 10^{-5} \\ 10^{-5} & \text{if } \sigma \leq 10^{-5} \end{cases}.$$

Given the projection function P, the update of σ is defined by:

$$\sigma_s^{(k+1)} \leftarrow P(\sigma_s^{(k)} - \eta_k \frac{\partial \mathcal{L}_{s,v}}{\partial \sigma_s^{(k)}}), \tag{2.20}$$

$$\sigma_j^{(k+1)} \leftarrow P(\sigma_j^{(k)} - \eta_k \frac{\partial \mathcal{L}_{s,v}}{\partial \sigma_j^{(k)}}), \tag{2.21}$$

where $P(\cdot)$ denotes the projection from $\mathbb{R}^n$ onto $\mathbb{R}^+$.

For b_v, as there is no constraint, we can simply deploy gradient descent for parameter update.

$$b_v^{(k+1)} \leftarrow b_v^{(k)} - \eta_k \frac{\partial \mathcal{L}_{s,v}}{\partial b_v^{(k)}}. \tag{2.22}$$

To speed up the PG, one crucial part is the tuning of step length η_k. It keeps changing in each iteration. There are a variety of strategies for searching η_k and we use the algorithm in [43].

2.4 Experiments

In this section, we present and analyze the experimental results on both real-world and simulated data. The results show that the proposed IATD method outperforms state-of-the-art truth discovery approaches. We first state the overall experiment settings in Section 2.4.1. Then we demonstrate and analyze the results on real-world and simulated data respectively.

2.4.1 Experiment Settings

Baseline Methods: To evaluate the effectiveness of the proposed model, we compare it with the following baseline methods:

- *Voting*: The truth estimates are obtained by the value which has the highest number of occurrences in the claim set.

- *GTM* [91]: GTM is a full Bayesian approach designed for truth discovery on numerical data.[1]

- *Invest* [61]: In this method, sources distribute their reliability scores uniformly on the claims they provide, and then collect their credibility from the confidence of those claims.

- *Pooled Invest* [61]: This method is a variant of *Invest*. The difference is that the confidence of claims is linearly scaled for *Pooled Invest*.

- *2-Estimate* [21]: 2-Estimate is an approach based on the assumption that "there is a single true value for each entity". Therefore, *2-Estimate* models the complementary votes.

- *3-Estimate* [21]: 3-Estimate extends *2-Estimate* by considering the difficulty of obtaining the true claim for an entity.

- *TruthFinder* [87]: TruthFinder is a Bayesian-based approach, which computes the probability of a claim being true given the sources. Claim similarity is modeled as an implication function.

- *AccuSim* [18]: AccuSim is a Bayesian approach that is similar to TruthFinder. However, it considers complementary votes for claims, which is similar to 2-Estimate. AccuSim also considers claim similarity.

- *CRH* [36]: CRH is an optimization framework which handles different data types jointly. The goal of the optimization problem is to minimize the weighted loss of the aggregation results.

- *IATD-ni*: This is a variant for the proposed IATD, which does not take inter-source influence information into consideration.

[1] **Note**: This approach is used on numerical data only, as it is incompatible with categorical data.

The parameters of baseline methods are set according to their authors' suggestions.

Evaluation Metrics: To evaluate the performance of truth discovery methods, two classical metrics are employed for different types of data. For both metrics, a lower value indicates a better performance.

- *Error Rate*: This metric is used for performance evaluation on categorical data. It is defined as the percentage of false values using an approach according to the ground truth.

- *MNAD*: This metric is used for performance evaluation on numerical data. *MAD (Mean Absolute Deviation)* is a quantity on how close truth estimates are to the ground truth. As numerical data may have different scales, we normalize *MAD* using the standard variance of each data type. *MNAD* can be formulated as:

$$MNAD = \frac{1}{M}\sum_{v=1}^{M}\frac{|t_v - truth_v|}{\text{std}(c_{1,v}, c_{2,v}\cdots, c_{N,v})},$$

  where $truth_v$ stands for the ground truth for entity v, and the other notations are listed in Table 2.1.

2.4.2 Experiments on Real-World Data

Datasets: In order to evaluate the performance of the proposed model, we use two real-world datasets for experiments:

- *Flight Dataset*: This dataset [37] consists of 37 sources, which is collected from multiple websites. There are six different attributes in this dataset including: *scheduled departure/arrival time*, *actual departure/arrival time*, and *actual departure/arrival gate*. The former four attributes are numerical, while the last two attributes are categorical.

- *Stock Dataset*: This dataset [37] consists of 55 sources, which is collected from web search results. Specifically, *Volume*, *Shares outstanding*, and *Market cap* are

Table 2.2: Statistics of the Real-world Datasets

	Flight Dataset	Stock Dataset
# of Claims	2,790,734	11,748,734
# of Entities	204,422	326,423
# of Truths	16,572	29,198

treated as numerical data, while the other attributes are treated as categorical. The statistics of these datasets are shown in Table 2.2.

The task for our experiment is to estimate the true value for each entity in these two datasets.

In our experiment, for entities with categorical claims, we set $\alpha_e = 4.5, \beta_e = 20, \mu_b = 0, \sigma_b^2 = 1$ and $g^{(c)} = 3$ for the Flight dataset, and $\alpha_e = 10, \beta_e = 10, \mu_b = 0, \sigma_b^2 = 10$ and $g^{(c)} = 10$ for the Stock dataset. For entities with numerical claims, we set $\alpha_e = 4.5, \beta_e = 100, \mu_b = 0, \sigma_b^2 = 1$ and $g^{(n)} = 0$ for the Flight dataset, and $\alpha_e = 0.05, \beta_e = 0.10, \mu_b = 0, \sigma_b^2 = 10$ and $g^{(n)} = 0$ for the Stock dataset. μ_t and σ_t^2 are parameters related to the conditions of different datasets. In the preprocessing step, we shift the mean values of numerical data to 0. Hence, μ_t is set to be 0 for both datasets. We set $\sigma_t^2 = 1$ for the Flight dataset and $\sigma_t^2 = 100$ for the Stock dataset.

Correlation Extraction: The Flight dataset and Stock dataset do not have explicit correlations. However, investigations in previous work indicate that there are implicit correlations among sources [37]. Therefore, we adopt a correlation extraction method based on source similarities. In this method, if two sources make many similar false claims, they are regarded to be correlated, which is one of the most important intuitions for correlation extraction. We extract the source correlations according to the claim history and calculate the Jaccard distance between each pair of sources. The similarity is defined as:

$$sim(i,j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|} = \frac{|W_i \cap W_j|}{|W_i| + |W_j| - |W_i \cap W_j|},$$

Table 2.3: Performance on Real-World Datasets

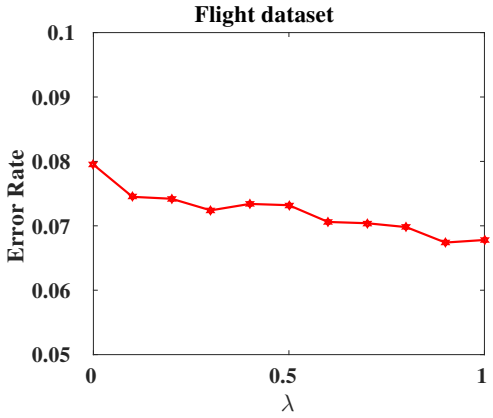
	Flight Dataset		Stock Dataset	
Method	Error Rate	MNAD	Error Rate	MNAD
IATD	**0.0674**	**2.7160**	**0.0689**	2.6734
IATD-ni	0.0795	2.7179	0.0789	2.6734
CRH	0.0823	4.8613	0.0700	**2.6445**
GTM	N/A	7.6703	N/A	2.8081
Voting	0.0859	N/A	0.0817	N/A
Invest	0.0919	6.4153	0.0983	2.8081
Pooled Invest	0.0925	5.8562	0.0990	2.7940
2-Estimate	0.0885	7.4347	0.0726	2.8509
3-Estimate	0.0881	7.1983	0.0818	2.7749
TruthFinder	0.0950	8.1351	0.1194	2.7140
AccuSim	0.0881	7.3204	0.0726	2.8503

where W_s is the set of entities on which source s makes wrong claims based on *Voting*. Then, we can use a threshold to determine the existence of correlations between two sources. The threshold is set to be 0.2 for the Flight dataset 0.08 for the Stock dataset. That is to say, on the Flight dataset, if $sim(i, j) > 0.2$, these sources are regarded as correlated. Otherwise, they are treated as independent. On the Stock dataset, if $sim(i, j) > 0.08$, these sources are regarded as correlated. Otherwise, they are treated as independent.

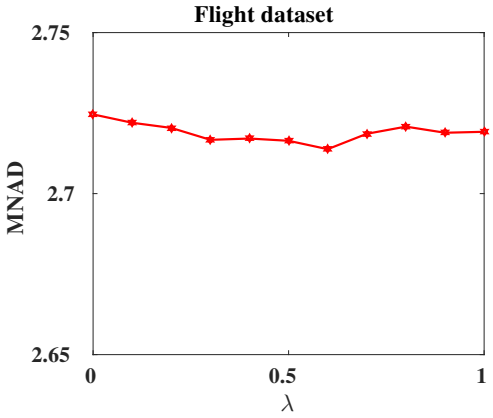
Overall Performance: The results of all the methods in terms of *Error Rate* and *MNAD* are shown in Table 2.3. From the table, we can see that the proposed IATD generally outperforms the baseline methods.

The reason that the proposed IATD approach can work comparable or better than other truth discovery approaches is due to the fact that influences are more precisely modeled. If we visualize the Jaccard distances between sources (Fig. 2.3), we can find that many sources are correlated. Therefore, the chance that these correlated sources influences each other when making claims is high. In contrast to baseline methods, the proposed IATD model takes these inter-source influences into consideration, which

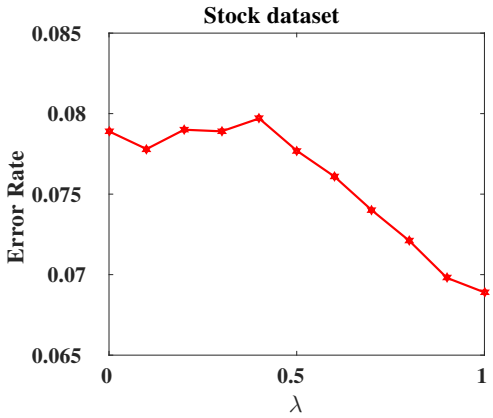
enables us to get more precise and interpretable estimates of source trustworthiness. Thus we can achieve a better performance in truth estimation.



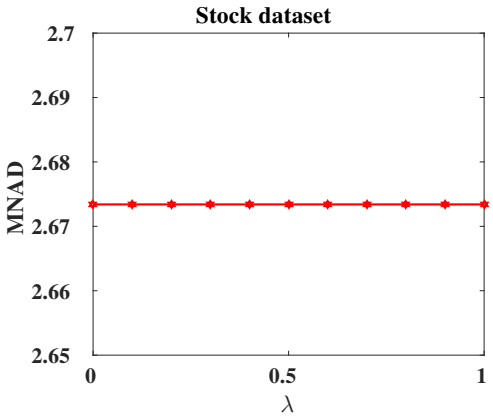
(a) Error Rate for the Flight dataset



(b) MNAD for the Flight dataset



(c) Error Rate for the Stock dataset



(d) MNAD for the Stock dataset

Figure 2.2: Performance w.r.t. influence ratio λ on the two real-world datasets.

Impact of Global Influence Factor λ: Global influence factor λ is used to adjust how much a source gets influenced by its related sources when it provides a specific claim. In order to better demonstrate the effect of λ, we show the variation of *Error rate* and *MNAD* with different values of influence parameter λ in Fig. 2.2. From Fig. 2.2, we can observe that the proposed IATD model performs differently with respect to the values of λ on the two datasets. The different trends imply that the datasets may have different correlation patterns (which is also suggested by Fig. 2.3). Based on the results

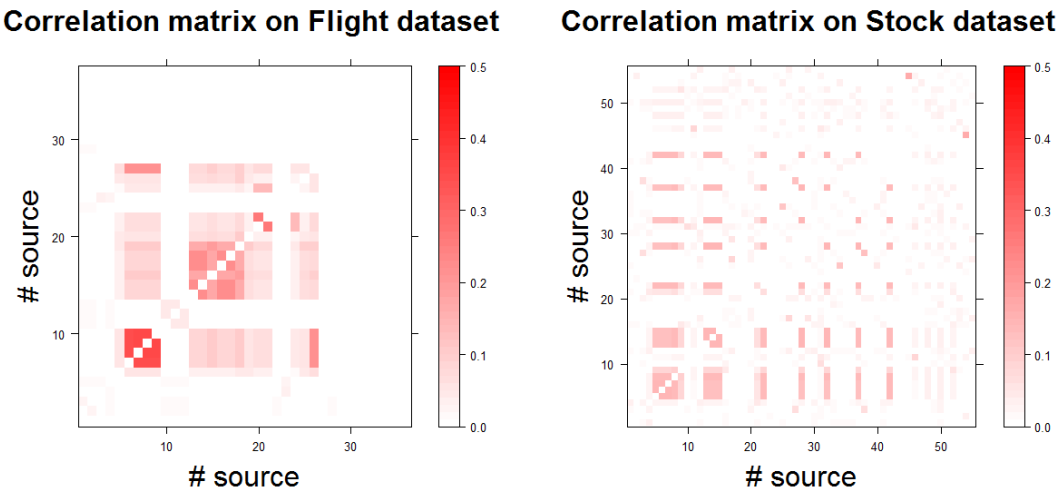


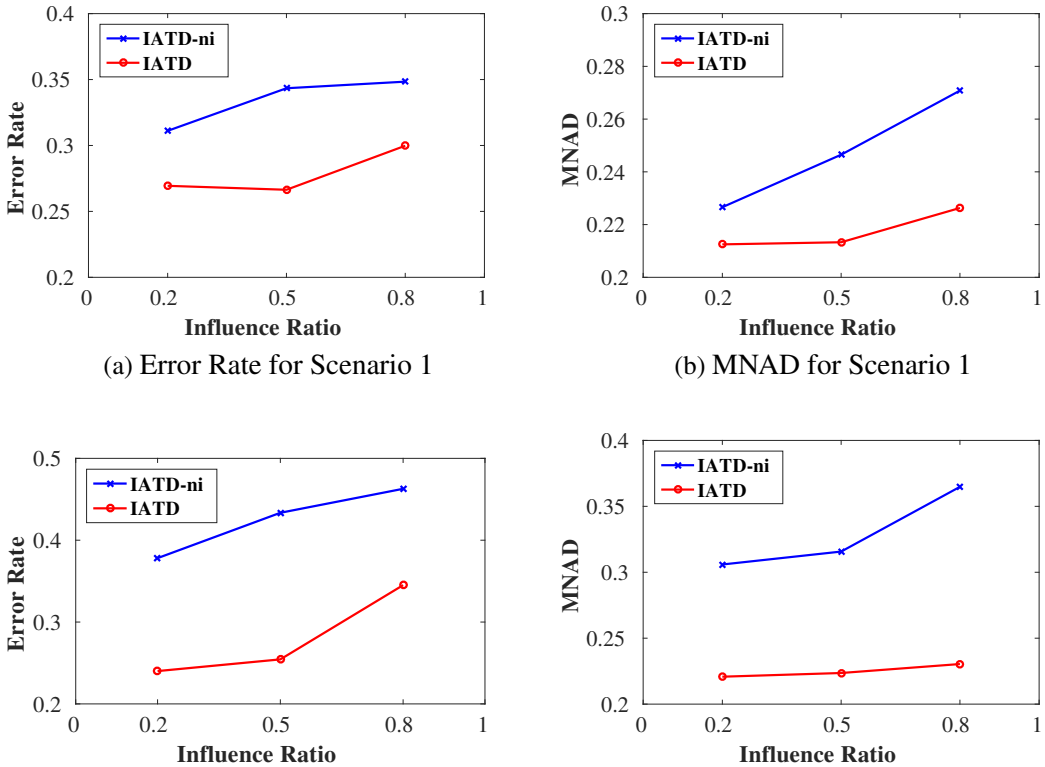
Figure 2.3: Visualization of source correlations on the Flight and Stock dataset.

from Fig. 2.2, a relatively large λ (close to 1) generally gives better results. This indicates that if two sources are correlated, they may influence each other strongly. These experiments illustrate that to enhance the performance of truth discovery, inter-source influences need to be utilized properly.

2.4.3 Experiments on Synthetic Data

In order to demonstrate the advantages of our proposed model comprehensively, we conduct experiments on simulated dataset.

Simulation Settings: Each synthetic dataset contains 10000 entities, where 7000 entities are numerical and 3000 entities are categorical. Different levels of noise are added to the ground truth to simulate sources with different levels of trustworthiness. In this section, we set the trustworthiness of a source to be consistent when it generates numerical and categorical claims. Specifically, we generate 10 high-quality sources ($\sigma^2 = 1$) and 10 low-quality sources ($\sigma^2 = 10$). For a specific source s, when simulating numerical claims, we sample the noise from a Gaussian distribution. The mean of this Gaussian distribution is zero and the deviation is set based on the level of source trustworthiness. When simulating categorical claims, we first sample a factor z from



(a) Error Rate for Scenario 1

(b) MNAD for Scenario 1

(c) Error Rate for Scenario 2

(d) MNAD for Scenario 2

Figure 2.4: Comparison of performance on the two scenarios.

the same Gaussian distribution, and then compare it with a confidence threshold δ. If $|z| \geq \delta$, we assign a random false choice to the claim; otherwise, the correct choice is assigned to the claim.

Two scenarios are considered in this experiment. For both scenarios, we test the proposed method with three levels of dependency. That is, we randomly allocate [20%, 50%, 80%] of all the sources as independent sources, with others as influenced sources. For the influenced sources, we consider two scenarios with different ratios of influenced claims. For *Scenario 1*, the influenced sources provide 20% of their claims independently. For *Scenario 2*, the influenced sources provide 80% of their claims independently. Such settings are based on the *Pareto Law*[2]. The task for this section is to estimate the true value for each entity in the dataset.

[2]For many events, roughly 80% of the effects come from 20% of the causes.

Results and Discussions: We choose IATD-ni as the baseline method for this part to demonstrate the effectiveness of utilizing inter-source influences in truth discovery. The results on the synthetic datasets are shown in Fig. 2.4. From the figures we can see that IATD method works consistently better than IATD-ni regardless the ratios of influences among sources and claims. Comparing the two scenarios, the results provided by IATD method are similar, while IATD-ni suffers a bigger performance degradation when there are more influenced claims. This again proves the importance of utilizing influences in truth discovery tasks, and demonstrates that the proposed method successfully models the source correlations and influences.

2.5 Summary

As an emerging topic, truth discovery has shown a great potential in a wide range of applications thanks to its ability to estimate the truths and source trustworthiness simultaneously. Many existing truth discovery methods assume that sources make claims independently, which may be violated in real world, as source correlations are ubiquitous. For those methods who do consider source correlations, they limit the claims to be categorical type. To better fit the real world applications, in this chapter, we propose a probabilistic model that can handle both challenges. By taking the source correlations as prior knowledge for influence derivation, the proposed influence-aware truth discovery model can estimate the trustworthiness of a source more accurately. Moreover, claims of both numerical and categorical types are modeled in a unified manner. Experimental results on two real world datasets prove the effectiveness of the proposed IATD model. Furthermore, experimental results on the simulated datasets illustrate the nice properties of the proposed IATD model under different scenarios.

Chapter 3

Truth Discovery for Unstructured Text Data

3.1 Introduction

In the big data era, tremendous data can be accessed on various online platforms, such as *Amazon Mechanical Turk*, *Stack Exchange* and *Yahoo Answers*. However, such multi-sourced data are usually contributed by non-expert online users, thus there may exist errors or even conflicts in the data. Therefore, how to automatically infer trustworthy information (i.e., the truths) from such noisy and conflicting data is a challenging problem.

To address this challenge, truth discovery methods have been proposed [16, 18, 21, 35–38, 42, 46, 47, 61, 62, 68, 80, 81, 89], which aim to estimate trustworthy information from conflicting data by considering user reliability degrees. Truth discovery approaches follow two fundamental principles: (1) If a user provides much trustworthy information or true answers, his/her reliability is high; (2) If an answer is supported by many reliable users, this answer is more likely to be true. Though yielding reasonably good performance, most existing truth discovery methods are designed for structured data, and are difficult to be directly applied to *text data*, which are unstructured and

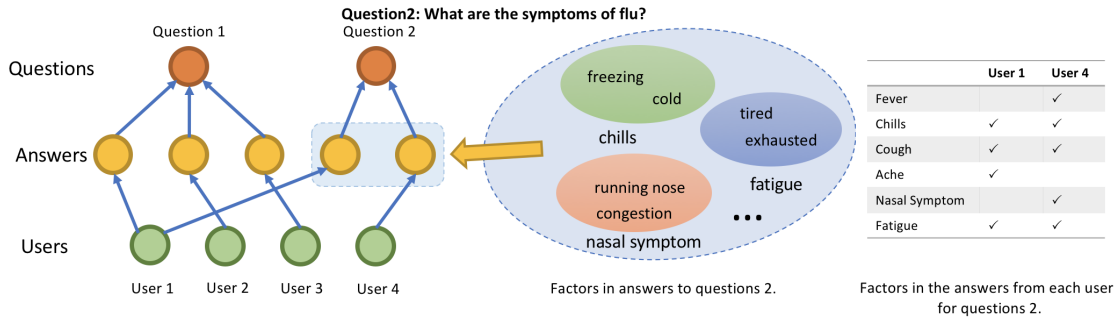


Figure 3.1: An Illustration of questions, answers, answer factors and keywords.

noisy. This significantly narrows the application domain of these truth discovery methods, as a large ratio of the multi-sourced data are text. Actually, there are several unique characteristics of natural language that hinder the existing truth discovery methods from being successfully applied to text data.

Figure 3.1 gives an illustration of these two characteristics of text data. First, the answer to a factoid question [1] may be *multifactorial*, and it is usually hard for a given text answer to cover all the factors. For the question '*What are the symptoms of flu?*', the correct answer should contain the following factors: *fever*, *chills*, *cough*, *nasal symptom*, *ache*, and *fatigue*. Even if the answer provided by a user covers two factors, such as *cough* and *chills*, the existing truth discovery methods may determine this answer to be totally wrong and assign a low reliability degree to this user. This is because these methods treat the whole answer as an integrated unit. However, if we take the fine-grained answer factors into consideration, the answer provided by this user is partially correct, which implies that we should give some credits to the user by increasing his/her reliability degree. Thus, how to identify partially correct answers and model factors of text answers is critical for the task of truth discovery on text data.

The second characteristic of text data is the *diversity of word usages*. Answers provided by online users may convey a very similar meaning with different keywords. For example, users may use words such as *tired* or *exhausted* to describe the symptom of

[1] **Note**: This chapter merely focuses on finding trustworthy answers for **factoid questions**. Factoid questions are defined as questions that can be answered with **simple facts** expressed in **short text answers**. [27]

fatigue. However, existing truth discovery approaches may treat them as totally different answers. Thus, it is of great importance to model the diversity among answers in the text data when inferring trustworthy information.

In order to tackle the aforementioned challenges for inferring trustworthy information from text data, in this chapter, we propose a model named "*TextTruth*", which takes the keywords in each answer as inputs and outputs a ranking for the answer candidates based on their trustworthiness. Specifically, we first transform the keywords in text answers into pre-trained computable vector representations. Due to the fact that an answer may contain multiple factors, the "answer-level" or coarse-grained representations may not be able to capture the partially correct answers. Thus, we need to convert the whole answer into fine-grained factors. Then, we model the diversity of answers by clustering the keywords with similar semantic meanings. By doing so, we can estimate the trustworthiness of each answer factor instead of the whole answer and infer the correctness of each factor in the answer.

3.2 Problem Definition

In this chapter, we consider a general truth discovery scenario for factoid text questions and answers. Before introducing the problem formulation, we first define some basic terminologies that will be used in the rest of the chapter:

Definition 1 (Question)**.** *A* question q *contains* N_q *words and can be answered by users.*

Definition 2 (Answer)**.** *An* answer *given by user* u *to question* q *is denoted as* a_{qu}*.*

Definition 3 (Answer Keyword)**.** Answer keywords *are domain-specific content words / phrases in answers. The* m*-th answer keyword of the answer given by user* u *to question* q *is denoted as* x_{qum}*.*

Definition 4 (Answer Factor)**.** Answer factors *are the key points of the answers, which are represented as clusters of* answer keywords*. The* k*-th answer factor in the answers to question* q *is denoted as* c_{qk}*.*

For each question, there can be different answers provided by different users. These answers may consist of complex sentences with multiple factors and can be partially correct. This setting can support a broad range of text data. Formally, the problem discussed in this chapter can be defined as:

Definition 5 (Problem Definition)**.** *Given a set of* users $\{u\}_1^U$*, a set of questions* $\{q\}_1^Q$ *and a set of answers* $\{a_{qu}\}_{q,u=1,1}^{Q,U}$*, where* U *denotes the number of users and* Q *stands for the number of questions. The goal of this chapter is to extract highly-trustworthy answers and highly-trustworthy key factors in answers for each question.*

3.3 Methodology

In this section, we first offer an overview of the proposed TextTruth model, and then explain in detail each component of it.

3.3.1 Overview

When applying truth discovery methods to find the trustworthy answers to complex natural language questions, semantic correlations among answers should be taken into consideration, so that user reliability can be accurately estimated. However, learning accurate vector representations for the whole answers is difficult especially when the context corpus of these answer paragraphs is not sufficiently large. Moreover, due to the complexity of natural language, the meaning of an answer is too complicated to be represented by a single vector. To tackle such challenges, we rely on more fine-grained semantic units (i.e., answer factors) in each answer to determine the trustworthiness of each answer.

In this chapter, for each question, we first extract the *keywords* in each answer and learn their vector representations. Then we cluster these word/phrase-level keywords into semantic clusters (i.e., factors). These factors represent all the possible key points in the answers to a question and can be used to determine the trustworthiness of an answer.

For the keywords within each cluster, as they share very similar semantic meanings, their trustworthiness should be almost the same. In addition, users may have different reliabilities, which can be reflected in the answers they provided.

Based on the above ideas, we propose a two-step method to estimate the trustworthiness of each answer. In the first step, we specify a probabilistic model to model the generation of keywords with user reliabilities taken into consideration in Section 3.3.2. The generative model, which consists of three major components, jointly learns the answer factors and their truth label. The generative model first generates a mixture of answer factors and their semantic parameters. After that, the model generates two-fold user reliability variables, which model the comprehensiveness and accuracy of answer factors provided by a specific user. These two variables capture a whole spectrum of the user reliability. Finally, the model selects an answer factor based on the semantics, the trustworthiness of the answer factor as well as the reliability of the user that provides the answer, and generate the keyword embedding vector via a von Mises-Fisher (vMF) distribution. The vMF distribution is centralized at the semantic centroid of that answer factor. This way, the design of answer factor and user reliability takes the multifactorial characteristics of answers into consideration. Meanwhile the keyword embedding vector generation also captures the diversity of word usages. These designs make the model capable of capturing the unique characteristics of text data. In section 3.3.3, we design a straightforward scoring mechanism to evaluate the trustworthiness score of each answer. We provide the parameter estimation of the proposed method in Section 3.3.4.

3.3.2 Generative Model

We develop a probabilistic model to jointly learn the answer factors and the truth labels of each answer factor for every question. For an answer a_{qu}, we extract domain-specific answer keywords and get their normalized [2] vector representations [52]. The set of all the vector representations is denoted as $\{\mathbf{v}_{qum}\}$, which also serves as the observation of

[2]The normalized vector of $\mathbf{v}$ is given by $\hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}$, where $|\mathbf{v}|$ is the $l2$-norm of $\mathbf{v}$.

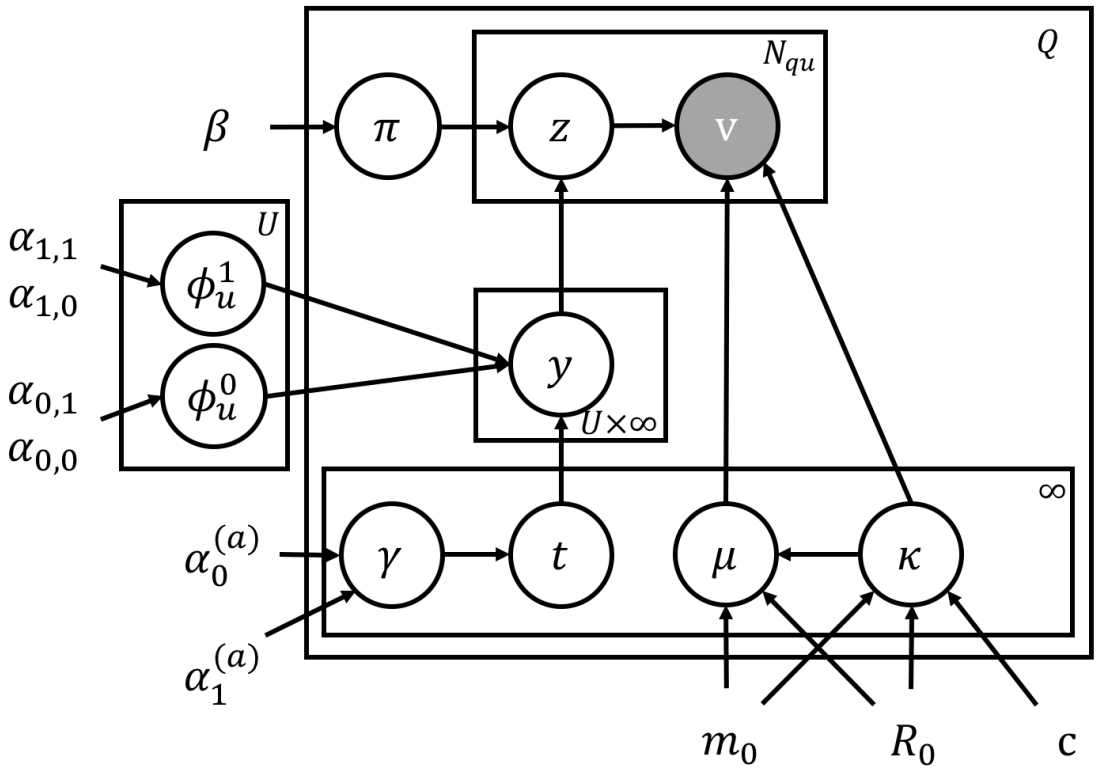


Figure 3.2: Plate notation for the proposed TextTruth Model.

the probabilistic model. Figure 3.2 shows the plate notation of the proposed model. The generative model consists of three major components, which are listed as follows:

I. Answer Factor Modeling: The model first generate the mixture of factors according to the Dirichlet distribution, which is commonly used to generate mixture models. Formally, the mixture distribution $\boldsymbol{\pi}_q$ is generated as:

$$\boldsymbol{\pi}_q \sim \text{Dirichlet}(\boldsymbol{\beta}). \tag{3.1}$$

Here, $\boldsymbol{\beta}$ is a K_q-dimensional vector, where K_q denotes the number of factors in the answers to question q.

For the k-th answer factor under question q, we model its trustworthiness via a binary truth label t_{qk}. Specifically, the model first generates the prior truth probability γ_{qk}. It

determines the prior distribution of how likely each factor is to be true, from a Beta distribution with hyper-parameter $\alpha_1^{(a)}$ and $\alpha_0^{(a)}$:

$$\gamma_{qk} \sim \text{Beta}(\alpha_1^{(a)}, \alpha_0^{(a)}). \tag{3.2}$$

Then the truth label t_{qk} is generated from a Bernoulli distribution with parameter γ_{qk}:

$$t_{qk} \sim \text{Bernoulli}(\gamma_{qk}). \tag{3.3}$$

Finally, to model the semantic characteristic of each answer factor, we define the centroid parameter $\boldsymbol{\mu}_{qk}$ and concentrate parameter κ_{qk} of vMF distributions from its conjugate prior distribution $\Phi(\boldsymbol{\mu}_{qk},\ \kappa_{qk}; \boldsymbol{m}_0, R_0, c)$ [58], i.e.:

$$\boldsymbol{\mu}_{qk},\ \kappa_{qk} \sim \Phi(\boldsymbol{\mu}_{qk},\ \kappa_{qk}; \boldsymbol{m}_0, R_0, c), \tag{3.4}$$

where $\Phi(\boldsymbol{\mu}_{qk},\ \kappa_{qk}; \boldsymbol{m}_0, R_0, c)$ is defined as:

$$\Phi(\boldsymbol{\mu}_{qk},\ \kappa_{qk}; \boldsymbol{m}_0, R_0, c) \propto \{C_D(\kappa_{qk})\}^c \exp(\kappa_{qk} R_0 \boldsymbol{m}_0^T \boldsymbol{\mu}_{qk}).$$

Here, $C_D(\kappa) = \frac{\kappa^{D/2-1}}{I_{D/2-1}(\kappa)}$, and $I_{D/2-1}(\cdot)$ is the modified *Bessel function* of the first kind. In practice, there may be few answers that are totally irrelevant to the question. Since the answer factors in irrelevant answers are usually supported by very few users, they will not be regarded as trustworthy.

II. User Reliability Modeling: The reliability of each user is inferred according to the answers they provide. As aforementioned, the answer of a user u may merely cover part of the trustworthy answer factors, and at the same time may consist of untrustworthy answer factors. For instance, some users may only provide the factors that they are very confident of. On the contrary, other users may cover a broad collection of answer factors with different trustworthinesses in their answers. This naturally motivates us to use a two-fold score like [92] to model the reliability of a user.

Suppose we know all the answer factors and their truth labels in advance, for all the questions and their answers, we use TP_u and FP_u to denote the number of trustworthy and untrustworthy answer factors that are covered by the answers from user u (i.e., the number of true positive and false positive factors), respectively. Similarly, we use FN_u and TN_u to denote the number of trustworthy and untrustworthy answer factors that are not covered by the answers from user u (i.e., the number of false negative and true negative factors), respectively. Based on these statistics, we can intuitively use the false positive rate (defined as: $\frac{FP_u}{FP_u+TN_u}$), and the true positive rate (defined as: $\frac{TP_u}{TP_u+FN_u}$) to fully characterize u's reliability.

Let's resume the discussion of the proposed model. During the generative process, the answer factors and their truth labels are not known in advance. Inspired by [92], we also define two-fold user reliability variables ϕ_u^0 and ϕ_u^1 to model the false positive rate and the true positive rate of factors that are covered by the answers of user u. Specifically, for each user u, we generate ϕ_u^0 and ϕ_u^1 from two Beta distributions with hyper-parameters $(\alpha_{0,1}, \alpha_{0,0})$ and $(\alpha_{1,1}, \alpha_{1,0})$, respectively. Here, $\alpha_{0,1}$ and $\alpha_{0,0}$ are the prior false positive count and true negative count, respectively. Similarly, $\alpha_{1,1}$ and $\alpha_{1,0}$ stand for the prior true positive count and the false negative count of each source, respectively. Formally:

$$\begin{aligned} \phi_u^0 &\sim \mathrm{Beta}(\alpha_{0,1}, \alpha_{0,0}) \qquad \text{(False Positive Rate)} \\ \phi_u^1 &\sim \mathrm{Beta}(\alpha_{1,1}, \alpha_{1,0}) \qquad \text{(True Positive Rate)}. \end{aligned} \tag{3.5}$$

III. Observation Modeling: As aforementioned, we use the vector representations of keywords as observations. For the m-th word representation from user u for question q, we specify the following generation process.

Firstly, we define a binary indicator $y_{u,qk}$, which denotes whether the k-th factor of the answers to question q should be covered by user u, based on the reliability of u. For question q, if its truth label $t_{qk} = 1$, the probability of user u covering the k-th factor in its answer follows a Bernoulli distribution with reliability parameter ϕ_u^1. Otherwise,

if its truth label $t_{qk} = 0$, the probability follows a Bernoulli distribution with reliability parameter ϕ_u^0. Formally, this process can be written as:

$$\begin{aligned} y_{u,qk} &\sim \text{Bernoulli}(\phi_u^0) \qquad \text{If } t_{qk} = 0, \\ y_{u,qk} &\sim \text{Bernoulli}(\phi_u^1) \qquad \text{If } t_{qk} = 1. \end{aligned} \tag{3.6}$$

To this point, we have determined the set of answer factors that should be covered by the answer a_{qu}, with the reliability of u taken into consideration.

Then, for the m-th keyword in the answer a_{qu}, its factor label z_{qum} is drawn from a probability density function defined as:

$$P(z_{qum} = k|\boldsymbol{\pi}_q, y_{u,qk}) \propto \begin{cases} \pi_{qk} & \text{if } y_{u,qk} = 1, \\ 0 & \text{if } y_{u,qk} = 0. \end{cases} \tag{3.7}$$

The density function jointly considers the answer factor mixture distribution and the set of binary indicators $y_{u,q\cdot}$. This means that both semantics and user reliabilities are used to determine the factor label of a specific answer keyword.

With the factor labels determined, the model samples keywords vectors that describe the semantic meaning of its corresponding factor. Note that this procedure should not involve the reliability of a user. The vector representation of a keyword (i.e. $\mathbf{v}_{qum}$) is randomly sampled from a vMF distribution with parameter $\boldsymbol{\mu}_{qk}, \kappa_{qk}$:

$$\mathbf{v}_{qum} \sim \text{vMF}(\boldsymbol{\mu}_{qk}, \kappa_{qk}). \tag{3.8}$$

Specifically, for a D-dimensional unit semantic vector $\mathbf{v}$ that follows vMF distribution, its probability density function is given by:

$$p(\mathbf{v}_{qum}|\boldsymbol{\mu}_{qk}, \kappa_{qk}) = C_D(\kappa_{qk}) \exp(\kappa_{qk} \boldsymbol{\mu}_{qk}^T \mathbf{v}_{qum}). \tag{3.9}$$

Algorithm 1: Generative Process of TextTruth

for *each question* q **do**
  Draw mixture $\pi_q \sim \text{Dirichlet}(\boldsymbol{\beta})$;
  for *each answer factor* k **do**
    Draw centroid and concentration: $\boldsymbol{\mu}_{qk},\ \kappa_{qk} \sim \Phi(\boldsymbol{m}_0, R_0, c)$;
    Draw truth parameter: $\gamma_{qk} \sim \text{Beta}(\alpha_0^{(a)}, \alpha_1^{(a)})$;
    Draw a truth label: $t_{qk} \sim \text{Bernoulli}(\gamma_{qk})$;
  end
end
for *each user* u **do**
  Draw: $\phi_u^0 \sim \text{Beta}(\alpha_{0,1}, \alpha_{0,0}), \quad \phi_u^1 \sim \text{Beta}(\alpha_{1,1}, \alpha_{1,0})$;
end
for *each answer* a_{qu} **do**
  for *each answer factor* k **do**
    Draw binary label: $y_{u,qk} \sim \text{Bernoulli}(\phi_u^{t_{qk}})$;
  end
  for *each keyword* m **do**
    Draw a answer factor label: $P(z_{qum} = k|\boldsymbol{\pi}, y_{u,qk})$;
    Draw keyword embedding: $\mathbf{v}_{qum} \sim \text{vMF}(\boldsymbol{\mu}_{qz_{qum}}, \kappa_{qz_{qum}})$;
  end
end

The vMF distribution has two parameters: the mean direction $\boldsymbol{\mu}_{qk}$ and the concentration parameter $\kappa_{qk}(\kappa_{qk} > 0)$. The distribution of $\mathbf{v}_{qum}$ on the unit sphere concentrates around the mean direction $\boldsymbol{\mu}_{qk}$, and is more concentrated if κ_{qk} is larger. In our scenario, the mean vector $\boldsymbol{\mu}$ acts as a semantic focus on the unit sphere, and produces relevant semantic embeddings around it. The superiority of the vMF distribution over other continuous distributions (e.g., Gaussian) for modeling textual embeddings has also been shown in the field of clustering [3] and topic modeling [22].

The overall generative process is summarized in Algorithm 1.

3.3.3 Trustworthy-Aware Answer Scoring

Intuitively, the trustworthiness of an answer should be evaluated by the volume of correct information it provides. Hence, we propose a straightforward scoring mechanism to

evaluate the trustworthiness score of each answer. Given the inferred truth labels for each answer factor of question q, we score the answers according to the number of answer keywords in the answer a_{qu} that are related to the factor with truth label $t_{qk} = 1$, i.e.:

$$score_{qu} = \sum_{k=1}^{K_q} N_{u,qk} \mathbb{I}(t_{qk} = 1), \tag{3.10}$$

where K_q is the number of answer factors for question q, $N_{u,qk}$ denotes the number of keywords that are provided by user u and are clustered into factor k. $\mathbb{I}(t_{qk} = 1) = 1$ if $t_{qk} = 1$, and $\mathbb{I}(t_{qk} = 1) = 0$ if $t_{qk} = 0$. Note that there are many alternative ways of designing scoring functions.

3.3.4 Model Fitting

In this section, we present the approach to estimating the latent variables and the user reliability parameters.

Latent Variable Estimation: We use MCMC method to infer the latent variables t, z, y and κ. As one can see, the values of y and z have a large impact on the final results, and they may be sensitive to the initialization. Therefore, we make an approximation in latent variable estimation to make the process stable. The detailed steps are specified in the following paragraphs.

First, using conjugate distributions, we are able to analytically integrate out the model parameters and only sample the cluster assignment variable z. This is done as follows:

$$\begin{aligned} & P(z_{qum} = k \mid \boldsymbol{z}_{q,\neg um}, \boldsymbol{\beta}, \boldsymbol{m}_0, R_0, c) \\ \propto\ & P(z_{qum} = k \mid \boldsymbol{z}_{q,\neg um}, \boldsymbol{\beta}) \\ & \times P(\mathbf{v}_{qum} | \mathbf{v}_{q,\neg um}, z_{qum} = k, \boldsymbol{z}_{q,\neg um}, \boldsymbol{m}_0, R_0, c), \end{aligned} \tag{3.11}$$

where $\mathbf{v}_{q,\neg um}$ stands for the set of all the keywords in the answers for question q, except the m-th keyword from user u.

Then we can derive the expressions for the two terms in Eq. (3.11). The first term $P(z_{qum} = k \mid \boldsymbol{z}_{q,\neg um}, \boldsymbol{\beta})$ can be written as:

$$P(z_{qum} = k|\boldsymbol{z}_{q,\neg um}, \boldsymbol{\beta}) \propto N_{qk\neg um} + \beta_k \tag{3.12}$$

where $N_{qk,\neg um}$ denotes the number of answer keywords under the k-th factor of question q except current keyword $\mathbf{v}_{qum}$. The second term in Eq. (3.11) is similar to the form of *vMF Mixture Model*, which can be written as:

$$\begin{aligned} &P(\mathbf{v}_{qum}|\mathbf{v}_{q,\neg um}, z_{qum} = k, \boldsymbol{z}_{q,\neg um}, \boldsymbol{m}_0, R_0, c) \\ &\propto \frac{C_D(\kappa_{qk})C_D(||\kappa_{qk}(R_0\boldsymbol{m}_0 + \mathbf{v}_{qk\neg um})||_2)}{C_D(||\kappa_{qk}(R_0\boldsymbol{m}_0 + \mathbf{v}_{qk})||_2)}, \end{aligned} \tag{3.13}$$

where $\mathbf{v}_{qk}$ denotes the sum of all the vector representations of keywords in factor k for question q. The concentration parameters κ_{qk} are sampled from the following distribution:

$$P(\kappa_{qk}|\boldsymbol{\kappa}_{q\neg k}, \boldsymbol{m}_0, R_0, c) \propto \frac{(C_D(\kappa_{qk}))^{c+N_{qk}}}{C_D(\kappa_{qk}||R_0\boldsymbol{m}_0 + \mathbf{v}_{qk}||_2)}. \tag{3.14}$$

The conditional distribution of κ_{qk} is again not of a standard form, we use a step of Metropolis Hasting sampling (with log-normal proposal distribution) to sample κ_{qk}. To this point, we get the full expression of Eq. (3.11). In the circumstance when the model fitting efficiency becomes a concern, the sampling process specified by Eq. (3.11) can be approximated via the method specified in [72], which also produces satisfactory results.

Here, we make an approximation by removing the impact of y in terms of determining the value z. For the answer provided by user u for questions q, $y_{u,qk}$ is determined via:

$$y_{u,qk} = \begin{cases} 0 & \text{If } \nexists\ m \text{ satisfies } z_{qum} = k \\ 1 & Otherwise. \end{cases} \tag{3.15}$$

Finally, we move on to sample the truth label for each answer factor under each question t_{qk} via the following posterior distribution:

$$\begin{aligned} & P(t_{qk} = x \mid \boldsymbol{t}_{q,\neg k}, \boldsymbol{z}_q, \boldsymbol{y}_q, \alpha_{0,0}, \alpha_{0,1}, \alpha_{1,0}, \alpha_{1,1}, \alpha_0^{(a)}, \alpha_1^{(a)}) \\ \propto\ & \alpha_x^{(a)} \prod_{u \in U_q} \frac{\alpha_{x,y_{u,qk}} + n_{u,x,y_{u,qk}}}{\alpha_{x,0} + \alpha_{x,1} + n_{u,x,0} + n_{u,x,1}}, \end{aligned} \tag{3.16}$$

where U_q is the set of users that provide answer for question q. Here, $x \in \{0, 1\}$. $n_{u,0,0}$, $n_{u,0,1}$, $n_{u,1,0}$ and $n_{u,1,1}$ denote the number of true negative, false positive, false negative and true positive factors provided by user u, respectively.

User Reliability Estimation: With t, y, κ and z determined, we are able to obtain the closed-form solution for ϕ_u^0 and ϕ_u^1 by setting the partial derivatives of the negative log-likelihood respective to ϕ_u^0 and ϕ_u^1 to zero:

$$\phi_u^0 = \frac{\alpha_{0,1} + n_{u,0,1}}{\alpha_{0,0} + \alpha_{0,1} + n_{u,0,1} + n_{u,0,0}}, \tag{3.17}$$

$$\phi_u^1 = \frac{\alpha_{1,1} + n_{u,1,1}}{\alpha_{1,0} + \alpha_{1,1} + n_{u,1,0} + n_{u,1,1}}, \tag{3.18}$$

where $n_{u,0,0}$, $n_{u,0,1}$, $n_{u,1,0}$ and $n_{u,1,1}$ are user reliability statistics, which denote the number of true negative, false positive, false negative and true positive factors provided by user u, respectively. Moreover, these statistics also allow us to calculate other user reliability metrics, e.g., precision score of a user:

$$prec_u = \frac{\alpha_{1,1} + n_{u,1,1}}{\alpha_{0,1} + \alpha_{1,1} + n_{u,0,1} + n_{u,1,1}}. \tag{3.19}$$

This score is also used in the experiment section to validate the estimated user reliability.

3.4 Experiments

In this section, we empirically validate the performance of the proposed method from the following aspects: Firstly, we compare the performance of the proposed method with the state-of-the-art truth discovery methods as well as a couple of retrieval based schemes to demonstrate the advantage of utilizing fine-grained semantic units of answers for better answer trustworthiness estimation. After that, we provide a case study to show that the results produced by the proposed method are highly interpretable. Finally, we validate the estimated user reliabilities with groundtruth to further prove that the proposed method can make a good estimation of user reliabilities.

3.4.1 Datasets

SuperUser Dataset & ServerFault Dataset: These two datasets are collected from the community question answering (CQA) websites *SuperUser.com* and *ServerFault.com*, respectively. These two websites are mainly focused on the questions about general daily computer usages and server administration, respectively. The task on these datasets is to extract the most trustworthy answer to each question. We use the answers' votes from *SuperUser.com* and *ServerFault.com* as the groundtruths for evaluation.

Student Exam Dataset [55]: This dataset is collected from introductory computer science assignments with answers provided by a class of undergraduate students in the University of North Texas. 30 students submit answers to these assignments. For each assignment, the students' answers are collected via an online learning environment. The task on this dataset is to extract Top-K (K is set to 1-10 in this chapter) trustworthy student answers for each question. The groundtruth answers are given by the instructors. All the answers are independently graded by two human judges, using an integer scale from 0 (completely incorrect) to 5 (perfect answer). The statistics of these three datasets are shown in Table 3.1.

Table 3.1: Data Statistics.

Item	SuperUser	ServerFault	Student Exam
# of Questions	3379	7621	80
# of Users	1036	1920	30
# of Answers	16014	40373	2273

Pre-Processing: For all the datasets, we discard all code blocks, HTML tags, and stop words in the text. Answer keywords are extracted using entity dictionary and Stanford POS-Tagger[3]. To train word vector representations, we utilize all the crawled texts as the corpus. Skip-gram architecture in package gensim[4] is used to learn the vector representation of every answer keyword. The dimensionality of word vectors is set to 100, context window size is set to 5, and the minimum occurrence count is set to 20. For more details on the embedding algorithm, please refer to [52].

3.4.2 Experiment Protocols

3.4.2.1 Comparison Methods

We compare the proposed TextTruth model against several state-of-the-art truth discovery and retrieval-based answer selection approaches.

Bag-of-Word (BOW) Similarity: The bag-of-word vectors of questions and their answers are extracted. Answers are ranked according to the similarity values between the question vector and its corresponding answer vectors.

Topic Similarity: We utilize Latent Dirichlet Allocation (i.e. LDA [6]) to extract a 100-dimension topic representation for each question and its corresponding answers. Similar to BOW, answers are ranked according to the cosine similarity to the question.

CRH [36] + Topic Dist.: CRH is an optimization based truth discovery framework which can handle both categorical and continuous data. The goal of the optimization problem is to minimize the weighted loss of the aggregation results. In the experiment,

[3]https://nlp.stanford.edu/software/tagger.shtml
[4]https://pypi.python.org/pypi/gensim, an implementation of Word2Vec

we use the topic distributions as the representations of the whole answers to be fed to CRH.

CRH [36] + Word Vec.: This baseline approach is similar to *CRH + Topic Dist.* except that the inputs are changed to the average word vectors of answers. These word vector representations are learned as in [52].

CATD [35] + Topic Dist.: CATD is another optimization based truth discovery framework which considers the long-tail phenomena in the data. The optimization objective is similar to that of CRH. However, the upper bounds of user reliability are used for weight loss calculation. Similar to *CRH + Topic Dist.*, we use the topic distributions as the representations of the whole answers to be fed to CATD.

CATD [35] + Word Vec.: This baseline approach is similar to *CATD + Topic Dist.* except that the inputs are changed to the average word vectors of answers. The word vector representations are the same as those in *CRH + Word Vector*.

For each baseline approach, we implement it and set its parameters according to the method recommended by the original paper.

3.4.2.2 Evaluation Metrics

Due to the differences in dataset characteristics, evaluation metrics for three datasets are slightly different. On CQA datasets, we report the precisions of returned best answers from each method for each question. On student test dataset, we report the average score of returned top-K (K is set to 1-10 in this chapter) trustworthy answers from each method for each question.

3.4.3 Performance and Analysis

The results are shown in Figure 3.3 and Table 3.2. For student exam dataset, we only show the results on exam 1 3 data. The results on rest exams follow the same tendency. As one can see, the proposed method TextTruth consistently outperforms all the baseline methods. By outperforming various retrieval-based approaches and state-of-the-art

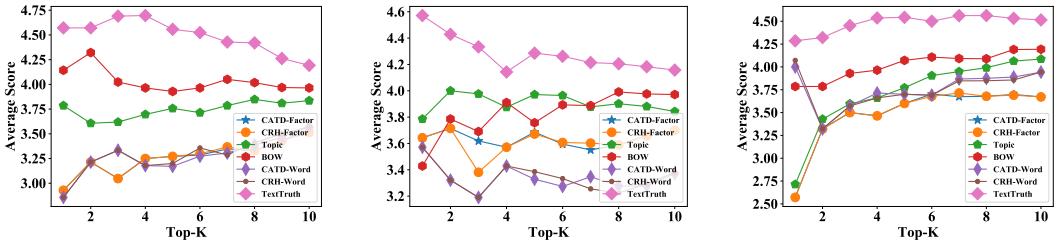
Table 3.2: Results on ServerFault Dataset & SuperUser Dataset.

Method	ServerFault	SuperUser
BOW Similarity	0.2077	0.1944
Topic Similarity	0.2462	0.2462
CATD + Topic Dist.	0.2311	0.2308
CATD + Word Vec.	0.1821	0.2234
CRH + Topic Dist.	0.2453	0.2453
CRH + Word Vec.	0.1847	0.2231
TextTruth	**0.3985**	**0.4019**

truth discover approaches, the proposed TextTruth demonstrates its great advantages on natural language data.

The reasons why the proposed TextTruth surpasses all the baseline methods are as follows. First, retrieval-based approaches (i.e., *BOW Similarity* and *Topic Similarity*) rank the answers merely based on the semantic similarity between the question and answers. However, a question itself does not necessarily cover all the semantics that should be covered in ideal answers. Therefore, retrieval-based methods only discover relevant answers instead of trustworthy answers. On the other hand, although existing truth discovery methods can capture user reliability for answer ranking, the performance is not very satisfactory. This is because these truth discovery approaches treat the answers as an integrated semantic unit, and ignore the fact that the semantic meaning of each answer may be complicated. Therefore, single vector representations fail to capture the innate correlations among these answers. To make things worse, CRH and CATD regard the weighted aggregation of these single vector representations as the "true" semantic representation to evaluate user reliabilities. However, answers from different users may involve distinct aspects of answers. Therefore, aggregating semantic representation of answers with distinct aspects only produces an inaccurate representation, which cannot be used to correctly estimate the reliabilities of users. The inaccurate user reliability estimation would further lead to incorrect aggregated results.

In contrast to existing approaches, the proposed TextTruth regards each answer as a collection of fine-grained semantic units (i.e., factors), which are represented by sepa-



(a) Exam 1 (b) Exam 2 (c) Exam 3

Figure 3.3: Performance on Exam Datasets.

Table 3.3: Case Study of Real Question and Answers.

	Content
Question	What is a tree?
Groundtruth Answer	A collection of nodes, which has a special node called root, and the rest of the nodes are partitioned into one or more disjoint sets, each set being a tree.
Top Answer 1	A tree is a finite set of one or more nodes with a specially designated node called the root and the remaining nodes are partitioned into disjoint sets where each of these sets is a tree.
Top Answer 2	A a finite collection of nodes, where it starts, with an element, called the root,, which has children, and its children have children until you get to the leaves which are the last elements and have to children
Untrustworthy Answer	It is a list of numbers in a list made by comparing values of nodes already in the tree and adding to the appropriate spot. Its a list made up of nodes with left and right points.

rated keyword vector representations. Based on these semantic units, TextTruth discovers the innate factors of each answer by grouping keywords into factors, and evaluates the trustworthiness of each answer on the top of these factors. As mentioned in the above paragraph, the major reason why existing truth discovery methods cannot produce satisfactory results is that these methods cannot aggregate the semantic representation of answers with distinct aspects effectively. Instead, the proposed TextTruth evaluate the users' reliabilities according to whether their answers contain keywords from the fac-

tors that are regarded to be correct (or incorrect). Therefore, the trustworthiness of each answer is better evaluated, which leads to the best result.

3.4.4 Case Study

To better evidence the analysis above, we give a case study on a question in the exam dataset. The question is related to the data structure. The result of the case study is shown in Table 3.3. In Table 3.3 words in blue color are keywords that are estimated to be trustworthy, while words in red color are keywords that are estimated to be untrustworthy or unrelated. The groundtruth answer is provided by the instructors.

As one can see, the proposed method can automatically select keywords that are meaningful to the questions, such as "node", "tree" and "root". Moreover, we can observe that the top-ranked answers have more true keywords than low-ranked untrustworthy answers. These phenomena again demonstrate that the results of the proposed model are both effective and interpretable. The case study also demonstrates why existing approaches fail to produce satisfactory results. First, the question itself merely consists of one keyword 'tree'. Therefore, retrieval-based methods, rank '*Untrustworthy Answer*' over '*Top Answer 2*', because it contains exactly the same keyword that exists in the question. This indicates that we cannot rely merely on relevance to find trustworthy answers. Second, we can see that the correct keywords involve multiple aspects (i.e., factors). These factors shape a comprehensive description of a tree. Such phenomenon is very common in natural language questions and answers, but cannot be successfully handled by the existing methods. That is why the proposed method can produce better results than the state-of-the-art truth discovery methods.

3.4.5 User Reliability Validation

The quantitative results and the case study shown above have demonstrated that the proposed method can outperform other baseline methods. In this section, we *further*

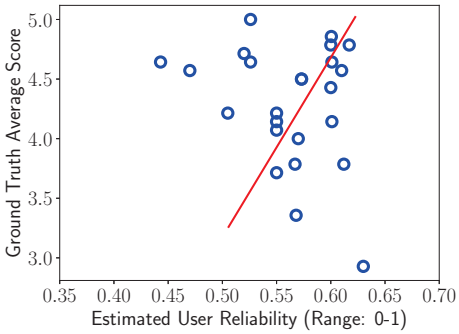


Figure 3.4: Estimated User Reliability V.S. Ground Truth User Score.

exhibit the estimated user reliabilities by the proposed approach. As there are no direct user reliability values on the CQA dataset, we only investigate the estimated user reliabilities on the student exam dataset. Specifically, we use the average score of a student's answer to each question as the groundtruth reliability. Intuitively, the learned two-fold user reliability parameters (i.e., ϕ^0 and ϕ^1) are not directly proportional to the true user reliability; we use the metric $prec$ defined in Eq. (3.19) for user reliability validation. Due to space limitation, we only show one example, which comes from the mid-result of TextTruth on exam 10, in Figure 3.4. In Figure 3.4, each point denotes a user. The Y-axis is the user reliability groundtruth and the X-axis is the estimated user precision score. As one can see, the estimated user reliability score (X) typically increases when the groundtruth user reliability (Y) increases which means that the proposed TextTruth successfully captures the reliabilities of users.

3.5 Summary

Truth discovery has shown its effectiveness in a wide range of applications with structured data. However, existing methods all suffer on unstructured text data, due to the semantic ambiguity of natural languages and the complexity of text answers. To tackle these challenges, in this chapter, we propose a probabilistic model named TextTruth that takes vector representations of key factors extracted from answers as inputs and

outputs the ranking of answers based on the trustworthiness of key factors within each answer. Specifically, the model jointly learns the clustering label and truth label for each answer factor cluster through modeling the generative process of answer factors' embedding representations. Experimental results on three real-world datasets prove the effectiveness of the proposed TextTruth model. Furthermore, case studies illustrate that the learned labels are interpretable.

Part II

Vulnerabilities Analysis of Multi-Sourced Data

Chapter 4

Data Poisoning Attack against Knowledge Graph Embeddings

4.1 Introduction

Knowledge graphs have become a critical resource for a large collection of real world applications, such as information extraction [53], question answering [86] and recommendation system [88]. Due to its wide application domains, both academia and industry have spent considerable efforts on constructing large-scale knowledge graphs, such as YAGO [26], Freebase [8], and Google Knowledge Graph[1]. In knowledge graphs, knowledge facts are usually stored as (*head entity, relation, tail entity*) triples. For instance, the fact triple *(Albert Einstein, Profession, Scientist)* means that Albert Einstein's profession is a scientist.

Although such triples can effectively record abundant knowledge, their underlying symbolic nature makes them difficult to be directly fed to many machine learning models. Hence, knowledge graph embedding (KGE), which projects the symbolic entities and relations into continuous vector space, has quickly gained significant attention [9, 44, 57, 76, 84]. These compact embeddings can preserve the inherent charac-

[1] https://developers.google.com/knowledge-graph/

teristics of entities and relations while enabling the use of these knowledge facts for a large variety of downstream tasks such as link prediction, question answering, and recommendation.

Despite the increasing success and popularity of Knowledge graph embeddings, their robustness has not been fully analyzed. In fact, many knowledge graphs are built upon unreliable or even public data sources. For instance, the well known *Freebase* harvests its data from various sources including individual, user-submitted wiki contributions[2]. The openness of such data unfortunately would make KGE vulnerable to malicious attacks. When being attacked, substantial unreliable or even biased knowledge graph embeddings would be generated, leading to serious impairment and financial loss of many downstream applications. For instance, a variety of recommendation algorithms (e.g., [79, 88]) utilize KGEs of products as external references. If KGEs are manipulated, the recommendation results will be biased. This phenomenon can largely hurt user experiences. Therefore, there is a strong need for the analysis of the vulnerability of knowledge graph embeddings.

In this chapter, for the first time, we systemically investigate the vulnerability of KGE, through designing efficient adversarial attack strategies. Due to the unique characteristics of knowledge graph and its embedding models, existing adversarial attack methods on graph data [7, 74, 94] cannot be directly applied to attack KGE methods. First, they are all designed for homogeneous graphs, in which there is only a single type of nodes or links. However, in a knowledge graph, both the entities (nodes) and the relations (links) between entities are of different types. Second, existing attack methods for homogeneous graphs usually have strict requirements on the formulation of the targeted methods. For instance, the attack strategies proposed in [7, 74] can only work for the embedding methods that can be transformed into matrix factorization. However, the KGE methods are diverse and may not be able to be transformed into matrix factorization problems.

[2]https://www.nytimes.com/2007/03/09/technology/09data.html

Our proposed attack strategies can guide the adversary to manipulate the training set of KGE by adding and/or deleting some specific facts to promote or degrade the plausibility of specific targeted facts, which can potentially influence a large variety of applications that utilize the knowledge graph. The proposed strategies include both direct scheme which directly manipulates the embeddings of entities involved in the targeted facts and indirect scheme which utilizes other entities as proxies to achieve the attack goal.

4.2 Problem Definition

Let us consider a knowledge graph $\mathcal{KG}$, with a training set denoted as $\{(e_n^h, r_n, e_n^t)\}_{n=1}^N$ and a targeted fact triple $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ that does not exist in the training set. The goal of the attacker is to manipulate the learned embeddings, which would *degrade* (or *promote*) the plausibility of $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ measured by a specific *fact plausibility scoring* function f. Without loss of generality, we focus on *degrading* the targeted fact. We also assume that the attacker has a limited attacking budget. *In this chapter, the attacking budget is the number of perturbations per target*. Formally, the attack task is defined as follows:

Definition 6 (Problem Definition)**.** *Consider a targeted fact triple* $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ *that does not exist in the training set, we use* $\boldsymbol{e}_x^{h,target}$ *to denote the embedding of the head entity* $e_x^{h,target}$*,* $\boldsymbol{e}_x^{t,target}$ *to denote the embedding of the tail entity* $e_x^{t,target}$ *and* $\boldsymbol{r}_x^{target}$ *to denote the embedding of the relation* r_x^{target} *from the original training set. Our task is to minimize the plausibility of* $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$*, i.e.,* $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})$*, by making perturbations (i.e., adding/deleting facts) on the training set. We assume the attacker has a given, fixed budget and is only capable of making* M *perturbations.*

Due to the discrete and combinatorial nature of the knowledge graph, solving this problem is highly challenging. Intuitively, in order to manipulate the plausibility of a

specific targeted fact, we need to shift either the embedding vectors related to its entities or the embedding vectors/matrices related to its relations. However, in a knowledge graph, the number of facts that a relation type involves is much larger than the number of facts that an entity type involves. For instance, in the well-known knowledge graph *Freebase*, the number of entities is over 30 million, while the number of relation types is only 1345. This leads to the fact that the innate *characteristics* of each relation type is far more stable than that of entities and is difficult to be manipulated via a small number of modifications. Hence, in this chapter, we focus on manipulating the plausibility of targeted facts from the perspective of entities. To achieve the attack goal, in the rest of this section, we propose a collection of effective yet efficient attack strategies.

4.3 Direct Attack

Given the uncontaminated knowledge graph, the goal of direct attack is to determine a collection of perturbations (i.e., fact adding/deleting actions) to shift the embeddings of the entities involved in the targeted fact to minimize the plausibility of the targeted fact. First, we determine the optimal shifting direction that the entity's embedding should move towards. Then we rank the possible perturbation actions by analyzing the training process of KGE models and designing scoring functions, which estimate the benefit of a perturbation, i.e., how much shifting can be achieved by this perturbation along the desired direction. We name the score as *perturbation benefit score* and calculate such score for every possible perturbation. Finally, we conduct the Top-M perturbations with highest perturbation benefit scores, where M is the attack budget.

Suppose we want to *degrade* the plausibility of the fact $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$. For simplicity, let's focus on shifting the embedding of one of the entities in $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$, say head entity $e_x^{h,target}$, from $\boldsymbol{e}_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$, without loss of generality. Here, $\boldsymbol{\epsilon}_x^*$ denotes the *embedding shifting vector*. The fastest direction of decreasing $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})$ is opposite to its partial derivative with respect

to $\boldsymbol{e}_x^{h,target}$. Let ϵ_h be the perturbation step size, the optimal embedding shifting vector is:

$$\boldsymbol{\epsilon}_x^* = -\epsilon_h \cdot \frac{\partial f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})}{\partial \boldsymbol{e}_x^{h,target}}. \tag{4.1}$$

As mentioned in the problem definition, in order to shift $\boldsymbol{e}_x^{h,target}$ by $\boldsymbol{\epsilon}_x^*$, the adversary is allowed to add perturbation facts to the knowledge graph or delete facts from the knowledge graph. Given the optimal embedding shifting vector $\boldsymbol{\epsilon}_x^*$, we then find a ranking of the all the perturbation (add or delete) candidates. We discuss the two schemes in detail as follows.

4.3.1 Direct Deleting Attack.

Consider the uncontaminated training set, under the direct adversarial attack scheme, in order to shift the embedding of $e_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$, we need to select and delete one or more facts that directly involve entity $e_x^{h,target}$. Intuitively, the fact to delete should have a great influence on the embedding of $e_x^{h,target}$, while at the same time not hinder the process of shifting the embedding of $e_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$. To design a scoring criterion that captures these intuitions, let us look into the training process of KGE model. Consider the specific deletion candidate $(e_x^{h,target}, r_i, e_i^t)$ that involves $e_x^{h,target}$. During training, the sum of the fact plausibility scores of the observed training samples is maximized. On one hand, the more plausible the fact $(e_x^{h,target}, r_i, e_i^t)$ is, the more it contributes to the final embedding of $e_x^{h,target}$. Hence, the perturbation benefit score of deleting $(e_x^{h,target}, r_i, e_i^t)$ should be proportional to $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$. On the other hand, if the plausibility of fact $(e_x^{h,target}, r_i, e_i^t)$ is large after $\boldsymbol{e}_x^{h,target}$ is shifted to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$ (i.e., $f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$ is large), it means that the fact $(e_x^{h,target}, r_i, e_i^t)$ has a great positive impact on the embedding shifting and should not be deleted. Hence, the perturbation benefit score of deleting $(e_x^{h,target}, r_i, e_i^t)$ should be inversely proportional to $f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$. Formally, let the set of all the delete candidates be: $\mathcal{D}_{\mathcal{D}} = \{(e_i^h, r_i, e_i^t) \mid e_i^h = e_x^{h,target} \text{ and } (e_i^h, r_i, e_i^t) \in \mathcal{KG}\}$, which intuitively denote the set of

facts that involve $e_x^{h,target}$ as the head entity in the training set. The perturbation benefit score of deleting a specific perturbation fact $(e_x^{h,target}, r_i, e_i^t)$ can be estimated as:

$$\begin{aligned}\eta^-(e_x^{h,target}, r_i, e_i^t) =& f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_i, \boldsymbol{e}_i^t) \\ &- \lambda_1 f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t),\end{aligned} \tag{4.2}$$

where $\boldsymbol{e}_x^{h,target}$, $\boldsymbol{r}_i$, and $\boldsymbol{e}_i^t$ denote the embeddings of $e_x^{h,target}$, r_i and e_i^t on the uncontaminated training set.

4.3.2 Direct Adding Attack.

Now we discuss how to conduct direct adding perturbation. To shift the embedding of $e_{x^{h,target}}$ by $\boldsymbol{\epsilon}^*$, we just need to add new facts that involve $e_x^{h,target}$ to make $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_j, \boldsymbol{e}_j^t)$ less plausible. The set of all the possible adding candidates can be denoted as $\mathcal{D}_{\mathcal{A}} = \{e_x^{h,target}\} \times \{(r_j, e_j^t) \mid \forall r_j \in \mathcal{KG} \text{ and } e_j^t \in \mathcal{KG}\}$, where $\{(r_j, e_j^t) \mid \forall r_j \in \mathcal{KG} \text{ and } e_j^t \in \mathcal{KG}\}$ denotes all the possible "relation-tail entity" combinations in the knowledge graph and $\times$ stands for Cartesian product. In practice, for better efficiency, we can downsample a subset from all the possible "relation-tail entity" combinations. Formally, the perturbation benefit score of a specific candidate to add (i.e., $(e_x^{h,target}, r_j, e_j^t)$) can be estimated as:

$$\begin{aligned}\eta^+(e_x^{h,target}, r_j, e_j^t) =& \lambda_2 f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_j, \boldsymbol{e}_j^t) \\ &- \lambda_3 f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_j, \boldsymbol{e}_j^t),\end{aligned} \tag{4.3}$$

where $\boldsymbol{e}_x^{h,target}$, $\boldsymbol{r}_j$, and $\boldsymbol{e}_i^t$ denote the embeddings on the uncontaminated training set.

4.4 Indirect Attack

Although the direct attack strategy is intuitive and effective, it is possible to be detected by data sanity check. In this section, we move on to introduce a more complicated yet

more stealthy adversarial attack scheme, i.e., indirect attack. Suppose a KGE user want to query the plausibility of a potential fact (h, r, t). Due to the huge scales of real-world knowledge graphs, even in the most optimistic situation, we may merely carry data sanity test on the facts related to h and t. However, for indirect attack, instead of adding or deleting the facts that involve the entities in the targeted fact, we propose to perturb the facts that involve other entities in the knowledge graph and let the perturbation effect propagate to the targeted fact. Thus, detecting these perturbations requires data sanity tests on facts that involves every entity that are hops away from h and t. When the number of hops increases linearly, the data sanity cost will have a exponential growth. Even though there is an Oracle that can find these anomalous facts effectively, defenders cannot determine the targeted fact(s) of these perturbations. For a better description, we provide the following toy example, which is used throughout this section.

Example 1. *Suppose we want to degrade the plausibility of the targeted fact* $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ *via shifting the embedding of the targeted entity* $e_x^{h,target}$ *by* $\boldsymbol{\epsilon}_x^*$*, without loss of generality. Under indirect attack scheme, we perturb the facts that involve the K-hop neighbors of* $e_x^{h,target}$*. These K-hop neighbors are called* proxy entities*. Then the entities between the K-hop neighbors (proxy entities) and* $e_x^{h,target}$ *are* intermediate entities *to propagate the influence of the perturbations to* $e_x^{h,target}$*. The propagation path can be illustrated as follows:*

$$e_x^{h,target} \overset{r_{x,1}}{\longleftrightarrow} e_{x,1} \overset{r_{x,2}}{\longleftrightarrow} e_{x,2} \cdots \overset{r_{x,K}}{\longleftrightarrow} e_{x,K}$$

where we use $\overset{r_{x,\cdot}}{\longleftrightarrow}$ *to denote the directional relation and use notation* $e_{x,\cdot}$ *to denote the entities on the path. A specific* $e_{x,\cdot}$ *can work as both the head entity and the tail entity. The notations in the path above are adopted in the rest of this section.*

When the perturbations on the proxy entity cause an embedding shift on itself, the embeddings of its neighboring entities will also be influenced. The influence will propagate back to the embedding of the targeted entity ultimately.

However, finding the effective perturbations on the proxy entities, which are K-hop away from the targeted entity, is indeed a challenging task. The task involves two key problems: **(1)** *Given a specific propagation path, how can we determine the desired embedding shifting vectors on its intermediate entities and its proxy entity, in order to accomplish the embedding shifting goal on the targeted entity?* **(2)** *How do we select the propagation paths to propagate the influence of perturbation to the targeted entity?* In the rest of this section, we discuss strategies to solve these key problems and propose a criterion to evaluate the benefit of an indirect perturbation (i.e., the *perturbation benefit score*).

For the first problem, given a specific path, in order to conduct a perturbation that makes the embedding of $e_x^{h,target}$ shift towards the desired direction (i.e., the direction of $\boldsymbol{\epsilon}_x^*$), we decide the shifting goal for each entity on the path in a *recurrent* way. Suppose we want to shift $\boldsymbol{e}_x^{h,target}$ by $\boldsymbol{\epsilon}_x^*$ via the intermediate entities along the path specified in Example 1. The entity that directly influences $\boldsymbol{e}_x^{h,target}$ is its neighbor $e_{x,1}$ and what we need to do is to determine the ideal embedding shifting vector $\boldsymbol{\epsilon}_{x,1}^*$ on $e_{x,1}$, so that the desired embedding shift on $e_x^{h,target}$ (i.e., $\boldsymbol{\epsilon}_x^*$) is approached to the greatest extent. Formally, $\boldsymbol{\epsilon}_{x,1}^*$ should satisfy:

$$\begin{aligned} \boldsymbol{\epsilon}_{x,1}^* = \arg\max_{\boldsymbol{\epsilon}} \; & f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_{x,1}, \boldsymbol{e}_{x,1} + \boldsymbol{\epsilon}) \\ & - f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_{x,1}, \boldsymbol{e}_{x,1} + \boldsymbol{\epsilon}) \\ s.t. \; & ||\boldsymbol{\epsilon}||_2 = \epsilon_h, \end{aligned} \tag{4.4}$$

where ϵ_h is the perturbation step size, $\boldsymbol{e}_{x,1}$ denotes the embedding of $e_{x,1}$, and $\boldsymbol{r}_{x,1}$ denotes the embedding of $r_{x,1}$. As a result, the embedding of $e_x^{h,target}$ will have a larger tendency to move towards $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$ than towards $\boldsymbol{e}_x^{h,target}$, during the training process on the contaminated training data. When $\boldsymbol{\epsilon}_{x,1}^*$ is determined, we can further get the embedding shifting vector for $e_{x,2}, \cdots, e_{x,K}$, which are denoted as $\boldsymbol{\epsilon}_{x,2}^*, \cdots, \boldsymbol{\epsilon}_{x,K}^*$, respectively. This process is similar as above.

With the embedding shifting vectors on the proxy entities of each path determined, we calculate the scores η^- and η^+, defined in Eq. (4.2) and (4.3) for all the possible add/delete perturbations. These scores are later used to calculate the perturbation benefit score under indirect attack schemes.

For the second problem, we look into the training objective function. Suppose we want to shift the embedding of $e_{x,k-1}$ via its neighbor $e_{x,k}$, when the embedding shift on $\boldsymbol{e}_{x,k}$ is $\boldsymbol{\epsilon}^*_{x,k}$. To estimate the influence of such embedding shift on $\boldsymbol{e}_{x,k-1}$, we isolate all the facts that involve $e_{x,k-1}$ in the training objective function, force a embedding shift $\boldsymbol{\epsilon}_{x,k}$ on $\boldsymbol{e}_{x,k}$ and ignore the negative sampling terms. Formally, the objective function becomes: $\min_{\boldsymbol{e}_{e_{x,k-1}}} \sum_{(e_i^h, r_i, e_i^t) \in D_{e_{x,k-1}}^{\setminus e_{x,k}}} \mathcal{L}(\boldsymbol{e}_i^h, \boldsymbol{r}_i, \boldsymbol{e}_i^t) + \mathcal{L}(\boldsymbol{e}_{x,k-1}, \boldsymbol{r}_{x,k}, \boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k})$, where $D_{e_{x,k-1}}^{\setminus e_{x,k}}$ stands for the set of all the observed facts, which involve $e_{x,k-1}$ except the fact $(e_{x,k-1}, r_{x,k}, e_{x,k})$, in the training set. $\mathcal{L}$ denotes the loss function for a single fact. $\boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k}$ in $\mathcal{L}(\boldsymbol{e}_{x,k-1}, \boldsymbol{r}_{x,k}, \boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k})$ indicates that the embedding of $e_{x,k}$ is already shifted. Clearly, if we fix the embeddings of all the relations and entities except $e_{x,k-1}$, the impact of shifting $\boldsymbol{e}_{x,k}$ to $\boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k}$ is highly correlated with the number of facts that involves $e_{x,k-1}$, i.e., $|D_{e_{x,k-1}}|$. That is to say, the more neighbors an entity has, the *less* it will be influenced by a specific perturbation on one of its neighbors.

Based on above discussions, we propose an empirical scoring function to evaluate the perturbation benefit score of every possible perturbation. We still consider the scenario specified in Example 1. Suppose we conduct an add/delete perturbation $(e_{x,K}, r_{x,K}, e_{x,K+1})$ on the proxy entity $e_{x,K}$. The perturbation benefit score of this indirect perturbation is defined as:

$$\begin{aligned} &\psi(e_{x,K}, r_{x,K}, e_{x,K+1}) \\ &= \eta(e_{x,K}, r_{x,K}, e_{x,K+1}) - \lambda \log \big(\frac{1}{K} \sum_{k=1}^{K-1} |D_{e_{x,k-1}}| \\ &\quad + \max(\{|D_{e_{x,k-1}}|\}_{k=1}^{K-1}) \big), \end{aligned} \tag{4.5}$$

where $\max(\{|D_{e_{x,k-1}}|\}_{k=1}^{K-1})$ stands for the maximum number of facts that involves each entity k on the path. η is the same as η^+ under add perturbation scheme and is the same as η^- under delete perturbation scheme. λ is a trade-off parameter. The first term estimates the direct perturbation benefit of the perturbation in terms of shifting the proxy entity as desired. The second term evaluates the capability of the intermediate entities on the path in terms of propagating the influence to the targeted entity. As the influence may be diluted by the facts that involve each entity $e_{x,k}$ on the path. A smaller averaged number of facts that involves each entity $e_{x,k}$ on the path indicates a larger capability of the path in terms of propagating the influence. Moreover, we also consider the maximum number of facts that involves each entity $e_{x,k}$ on the path. This is to avoid the case when some intermediate entities, whose embedding is difficult to shift, "block" the propagation path. *In practices, we can first utilize the second term to determine the best P paths in terms of propagating the influence from proxy entities to the targeted entity and then choose what facts to add or delete upon these proxy entities in the best P paths.* [3]

4.5 Experiments

4.5.1 Datasets.

In this chapter, we use two common KGE benchmark datasets for our experiment: FB15k and WN18. FB15k is a subset of Freebase, which is a large collaborative knowledge base consisting of a large number of real-world facts. WN18 is a subset of Wordnet [4], which is a large lexical knowledge graph. Both FB15k and WN18 are first introduced by [9]. The training set and the test set of these two datasets are already fixed. *We randomly sample 100 samples in the test set as the targeted facts for the proposed attack strategies.*

[3] This strategy is used in the experiments of this chapter.

[4] https://wordnet.princeton.edu/

4.5.2 Baseline & Targeted Models.

Since there are no existing methods that can work under the setting of this chapter, we compare the proposed attack schemes with several naive baseline strategies. Specifically, we design *random-dd* (random direct deleting), *random-da* (random direct adding), *random-id* (random indirect deleting), *random-ia* (random indirect adding) as comparison baselines for our proposed *direct deleting attack*, *direct adding attack*, *indirect deleting attack*, *indirect adding attack*, respectively. The difference between the baseline and its corresponding proposed methods is that the perturbation facts to add/delete are randomly selected. For the targeted KGE models, we choose three most representative *TransE* [9], *TransR* [44] and *RESCAL* [57] as attack targets.

4.5.3 Metrics.

In order to evaluate the effectiveness of the proposed attack strategies. We compare the plausibility change of the targeted fact before and after the adversarial attack. Specifically, we follow the evaluation protocol of KGE models described in the previous works like [9]. Given a targeted fact (e_h, r, e_t), we remove the head or tail entity and then replace it with all the possible entities. We first compute plausibility scores of those corrupted facts and then rank them by descending order; the rank of the correct entity is stored. After that, we use *MRR* (Mean Reciprocal Rank of all the ground truth triples) and *H@10* (the proportion of correct entities ranked in top 10, for all the ground truth entities.) as our evaluation metrics. *The smaller MRR and H@10 are on the contaminated dataset, the better the attack performance is.*

4.5.4 Results and Analysis

In this section, we report and analyze the attack results of the proposed attack strategies under different settings. To avoid confusion, the performance of direct adding attack,

		Clean		random-da		Direct Add	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.24	0.46	0.24	0.42
	TransR	0.24	0.52	0.23	0.42	0.21	0.41
	RESCAL	0.19	0.42	0.20	0.40	0.17	0.39
WN18	TransE	0.39	0.70	0.30	0.68	0.21	0.53
	TransR	0.44	0.73	0.41	0.71	0.22	0.51
	RESCAL	0.41	0.72	0.44	0.69	0.30	0.57

Table 4.1: Overall Results of Direct Adding Attack

		Clean		random-dd		Direct Delete	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.26	0.54	0.19	0.37
	TransR	0.24	0.52	0.25	0.49	0.18	0.41
	RESCAL	0.19	0.42	0.19	0.38	0.13	0.30
WN18	TransE	0.39	0.70	0.36	0.71	0.11	0.26
	TransR	0.44	0.73	0.43	0.68	0.11	0.24
	RESCAL	0.41	0.72	0.40	0.67	0.02	0.05

Table 4.2: Overall Results of Direct Deleting Attack

direct deleting attack, indirect adding attack, and indirect deleting attack are reported separately in Table 4.1, 4.2, 4.3 and 4.4.

4.5.5 Overall Attack Performance.

Let us first discuss the performances of the direct attack schemes on two datasets. For the direct deleting attack scheme, we set the attack budget for each targeted fact to 4 and 1 on FB15K and WN18 dataset, respectively. For the direct attacking attack scheme, the attack budgets for each targeted fact are 8 and 6 for FB15K and WN18 dataset, respectively. These budgets are low enough to make the whole attack process unnoticeable. From the results, we can clearly see that the plausibilities of these targeted facts significantly degrade as desired. We can conclude that these KGE models are quite vulnerable to even a small number of perturbations generated by well-designed attack strategies. For comparison, we have also tested the baseline methods *random-da* and

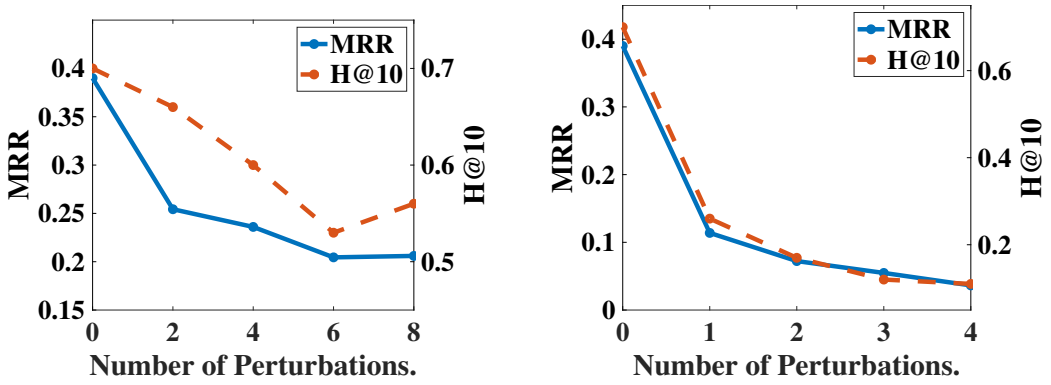
		Clean		random-ia		Indirect Add	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.25	0.50	0.23	0.47
	TransR	0.24	0.52	0.25	0.51	0.22	0.49
	RESCAL	0.19	0.42	0.19	0.40	0.17	0.36
WN18	TransE	0.39	0.70	0.42	0.71	0.32	0.67
	TransR	0.44	0.73	0.40	0.73	0.34	0.69
	RESCAL	0.41	0.72	0.41	0.69	0.39	0.63

Table 4.3: Overall Results of Indirect Adding Attack

random-dd, which cannot achieve satisfactory attack performances. This demonstrates the effectiveness of the proposed strategies. Moreover, we observe that the effectiveness of the proposed strategies is more significant on WN18 dataset than on FB15K dataset. This is because the average number of facts that each entity involves in WN18 dataset is significantly smaller than that in FB15K dataset. Hence, the graph structure of FB15K is more stable and robust. Then, let us move on to the discussion of indirect attack schemes. For the indirect adding attack, we set the attack budget for each targeted fact to 60 and 20 for FB15K and WN18 dataset, respectively. For the indirect deleting attack, the attack budgets for each targeted fact are set to 20 and 5 for FB15K and WN18 dataset, respectively. The reason why indirect attacks need more attack budgets to get comparable results is that only a small portion of the influence caused by the perturbations on proxy entities is propagated to the targeted entity. In contrast, nearly all of the influence of the perturbation is exerted on the targeted entity under direct attack schemes. Like direct attack schemes, these indirect attack schemes also demonstrate their effectiveness. For instance, under the indirect deleting attack scheme, the H@10 and MRR metrics of the targeted facts decrease by approximate 0.03 on FB15K dataset. Thus, the indirect deleting attack schemes can also be used in practices to make the attack process more stealthy.

		Clean		random-id		Indirect Delete	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.27	0.50	0.22	0.44
	TransR	0.24	0.52	0.25	0.53	0.21	0.48
	RESCAL	0.19	0.42	0.20	0.36	0.16	0.34
WN18	TransE	0.39	0.70	0.44	0.74	0.35	0.68
	TransR	0.44	0.73	0.45	0.74	0.41	0.71
	RESCAL	0.41	0.72	0.42	0.70	0.38	0.64

Table 4.4: Overall Results of Indirect Deleting Attack



(a) Direct Adding Attack on WN18 Dataset against TransE

(b) Direct Deleting Attack on WN18 Dataset against TransE

Figure 4.1: Analysis of the Number of Perturbations

4.5.6 Analysis of the Number of Perturbations.

When conducting the data poisoning attack, one of the most important factors is the number of perturbations (i.e. attack budget). Due to space limit, we merely plot performances of direct attack schemes against TransE w.r.t. the number of perturbations (i.e., attack budgets) on WN18 dataset in Figure 4.1. From Figure. 4.1, we can clearly see that the proposed attack strategies consistently degrade the plausibility of the targeted facts under both setting. When the number of perturbations keeps increase, the growth of attack performance becomes slower. This is because when the number of perturbations is small, the selected perturbations are usually of high value in terms of manipulating

the plausibility of the targeted facts. When the number of perturbations keeps increase, the high-value perturbations are used up. Hence, the performances become stable.

4.6 Summary

We present the first study on the vulnerability of existing KGE methods and propose a collection of data poisoning attack strategies for different attack scenarios. These attack strategies can be efficiently computed. Experiment results on two benchmark dataset demonstrate that the proposed strategies can effectively manipulate the plausibility of arbitrary facts in the knowledge graph with limited perturbations. Moreover, the perturbation generation processes are quite efficient and can be parallelized. As future work we aim to derive defence strategies for KGE models so that these models are more robust against adversarial attacks.

Chapter 5

Data Poisoning Attack against Next-Item Recommendation

5.1 Introduction

It is commonly assumed that online recommendation systems are honorable and unbiased. They recommend users the items that match their personal interests. However, the openness of recommendation systems and the potential benefit of manipulating recommendation systems offer both opportunities and incentives for malicious parties to launch attacks. Recent studies [32, 34, 54, 59, 85] have demonstrated that recommendation systems are vulnerable to poisoning attacks. In these poisoning attacks, well-crafted data is injected into the training set of a recommendation system by a group of malicious users. Such poisoning attacks make the system deliver recommendations as attackers desire.

Existing poisoning attacks can be categorized into two types. The first type of work is generally based on manually designed heuristic rules. For example, [59] design rules that leverage the following intuition: items that are usually selected together by users are treated as highly correlated by recommendation systems. To promote a target item to target users, attackers utilize controlled users to fake the co-occurrence between the target

item and popular items. Nevertheless, such heuristic rules are not able to cover various patterns of behavior in the recommendation data. Therefore, the performances of these attack methods are usually unsatisfactory. The other line of methods are designed for certain types of recommendation methods like matrix factorization based models [34]. However, the architecture and the parameters of the recommendation systems in real-world platforms are generally unknown to the attackers. Usually, the only information that the attackers can rely on to infer the characteristics of the recommendation systems is the recommendation results of the users they controlled, and the frequency of these interactions is often limited. Thus, there is still a noticeable gap before these attacks methods can be deployed in real practice.

In this work, we propose a novel practical adversarial attack framework against sophisticated blackbox recommendation systems. We focus on one of the most common next-item recommendation setting, which aims to recommend top-K potentially preferred items for each user. The proposed reinforcement learning based framework *LOKI* learns an attack agent to generate adversarial user behavior sequences for data poisoning attacks. Unlike existing attack methods designed for certain types of recommendation methods, reinforcement learning algorithms can utilize the feedback from the recommendation systems, instead of comprehensive knowledge of architecture and parameters, to learn the agent's policy. Nevertheless, in practice, the attacker cannot control the target recommendation system to be retrained to get the feedback and update the attack strategy. In addition, recommendation system service providers generally restrict feedback frequency, but a reinforcement learning based framework requires a large number of feedback to train a policy function. Due to this discrepancy, we cannot directly rely on the feedback from the target recommendation system to train a policy within a tolerated time period.

To tackle this challenge, we propose to construct a local recommender simulator to imitate the target model, and let the reinforcement framework get reward feedback from the recommender simulator instead of the target recommendation system. The local

recommender simulator is constructed by constructing an ensemble of multiple representative recommendation models. The intuition behind such a design is that if two recommenders can both get similar recommendation results on a given dataset, then the adversarial samples generated for one of the recommenders can be used to attack the other. Such transferability makes the recommender simulator a good substitute for the target recommendation system in terms of guiding the attack agent. Moreover, even with the help of a local simulator, it is still time-consuming to retrain the recommendation systems within the simulator using the contaminated data for attack outcome. To alleviate this problem, we design a component named outcome estimator, which is based on the influence function. The outcome estimator can efficiently estimate the influence of the injected adversarial samples on the attack outcomes. These designs ensure that the proposed adversarial attack framework for recommendation systems is practical and effective.

5.2 Problem Definition

To facilitate the discussions in the rest of this chapter, we specify and formulate the next-item recommendation task as follows:

Definition 7 (Next Item Recommendation)**.** *Let $\mathcal{U}$ be the set of users and $\mathcal{V}$ be the set of items, we use $\boldsymbol{x}_u = [x_u^1, x_u^2, \cdots, x_u^{m_u}]$ to denote a sequence of items that user u has chosen before in a chronological order in which $x_u^v \subseteq \mathcal{V}$. m_u denotes the number of items chosen by user u. Given existing sequences, the goal of the next-item recommendation is to output a K-sized ordered item list, which predicts the next item that the user will choose.*

With the aforementioned definition, let us detail the threat model of the attack against the next-item recommendation.

Attack Goal: An attacker's goal is to promote a set of target items to as many target users as possible. Specifically, suppose the system recommends K items to each user,

the attacker's goal is to maximize averaged display rate, which denotes the fraction of target users whose top-K recommendations results include the target items. Note that an attacker could also demote a target item. Demotion can be viewed as a special case of promotion as an attacker can promote other items such that the target item is demoted in recommendation lists. Thus, in this chapter, we focus on promotion attacks.

Attack Approach: To achieve the attack goal, we consider the most general scenario in which the attackers can inject controlled users into the recommendation system. These controlled users visit or rate to well-selected items, which are named as *proxy items*, step-by-step. Thus, the well-crafted activities of each controlled user form a *behavior sequence*. To make the injection unnoticeable, the number of visits or ratings each controlled user conducts is limited to at most M.

The Knowledge and Capability of the Attacker: In this chapter, we assume that the attacker is granted the following knowledge and capability.

1. The attacker can access the full activity history of all the users in the recommendation system.

2. The attacker has limited resources so the attacker can merely inject a limited number of controlled users which can easily be bought from the underground market[1].

3. The attacker does not know the details about the target recommendation system, for instance, the parameters and the architecture of the recommendation model. Such setting is also known as *blackbox setting*.

4. The attacker can only receive a limited number of feedback (e.g., display rates) from the blackbox recommendation model.

5. The attacker does NOT know when the target blackbox recommendation model is retrained.

[1] https://www.buzzfeednews.com/article/leticiamiranda/amazon-marketplace-sellers-black-hat-scams-search-rankings

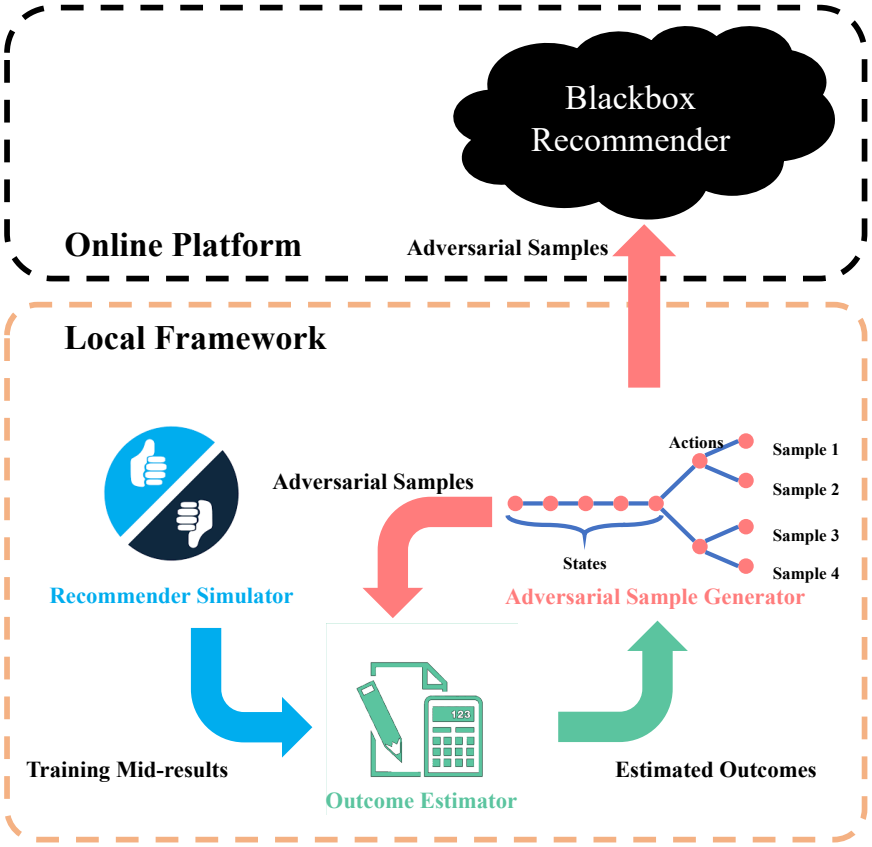


Figure 5.1: Overview of the proposed framework LOKI.

5.3 Methodology

In this section, we first provide an overview of the proposed reinforcement learning based framework. Then we describe the detailed design of each component of the framework.

5.3.1 Framework Overview

Intuitively, data poisoning can be regarded as the creation of new sequential patterns that involve the target items in the training set of the target recommendation system. In a crafted sequential adversarial sample, the user behavior history is inherently crucial in determining the next behavior. These sequential adversarial samples together contribute

to the manipulation goal. Generating adversarial samples is essentially a multi-step decision process, in which the generator ought to select specific actions for the controlled users to maximize attack outcome. This fits the reinforcement learning setting. From the perspective of reinforcement learning, the goal is to learn a policy function to generate sequential adversarial user behavior samples, which can maximize the averaged display rate of the target users.

Based on the aforementioned motivation, we propose a reinforcement learning based framework to learn the policy function. The overall architecture of the proposed framework LOKI is illustrated in Figure 5.1. The target blackbox recommendation system is deployed on an e-commerce platform. The proposed framework consists of three components: (1) recommender simulator, (2) outcome estimator, and (3) adversarial sample generator. In the following sections, we describe the details of these components one-by-one.

5.3.2 Recommender Simulator

The idea of constructing surrogate models and utilizing the transferability property of adversarial samples to attack the target machine learning models is adopted by multiple attack approaches [4, 17, 60]. In this chapter, the proposed recommender simulator simulates the recommendation preference of the target model. The simulator consists of multiple separated recommendation models, which are trained on the same dataset. Recommendation results from these models are aggregated via weighted voting. Suppose M different recommendation models are deployed to recommend items for user u and the rank of item i in the m-th model is denoted as $rank_m(i)$. *The higher item i ranks, the smaller $rank_m(i)$ is.* Here, we define the preference score of item i in the simulator via Eq. (5.1). All the items are then ranked according to this scoring function:

$$score(i) = -\frac{1}{M}\sum_{m=1}^{M} w_m \cdot rank_m(i), \tag{5.1}$$

where w_m stands for the weight of the m-th recommendation model. Ideally, these weights are used to adjust the simulator to match the characteristics of the target recommender.

5.3.3 Outcome Estimator

As mentioned in Section 5.3.1, we need to utilize the manipulation outcome of the current adversarial samples as reward feedback to update the policy network of the adversarial sample generator. The most straightforward way to obtain the outcome is to retrain the entire model. However, retraining the online recommendation system is prohibitively slow (from a few hours to days for a single retraining). To make the attack methodology practical, we propose to use influence function for an efficient estimation of the manipulation outcome, motivated by robust statistics.

Formally speaking, the parameter estimator of the recommendation models on the clean dataset is: $\hat{\theta} := \text{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(z_i; \theta)$, where θ denotes the parameter vector, $\mathcal{L}$ stands for the loss function of the recommendation model. z_i denotes a sample in the dataset, and N stands for the total number of samples in the training set. For collaborative filtering models, a sample is a single user-item pair (u, v). For session-based recommendation models, given the behavior sequence $\boldsymbol{x}_u = [x_u^1, x_u^2, \cdots, x_u^m]$ of a user u, each training sample is made up of a subsequence and the ground truth next item, i.e., $([x_u^1], x_u^2), ([x_u^1, x_u^2], x_u^3), \cdots, ([x_u^1, x_u^2, \cdots, x_u^{n-1}], x_u^n)$.

Now let us move on to the discussion of the influence function. Suppose we upweight a sample z_δ by a small ϵ in the training set, the new estimation of θ is given as: $\hat{\theta}_{z_\delta} := \text{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(z_i; \theta) + \epsilon \mathcal{L}(z_\delta; \theta)$. When $\epsilon \to 0$, according to the classic results in [13], the influence of upweighting z_δ on the parameter θ is given by:

$$\frac{d\hat{\theta}_{z_\delta}}{d\epsilon} = \hat{\theta}_{z_\delta} - \hat{\theta} \approx -H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z_\delta; \hat{\theta}), \tag{5.2}$$

where $H_{\hat{\theta}} := \frac{1}{N}\sum_{i=1}^{N}\nabla_{\theta}^{2}\mathcal{L}(z_i,\theta)$, denotes the Hessian matrix of the loss function. Given the fact that the number of users is large in the recommendation datasets, injecting a data sample is the same as upweighting the sample by $\epsilon \approx \frac{1}{N}$.

Here, the key computation bottleneck lies in the calculation of the huge inverse Hessian matrix H_{θ}^{-1}. Given a sample z_j, we use implicit Hessian-vector products (HVPs) [1, 30] to efficiently approximate $-H_{\theta}^{-1}\nabla_{\theta}L(z_j,\hat{\theta})$.

Based on an approximate estimate of the sample upweight's influence on parameter $\hat{\theta}$, we further calculate the influence on the prediction scoring function w.r.t. the perturbation. Specifically, suppose we want to promote an product v' to user u', we can treat this as a *target sample* $z_{u'v'}^{test}$in the test set. The influence on the prediction scoring function w.r.t. can be written as:

$$\begin{aligned}\frac{df_{test}(z_{u'v'}^{test};\hat{\theta})}{d\epsilon} =& \frac{df_{test}(z_{u'v'}^{test};\hat{\theta})}{d\hat{\theta}_{z_\delta}}\cdot\frac{d\hat{\theta}_{z_\delta}}{d\epsilon}\\ \approx& -\nabla_{\theta}f_{test}(z_{u'v'}^{test};\hat{\theta})H_{\hat{\theta}}^{-1}\nabla_{\theta}\mathcal{L}(z_\delta;\hat{\theta}),\end{aligned} \tag{5.3}$$

where f_{test} is the prediction scoring function used by the recommender system in the test phase. This result is further used to design rewards for efficient agent policy training.

5.3.4 Adversarial Sample Generator

The adversarial attack against a local recommender simulator is essentially interpreted as a multi-step decision problem. In this section, we translate this decision problem into a Markov Decision Process (MDP). MDP is defined as a tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P}$ is the transition probabilities, $\mathcal{R}$ is the immediate reward, and γ is the discount factor. In the context of this chapter, the MDP can be specified as follows:

- **Action space** $\mathcal{A}$: As mentioned in Section 5.3.1, the attacker determines specific items organized in a proper sequence for each controlled user. Instead of taking the set of all the possible items as action space, we divide the item set into

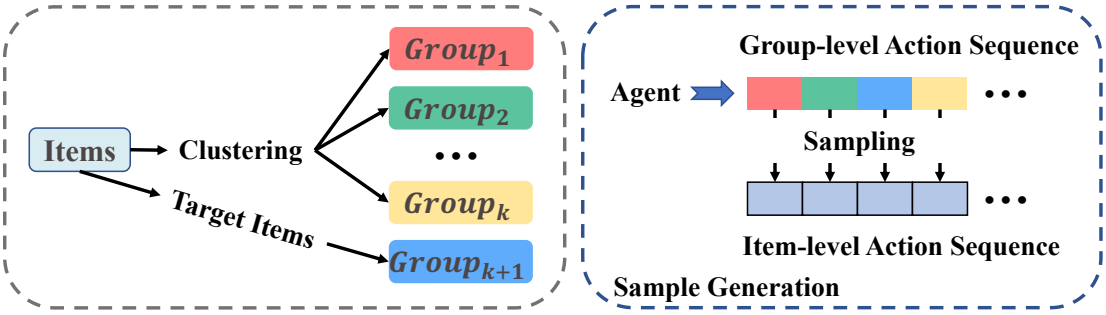


Figure 5.2: Generation of adversarial samples.

groups and use the set of all the groups as action space. The main reason for using coarse groups instead of items for action space is due to the concern in learning efficiency. Learning action strategies for every single item is not only costly but also unnecessary for the attack goal. This is because adversarial samples do not need to follow the exact sample pattern. Here we define one of the groups as the collection of all the target items, one of the groups as the collection of all the items already selected by the target users. The remaining groups are obtained by item clustering, in which each group represents items with similar properties. This item clustering takes the feature vectors of all the items and an integer c as input and divides the items into c clusters. Here, we utilize non-negative matrix factorization [33] to extract item features and use K-means [48] algorithm for clustering. After item groups are obtained, during the phases of training and testing the agent, *group-level actions* are sampled step-by-step from the policy, forming a *group-level action sequence*. Then sequential poisoning samples are sampled step-by-step from the corresponding group indicated by the current step of the *group-level action sequence*. This process is illustrated in Figure 5.2. The left side of Figure 5.2 shows the process of clustering items into groups and the right side illustrates the process of generating the poisoning samples.

- **State** $\mathcal{S}$: The state is defined as the action subsequence before current step t and the actions all come from the action space mentioned above.

- **Reward** $\mathcal{R}$: As aforementioned, the purpose of the attacker is to manipulate the local recommender simulator and further the target recommender. Hence, the RL framework should learn a policy that promotes the estimated prediction scores of target items given by the target users as much as possible. Thus, we design the reward as *the weighted averaged influence on the prediction scoring function, i.e., Eq.* (5.3)*, of all the target samples*. The weights are assigned manually to indicate the importance of each recommendation simulator.

Here, we apply Deep Q-Network (DQN) to estimate the action-value function. The representation of the existing sequence, i.e., state, is modeled via a GRU (Gated Recurrent Unit) layer, and the representation of each type of *actions* is extracted via a embedding layer. Finally, we deploy a fully connect layer which takes the final output of the GRU layer as input and output the estimated action-values.

The DQN is trained via an iterative algorithm. In each iteration, there are two stages, replay memory generation stage and parameters update stage. In replay memory generation stage, the agent generates a group-level action a_t according to an ϵ-greedy policy and current state s_t. Then the item-level sequences are generated by sampling items from the corresponding group suggested by each step in the group-level sequence. After that, the agent observes the reward r_t from the outcome estimator and updates the state. For parameter update stage: the agent samples a (s_t, a_t, r_t, s_{t+1}) from replay memory, and then updates the parameters.

In this section, we test the proposed *LOKI* on real against different recommendation methods. The experimental results show that the proposed method outperforms existing attack strategies. Besides, we systematically study the effect of some key factors.

5.3.5 Datasets

To demonstrate the performance of the proposed poisoning attack framework, we adopt the *Amazon Beauty*, which is one category of the widely used recommendation dataset series named *Amazon* [25]. The dataset used in this chapter mainly focuses on hair

and skin care products and is extracted from large corpora of product reviews crawled from *Amazon.com*. The number of users and items in the dataset are 22,363 and 12,101, respectively. The number of total user activities (i.e., purchase and review) is 146,031. On average, each user is involved in 6.53 activities and each item is involved in 12.06 activities. We followed the similar preprocessing procedure introduced in [28, 75] and filter out the users with less than five activities and items with less than five feedbacks.

5.3.6 Experimental Settings

5.3.6.1 Baseline Attack Methods

As aforementioned, there is no existing work solving exactly the same task considered in this chapter. Although there are some existing attack approaches [34] against recommendation methods, they are mostly designed for *whitebox setting* and require a strong knowledge of the architecture and parameters of the corresponding model. Therefore, these methods cannot be used in the *blackbox setting* discussed in this chapter. Hence, we compare the proposed *LOKI* with several existing heuristic-based attack strategies.

- **None**: This denotes the circumstance when no attack is conducted.
- **Random**: In this baseline method, the attacker mixes the target items and the randomly picked items to form a repository for each controlled account. In each step, the controlled user picks items at random from the item repository without repetition.
- **Popular**: This is a variant of [85]. In this baseline method, attackers inject fake co-visitations between the popular items and the target items, to promote the target items.

5.3.6.2 Target Recommendation Methods

In this section, we consider the following target methods for performance comparisons. The parameters of these target methods are set following the suggestion in the original papers.

- **BPRMF** [65] is a factorization based personalized ranking approach. It is a state-of-the-art method for non-sequential item recommendation on implicit feedback data.
- **FPMC** [66] is a classic hybrid model combing Markov chain and matrix factorization for next-basket recommendation. FPMC can model the user's long-term preference and the short-term item-to-item transition.

For each user u in the dataset, suppose the length of u's sequence is T_u, we hold the first $T_u - 2$ actions in the sequence as the training set and use the next one action as the validation set to search for the optimal hyperparameter settings for these recommendation models. The attack methods aim to manipulate the prediction of the next item, i.e. item T_u.

To simulate the interactions between the target blackbox recommendation system and the recommender simulator, we adopt the "leave-one-out strategy". That is to say we use a specific recommendation model as the target, which is blind to the attacker, and use the aggregation of all the methods as the recommender simulator. The number of target items and target users in this chapter are both fixed to be 20.

5.3.6.3 Evaluation Metric

We use the averaged *display rate*, which denotes the fraction of target users whose top-K recommendation results include the target items, as our evaluation metric. The larger the display rate is, the better the attack approach performs.

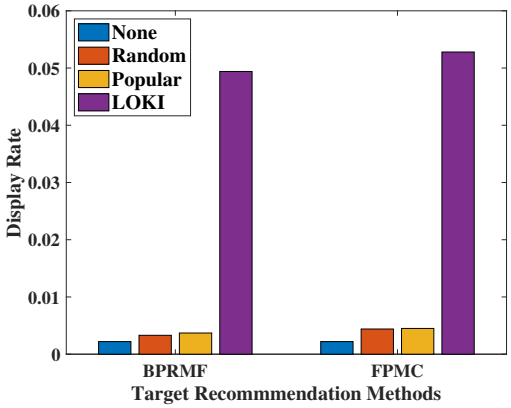
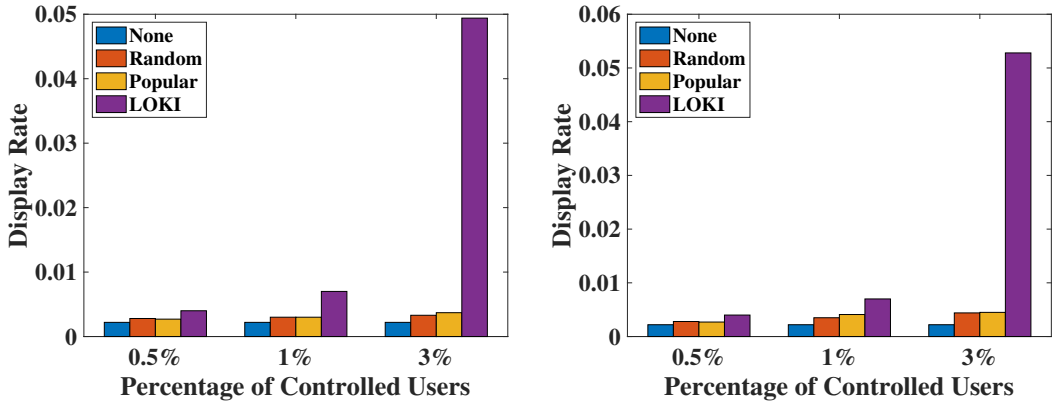


Figure 5.3: Overall performance of all the attack methods.

5.3.7 Result Analysis

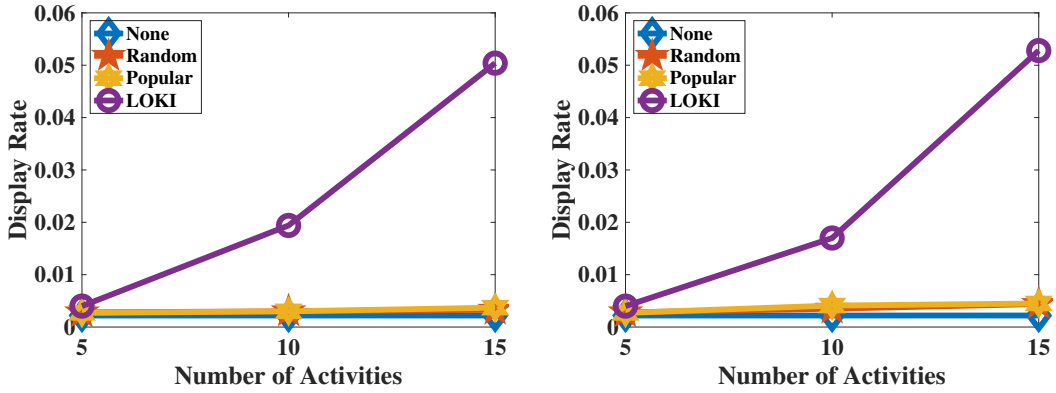
Figure 5.3 summarizes the results for all the attack methods. Here, we fix the percentage of controlled users to 3% and the number of actions per user to 15. The number of returned items is fixed to 10. In terms of attack outcome, the proposed *LOKI* achieves the best performance and the improvement is significant. For example, on compared with the best baseline, the proposed *LOKI*'s display rate increases by over eight times on average. Among the baseline methods, *Random* simply lets the target items occur in the poisoning sequences without actually creating any new pattern that favors the recommendation of the target items. Thus, *Random* has the worst performance. *Popular* fakes the co-visitations between the popular items and the target items without considering whether these popular items indeed overlap with the preferences of the target users. Thus, they cannot get a satisfactory performance too.

Compared with these baselines, the proposed *LOKI* takes advantage of the feedback from the local simulator to train an attack agent. The learned agent is capable of creating more complex patterns to achieve data poisoning goals. We also notice that the more complicated the target model is, the higher increase in performance metric the proposed *LOKI* achieves. For instance, the performance gap is larger when attacking FPMC than attacking BPRMF. This is because advanced methods are able to capture more complicated user patterns within user behavior sequences. The capability to capture various



(a) VS BPRMF (b) VS FPMC

Figure 5.4: Impact of the percentage of the controlled users.



(a) VS BPRMF (b) VS FPMC

Figure 5.5: Impact of the number of activities per controlled user.

user patterns leads to better prediction performance in general, but at the same time, enables more room for the attack improvement by the proposed *LOKI* with its ability of creating new patterns to poison the recommendation. That is to say, in some circumstances, the proposed *LOKI* can create certain sequential patterns. However, since relatively simple methods cannot capture these crafted patterns, these methods are less sensitive to the adversarial samples generated by the proposed *LOKI*.

5.3.8 Parameter Analysis

After discussing the overall experimental results and the characteristics of vulnerable users, we demonstrate the impact of two attack budgets, i.e. (1) the percentage of controlled users recruited by the attacker; (2) the number of activities that each controlled user conducts.

Impact of the Percentage of the Controlled Users. In this experiment, we consider the case where the percentage of the controlled user is low, and evaluate the performance of *LOKI* when this percentage is varied. Here "*percentage*" is calculated as the number of the controlled users over the number of normal users. The number of activities per controlled user is fixed to 15 and the recommendation system returns top 10 items. The *display rate* for the real-world datasets is shown in Figure 5.4. From the figure, we can clearly see that the proposed *LOKI* outperforms the baselines in all cases and can successfully promote the target items. For instance, when attacking against FPMC, the display rate increases to 0.055 even when the percentage of the controlled users is as low as 3%. Thus, we can conclude that the attack proposed in this chapter is effective even with a scant attack budget.

Impact of the Number of Activities per Controlled User. When the percentage of controlled users is given, the number of activities each controlled user conducts is another important attack factor. In this experiment, we fix the percentage of the controlled users to be 3% and vary the number of activities each controlled user conducts from 5 to 15 for all the datasets. The recommendation system returns top 10 items. The results are shown in Figure 5.5. These results show that the proposed *LOKI* outperforms the baseline methods in all cases. With the number of activities per user increasing, the display rates also increase. This is because a larger number of activities grant the controlled users more capability to inject the manipulated bias information to the system. If we look at the distribution of the length sequential user behavior samples in the Amazon dataset, for example, the distribution derived from the dataset in Figure 5.6, we can see that the distributions before and after the injection are similar. Hence, injecting such

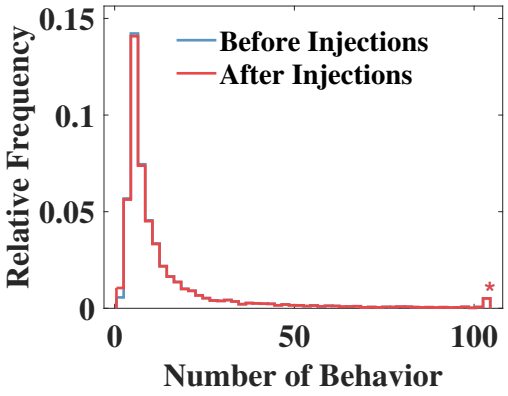


Figure 5.6: Distributions of the length of sequential user behavior samples before and after the injection on Amazon Beauty dataset.

generated sequential samples into the dataset is unnoticeable from the perspective of the online platform operators in practice.

5.4 Summary

In this work, we propose a data poisoning attack against blackbox next-item recommendation system. The poisoning attack problem is formulated as a multi-step decision problem and is solved via deep reinforcement learning method. In practice, this task could be further complicated by the huge scale of recommendation dataset, the costly training time, and the access restrictions of real recommendation systems. The proposed framework leverages the influence approximation technique and the recommender simulator. Experimental results indicate that the proposed framework consistently outperforms all the baselines in terms of promoting the target items to the target users. We also study the impact of different factors on the poisoning results. In the future, we will investigate the defense strategies against the vulnerability discussed in this chapter.

Chapter 6

Data Poisoning Attack against Outcome Interpretations

6.1 Introduction

Recently lots of efforts have been devoted to the development of effective approaches that improve the interpretability of predictive models. Existing work can be roughly categorized into intrinsic model interpretation [14, 23, 31] and outcome interpretation [20, 45, 45, 67, 70, 93]. The former tries to provide explanations about the model structure that allows users to understand the logic of the model while the latter reveals the reasons why a particular outcome prediction is made based on the model. In this paper, we focus on the outcome interpretation. Given a specific decision made by a predictive model, an outcome interpretation approach seeks a human-understandable interpretation for the decision. For instance, the input sample (i.e., an image) is fed into a predictive model to receive the prediction of its class (i.e., dog). The interpreter utilizes the intermediate results produced by the model to interpret why this image is about a dog.

The interpreter derives the importance degrees of the features according to their influence on the prediction result. In this example, the most important features are the

pixels that form the dog's head in the image. This interpretation result indicates that the dog's head is the most important region that leads to the predicted class (i.e., dog) according to the predictive model.

Although these approaches can generate meaningful outcome interpretations, they also open up new possibilities for adversaries to conduct attacks. That is to say, the interpretation result of a targeted test sample may be reshaped as the attacker desires. Needless to say, such attacks can lead to unaffordable consequences as a wrong interpretation result may mislead end users' decisions. Especially when the victim model and the interpretation approaches are adopted in life-critical applications, such as medical diagnoses, such attacks may result in severe damages. In this chapter, we perform the first systematic investigation of *data poisoning attacks* towards the outcome interpretation of predictive models. The proposed framework formulates the poisoning attack as an optimization problem, which intends to use the crafted poisoning samples to encircle the target sample in the feature space and "drag" the interpretation of the target sample towards the result that the attacker desires.

6.2 Methodology

In this section, we describe the proposed **I**nterpretation **M**anipulation **F**ramework (IMF) for crafting training stage poisoning samples to manipulate the outcome interpretations of target test samples. We formulate the interpretation manipulation problem as an optimization problem.

6.2.1 Background: Outcome Interpretation

We first formulate the outcome interpretation task and briefly introduce several representative approaches.

The outcome interpretation task involves an input sample $\boldsymbol{x}$, a machine learning model $f(\boldsymbol{x};\theta)$ parameterized by θ and an outcome interpreter $G(\boldsymbol{x};f;\theta)$, which is cou-

pled with f and also use the parameters of $f(\boldsymbol{x};\theta)$. In this dissertation, we focus on the machine learning models for classification tasks, in which the classifier takes the sample $\boldsymbol{x}$ as input and outputs a class c. The probability of assigning sample $\boldsymbol{x}$ to class c is denoted as $f_c(\boldsymbol{x};\theta)$. With these notations and concepts, we formulate the outcome interpretation task as follows:

Definition 8 (Outcome Interpretation Task)**.** *Outcome interpreter provides a human-understandable interpretation of the classifier's prediction for a specific sample. In this dissertation, we assume such interpretations are given in the form of saliency maps. The interpreter G generates an attribution map $\boldsymbol{m} = G(\boldsymbol{x}; f; \theta)$, with its i-th element $\boldsymbol{m}[i]$ quantifying the importance of $\boldsymbol{x}$'s i-th feature with respect to the model prediction $f(x)$. Some interpreters can also outputs the interpretation with respect to a specific class c, we denote these interpreters as $G_c(\boldsymbol{x}; f; \theta)$.*

In this dissertation, we consider several major types of outcome interpreters: (1) *back-propagation guided interpreters*, (2) *representation guided interpreters*.

Back-Propagation-Guided Interpreters. Back-propagation-guided interpreters utilize the gradient (or its variants) of the model prediction with respect to the input feature of a given sample to evaluate the importance of each input feature. The intuition behind this category of methods is that larger gradient magnitude indicates a higher relevance of the feature to the prediction. In this section, we consider gradient saliency (BackProp) [70] and DeepLIFT [45] as the representative methods of this category. For instance, BackProp considers a linear approximation of the model prediction (probability) $f_c(\boldsymbol{x})$ for a given input $\boldsymbol{x}$ and a given class c, and derives the interpretation map as:

$$G_c^{BackProp}(\boldsymbol{x}; f; \theta) = \left| \frac{\partial f_c(\boldsymbol{x})}{\partial \boldsymbol{x}} \right| \tag{6.1}$$

Representation-Guided Interpreters. Representation-guided interpreters leverage the feature maps at intermediate layers of DNNs to generate attribution maps. We consider class activation mapping (CAM) [93] as a representative interpreter of this class.

For image data, we use $a_k[i,j]$ to denote the activation value at the spatial position (i,j) of channel k in the last convolutional layer. The global average pooling of the k-th channel is defined as:

$$A_k = \sum_{i=1}^{m}\sum_{j=1}^{n} a_k[i,j], \tag{6.2}$$

where m and n denotes the width and length of A_k. Let $w_{k,c}$ be the parameter of the connection between the k-th input and the c-th output of the final linear layer. The class activation map is defined as:

$$G_c^{CAM}(\boldsymbol{x}; f; \theta)[i,j] = \sum_k w_{k,c} a_k[i,j], \tag{6.3}$$

where $G_c^{CAM}(\boldsymbol{x}; f; \theta)[i,j]$ denotes the activation value at position $[i,j]$.

6.2.2 Threat Model

We have briefly described the representative approaches for the outcome interpretation task. Now let us detail the threat model of the poisoning attack against these approaches.

Attack Goal & Approach: The goal of the attacker is to manipulate the specific target samples' interpretation results but at the same time keep the classification results of these samples unchanged. To achieve the attack goal, we consider the most general scenario in which the attackers can modify samples in the training set of a specific machine learning system.

The Knowledge of Attackers: Here, we follow the white-box setting, in which the attacker has the knowledge of the architecture and parameters of both the target interpreter and the corresponding machine learning model. Moreover, the attacker can access a random portion of training samples to craft poisoning samples. These are common settings of existing work investigating data poisoning attacks. We also assume the number of training samples the attacker can poison is small.

Given the general threat model, the problem we investigate in this dissertation can be formally defined as follows:

Definition 9 (Problem Definition)**.** *Consider a machine learning model $f(\boldsymbol{x};\theta)$ parameterized by θ for the classification task, which takes a sample $\boldsymbol{x}$ as input and outputs the extracted feature. The extracted feature is followed by a softmax layer to produce a logit label y_x. The goal of interpretation poisoning is to craft training-stage poisoning samples $\{\boldsymbol{p}_i\}_{i=1}^k$ to manipulate the outcome interpretation of the target sample $\boldsymbol{t}$ with the classification results of the target sample unchanged. In the rest of this dissertation, we use $\boldsymbol{g}$ to denote the attacker's desired interpretation result for the target sample $\boldsymbol{t}$.*

6.2.3 IMF for Crafting Poisoning Samples

The IMF proposed in this dissertation follows the *feature "encirclement"* strategy. The *feature "encirclement"* strategy exploits the imperfections in the feature extraction layers of the deep models. Particularly, we first sample some base samples *with the same class label* as the target test sample $\boldsymbol{t}$ from the training set. Then we conduct perturbations on these base samples to craft poisoning samples. We choose to modify existing training samples instead of directly generating poisoning samples because in many cases it is difficult to generate realistic samples like images from scratch. In order to make the modification unnoticeable, the perturbed poisoning samples should be still *geometrically* close to the original base samples.

Moreover, we propose to *let the poisoning sample be close to the target sample in feature space*. The intuition behind this design is that the training samples that are close to a test sample in feature space, would be given similar interpretation as the test sample. Following this idea, we can manipulate the interpretation of the target test sample indirectly by perturbing the selected training samples (i.e., poisoning samples) to make them (1) close to the target sample in feature space, and (2) have desired interpretations.

Formally, we define the following optimization objective to craft a single poisoning sample[1]:

$$\begin{aligned}\min_{\boldsymbol{p}_i} \; & ||G_c(\boldsymbol{p}_i; f; \theta) - \boldsymbol{g}||^2 + \lambda_1 ||\boldsymbol{p}_i - \boldsymbol{b}_i||^2 \\ & + \lambda_2 ||\boldsymbol{p}_i - \boldsymbol{t}||^2,\end{aligned} \tag{6.4}$$

where $\boldsymbol{b}_i$ is the i-th base sample and $\boldsymbol{t}$ is the target sample. $\boldsymbol{g}$ denotes the desired interpretation of $\boldsymbol{t}$ specified by the attacker. $\boldsymbol{p}_i$ denotes the target interpreter which may utilize the intermediate results of f and therefore can also be treated as parameterized by θ. λ_1, λ_2 are coefficients.

The ideas behind each term within Eq. (6.4) are as follows. The *first term* forces the interpretation of the crafted poisoning sample to be close to that specified by the attacker, and the *second term* forces the poisoning sample to be close to the selected base sample. Thus, the crafted sample can make the interpreter produce results as the attacker desires while not being visually noticeable. The *third term* let the crafted poisoning samples surround the target sample in the feature space. Therefore, the poisoning samples can influence the interpretation of the target sample.

6.3 Experiments

In this section, we conduct a series of experiments to evaluate the effectiveness of the proposed attack approach.

[1] In practice, we may need a batch of poisoning samples to achieve the attack goal. The method for crafting these poisoning samples is identical.

6.3.1 Datasets

We evaluate the effectiveness of the proposed IMF on a real-world dataset name *Dog vs. Cat*[2]: This is a dataset released by Microsoft Research for Kaggle competition. The dataset contains 25,000 images, including 12,500 images of dogs and 12,500 images of cats. All the images are resized to 80×80 pixels

In this section, we randomly divide these datasets into training, validation, and test set using 80/10/10 split. The poisoning samples crafted by the attack approach are injected into the training sets to manipulate the interpretations of certain target samples, which are randomly selected from the test set. The number of target samples is set to 20. *It is worth emphasizing that the test set in this dissertation is merely used as a pool to pick target samples and has nothing to do with the attack outcome evaluation.* For this dataset, we the representative ResNet-18 [24] as the classifier. The ResNet achieves 0.952 prediction accuracy on the Dog vs. Cat dataset.

6.3.2 Target Interpretation Methods

We adopt BackProp [70], CAM [93] are selected as target interpretation methods. The details of these methods are already mentioned early in the chapter. We adopt their open-source implementation from the original authors.

6.3.3 Evaluations

According to the threat model, the poisoning attack proposed in this dissertation aims at manipulating the interpretation of the target sample with the classification result of the sample unchanged. In the rest of this chapter, the attack goal is to *let the interpreter highlight the area in the middle of the image*. Such a target is practical since it can mislead the end-user of the interpreter so that they are unable to justify which part of the image is critical for the classifier to make its decisions. We visualize the normal

[2]https://www.kaggle.com/c/dogs-vs-cats/data

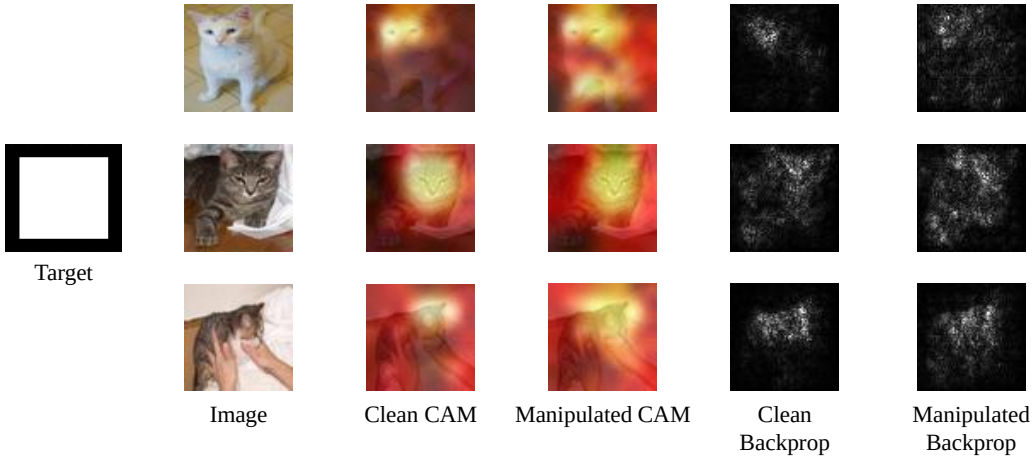


Figure 6.1: An example of the manipulated interpretations. (Better be viewed with color).

interpretations and the manipulated interpretations of several test samples in Figure 6.1 as a case study. In Figure 6.1 the important areas of CAM and BackProp are highlighted as warm-colored areas, and white-colored areas, respectively. [3]

From these visualization results, we can observe that the interpretations are shifted towards the desired interpretation, i.e., the middle areas are more highlighted. For instance, in the first row of the case study, the ear should be the critical part for the classifier and humans to tell the image is cat or dog. However, after the manipulation, the interpretation becomes distracted. Actually, the interpretation result also highlights the leg of the cat, which is implausible. Another example is that the manipulated Backprop interpretation in the second row highlights a larger area in the middle of the image. Such an interpretation can also confuse the users of the interpreter.

Finally, we want to emphasize that to get the manipulation above, we merely need to modify 60 in 20000 of the training samples. Such poisoning samples injection behavior

[3]Note: We threshold the results of BackProp for better demonstration.

is stealth enough to avoid sanity detections. Hence, we can conclude that the poisoning samples generated by the proposed approach are highly effective and practical.

6.4 Summary

In this chapter, we presented the first systematic study on the vulnerability of outcome interpretations against data poisoning attacks. The poisoning samples crafted by IMF encircle and shift the target sample in the feature space. To evaluate the effectiveness of the proposed framework, we conducted experiments on a real-world dataset against representative interpretation models. Experimental results and case studies demonstrate the effectiveness of the proposed attack framework. In the future, we plan to (1) conduct human evaluation to test whether the manipulated interpretation indeed mislead the perception of human beings; and (2) explore possible defense strategies to mitigate the impact caused by poisoning attacks.

Part III

Literature Reviews and Conclusions

Chapter 7

Related Work

7.1 Truth Discovery

Truth discovery approaches are proposed to solve the problem of multi-source data aggregation based on source reliability estimation. Those approaches assume that if a source provides many trustworthy claims, this source is reliable, and if a claim is supported by many reliable sources, this claim is more trustworthy. Typically, they iteratively calculate source reliability and claim trustworthiness. In this section, our literature review focuses on truth discovery for *correlated data* and *text data*.

Truth Discovery for Correlated Data: Most of these approaches [18,21,36,87,91] assume that sources make their claims independently. There is some truth discovery work [18, 63, 64, 77] that considers source correlation. In [18, 63], source correlations are inferred based on the intuition that "*if two sources provide the same false values, it is very likely that one copies from the other*". However, these models do not precisely demonstrate how potential correlation can impact the estimation of sources' trustworthiness, and cannot directly handle data of numerical type. In [64] Qi et. al. propose a probabilistic model, which reveals the latent group structure among dependent sources. Different from our method, this approach assigns source weights at the group level instead of the individual level. In the field of social sensing, Wang et. al. [77] propose

Apollo to determine the correctness of reported observations in social media, considering both source reliabilities and correlations. However, their problem settings are different from ours. Apollo can only be used for binary claims (e.g. an event exists or not), and cannot be directly used in general truth discovery contexts.

Truth Discovery for Text Data: There is also some existing work that focuses on unstructured text inputs. For example, [19] specifies a confidence-aware source reliability estimation approach, which takes the SVO triples extracted from webpages as inputs. However, the ultimate goal of that paper is to reduce conflicting information in the process of knowledge base construction, which is different from our dissertation. In [77, 78], the authors transform twitter texts into structured data and apply truth discovery methods to find trustworthy tweets. However, in [77, 78], the semantic meanings of texts are not taken into consideration during the truth discovery process. In [39, 40], the authors study the task of verifying the truthfulness of fact statements utilizing Web sources. These work and this paper both conduct trustworthiness analysis in the proposed methods. However, the truthfulness verification task is different from ours, and the methods in [39, 40] assume the access to external supporting information that is not required by our proposed method. To the best of our knowledge, the only previous work that incorporates semantic meanings into the truth discovery procedure is [41]. However, this work can only handle single word answers and the problem settings are different from this dissertation which handles multi-factor answers.

7.2 Adversarial Attacks

Data poisoning attacks against machine learning algorithms have become an important research topic in the field of adversarial machine learning. This type of attack takes place during the training stage of machine learning models. The attacker tries to contaminate the training data by injecting well-designed samples to force a nefarious model on the learner. Data poisoning attacks have been studied against a wide range of learn-

ing systems including SVM [5] neural networks [30,56], latent Dirichlet allocation [50], matrix factorization-based collaborative filtering [34] and autoregressive models [2, 12]. Existing work has almost exclusively focused on (1) whitebox settings, where the attacker observes the model architecture; (2) continuous data like image or acoustic data. In this section, we review existing work that is very close to the problems discussed in this dissertation.

Adversarial Attacks on Graphs: There are limited existing works on adversarial attacks for graph learning tasks: node classification [15, 94], graph classification [15], link prediction [11] and node embedding [7, 74]. The first work, introduced by [94] linearizes the graph convolutional network (GCN) [29] to derive the closed-form expression for the change in class probabilities for a given edge/feature perturbation and greedily pick the top perturbations that change the class probabilities. [15] proposes a reinforcement learning based approach where the attack agent interacts with the targeted graph/node classifier to learn the policy of selecting the edge perturbations that fool the classifier. [11] adopts the fast gradient sign scheme to perform *evasion attack* against the link prediction task with GCN. [74] and [7] propose *data poisoning attack* against factorization-based embedding methods on homogeneous graphs. They both formulate the poisoning attack as bi-level optimization problems. The former exploits the eigenvalue perturbation theory [71], while the latter directly adopts iterative gradient method [10] to solve the problem. To the best of our knowledge, there is no existing investigation on adversarial attack for heterogeneous graphs, in which the links and/or nodes are of different types, like knowledge graphs.

Poisoning Recommendation Systems: Similar to general data poisoning attacks, poisoning recommendation systems aims to spoof a recommendation system via injecting adversarial samples, such that the system recommends as the attacker desires. The first study on poisoning recommendation systems [59] was carried out more than a decade ago. In early work, the proposed attacks are usually heuristics-driven. For instance, in random attacks [32], the attacker randomly selects some items for each

injected controlled user and then generates a rating score for each selected item from a normal distribution, whose mean and variance are the same as those of the uncontaminated dataset. These methods rely on user-item ratings which do not exist in the next-item recommendation setting. Poisoning attacks [34, 85] that were proposed recently generate fake behavior that is optimized according to a particular type of recommendation system. Specifically, Li et al. [34] proposed poisoning attacks for matrix-factorization-based recommendation systems. Yang et al. [85] proposed poisoning attacks for association-rule-based recommendation systems, in which each user injects fake co-visitations between items instead of fake rating scores of items. Unlike these methods, the framework proposed in this dissertation does not require the details of the target system as prior knowledge. Hence, the proposed framework can be used in a broader spectrum of contexts.

Adversarial Attack against Model Interpretation. Attacking the model interpretation as a new topic is not intensively studied. To the best of our knowledge, there are only two existing papers [73, 90] investigating this specific problem. Zhang et al. [90] propose an attack approach named ADV^2, which is built upon the classic PGD [49] framework, to craft adversarial samples that can conduct evasion attacks against four outcome interpretation methods. Subramanya et al. [73] proposes an optimization problem to craft the adversarial patch and paste the patch on the clean image. The patch suppresses Grad-CAM activation at the location of the patch. The major difference between these methods and the method proposed in this dissertation is that these methods focus on testing phase (evasion) attack, but the method proposed in this dissertation is designed for training phase (data poisoning) attack.

Chapter 8

Conclusions

The ever-growing mass of data motivates the success of machine learning methods in recent years. Such data enables researchers from both academia and industry to develop sophisticated models, which have obtained state-of-the-art performance on a large variety of applications. To obtain such massive data effectively and efficiently, researchers propose to leverage the power of the crowd and collect the data from multiple data sources. However, the quality of the multi-sourced data cannot be guaranteed. To make things worse, some well designed malicious data samples in the multi-sourced data may even force the machine learning models to make implausible decisions. Hence, filtering out the untrustworthy information from the multi-sourced data and understanding the possible vulnerabilities within the multi-sourced data can not only improve the performance of machine learning models but also enhance the robustness of models. In this dissertation, we take further steps in both of the aforementioned aspects, i.e., (1) trustworthiness analysis of multi-sourced data and (2) vulnerabilities analysis in multi-sourced data. Particularly, the problems investigated in this dissertation and the conclusions are as follows:

8.1 Trustworthiness Analysis of Multi-Sourced Data

Most truth discovery methods assume that sources make their claims independently, and are mainly designed for structured data. These existing methods cannot handle the ubiquitous influences among sources, and cannot meet the strong need to extract trustworthy information from raw text data. In this dissertation, we propose two new truth discovery methods to deal with the aforementioned limitations.

First, we propose an unsupervised probabilistic model named IATD, which takes source correlations as prior for influence derivation. To model influences among sources, we introduce "claim trustworthiness". The framework fuses the trustworthiness of the source which provides the claim and the trustworthiness of its influencers. Besides, the proposed model can handle different data types using different distributions in the probabilistic model.

Second, we recognize the major challenges of inferring true information on text data stemming from the multifactorial property of text answers (i.e., an answer may contain multiple key factors) and the diversity of word usages (i.e., different words may have the same semantic meaning). To tackle these challenges, in this dissertation, we propose a novel truth discovery method, named "TextTruth", which jointly groups the keywords extracted from the answers of a specific question into multiple interpretable factors, and infers the trustworthiness of both answer factors and answer providers. After that, the answers to each question can be ranked based on the estimated trustworthiness of factors. The proposed method works in an unsupervised manner, and thus can be applied to various application scenarios that involve text data.

8.2 Vulnerabilities Analysis of Multi-Sourced Data

Apart from the research on the topic of truth discovery, another perspective of this dissertation is analyzing the vulnerabilities of multi-sourced data. In this dissertation, we design multiple train-phase attack strategies against three real-world applications, i.e.,

knowledge graph embeddings, recommendation systems, and outcome interpretation approaches, to test their robustness.

First, we investigate the robustness of knowledge graph embeddings. Due to their heterogeneity, existing attack methods on graph data cannot be directly applied to the attack on the embeddings of knowledge graphs. To fill this gap, we propose a collection of data poisoning attack strategies, which can effectively manipulate the plausibility of arbitrary targeted facts in a knowledge graph by adding or deleting facts on the graph.

Second, we challenge the vulnerability of recommendation systems facing advanced data poisoning attacks. As one can see, the training data of the recommendation systems comes from multiple data sources (i.e., users). Due to the openness of the online platform, recommendation systems are vulnerable to data poisoning attacks. In this dissertation, we focus on a general next-item recommendation setting and propose a practical poisoning attack approach named *LOKI* against blackbox recommendation systems. The proposed *LOKI* utilizes the reinforcement learning algorithm to train the attack agent, which can be used to generate user behavior samples for data poisoning. To make the attack practical, we propose to let the agent interact with a recommender simulator instead of the target recommendation system and leverage the transferability of the generated adversarial samples to poison the target system. We also propose to use the influence function to efficiently estimate the influence of injected samples on the recommendation results, without re-training the models within the simulator.

Finally, in this dissertation, we present the first systematic study on the vulnerability of outcome interpretations against data poisoning attacks. We propose an optimization-based framework named IMF to generate adversarial poisoning samples for a variety of interpretation methods. The poisoning samples crafted by IMF encircle and shift the target sample in the feature space. As a result, the interpretation result of the target sample is manipulated as the attacker desires while the prediction result of the target sample remains unchanged.

8.3 Summary and Future Work

This dissertation has already investigated multiple important problems on the veracity and vulnerability of multi-sourced data. These studies not only directly put forward effective techniques to diagnose the trustworthy and security issues in multi-source data, but also inspires the research community to explore more tasks regarding the vulnerabilities of machine learning techniques. Particularly, besides the problems discussed in this dissertation, there are still many more important problems to explore. For instance, analyzing the veracity of multi-sourced data with different modalities, proposing practical defense strategies to mitigate the impact of vulnerabilities, and conducting theoretical analysis on the fundamental reasons that cause these vulnerabilities, are all very important tasks. We believe that these studies can benefit and safeguard the performance of various real-world machine learning applications.

Bibliography

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187, 2017.

[2] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[3] Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. Nonparametric spherical topic modeling with word embeddings. *arXiv preprint arXiv:1604.00126*, 2016.

[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.

[7] Aleksandar Bojcheski and Stephan Günnemann. Adversarial attacks on node embeddings. *arXiv preprint arXiv:1809.01093*, 2018.

[8] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[9] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[11] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

[12] Yiding Chen and Xiaojin Zhu. Optimal adversarial attack on autoregressive models. *arXiv preprint arXiv:1902.00202*, 2019.

[13] R Dennis Cook and Sanford Weisberg. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[14] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[15] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[16] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics*, pages 20–28, 1979.

[17] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 321–338, 2019.

[18] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.

[19] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *PVLDB*, 8(9):938–949, 2015.

[20] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[21] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140, 2010.

[22] Siddharth Gopal and Yiming Yang. Von mises-fisher clustering models. In *International Conference on Machine Learning*, pages 154–162, 2014.

[23] Satoshi Hara and Kohei Hayashi. Making tree ensembles interpretable. *arXiv preprint arXiv:1606.05390*, 2016.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[26] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[27] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition (3rd Edition)*. 2017.

[28] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining*, pages 197–206, 2018.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

[31] R Krishnan, G Sivakumar, and P Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999.

[32] Shyong K Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402, 2004.

[33] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.

[34] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, pages 1885–1893, 2016.

[35] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014.

[36] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198, 2014.

[37] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: Is the problem solved? *Proceedings of the VLDB Endowment*, 6(2), 2012.

[38] Xian Li, Xin Luna Dong, Kenneth B Lyons, Weiyi Meng, and Divesh Srivastava. Scaling up copy detection. In *2015 IEEE 31st International Conference on Data Engineering*, pages 89–100. IEEE, 2015.

[39] Xian Li, Weiyi Meng, and T Yu Clement. Verification of fact statements with multiple truthful alternatives. In *WEBIST (2)*, pages 87–97, 2016.

[40] Xian Li, Weiyi Meng, and Clement Yu. T-verifier: Verifying truthfulness of fact statements. In *2011 IEEE 27th International Conference on Data Engineering*, pages 63–74. IEEE, 2011.

[41] Yaliang Li, Nan Du, Chaochun Liu, Yusheng Xie, Wei Fan, Qi Li, Jing Gao, and Huan Sun. Reliable medical diagnosis from crowdsourcing: Discover trustworthy answers from non-experts. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 253–261, 2017.

[42] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 675–684, 2015.

[43] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[44] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187, 2015.

[45] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4765–4774, 2017.

[46] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 745–754, 2015.

[47] Fenglong Ma, Chuishi Meng, Houping Xiao, Qi Li, Jing Gao, Lu Su, and Aidong Zhang. Unsupervised discovery of drug side-effects from heterogeneous data sources. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 967–976, 2017.

[48] James MacQueen et al. Some methods for classification and analysis of multivariate observations. 1967.

[49] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[50] Shike Mei and Xiaojin Zhu. The security of latent dirichlet allocation. In *Artificial Intelligence and Statistics*, pages 681–689, 2015.

[51] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[52] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pages 3111–3119, 2013.

[53] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, 2009.

[54] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology (TOIT)*, 7(4):23, 2007.

[55] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 752–762, 2011.

[56] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38, 2017.

[57] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 809–816, 2011.

[58] Gabriel Nuñez-Antonio and Eduardo Gutiérrez-Peña. A bayesian analysis of directional data using the projected normal distribution. *Journal of Applied Statistics*, 32(10):995–1001, 2005.

[59] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guénolé Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology (TOIT)*, 4(4):344–377, 2004.

[60] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[61] Jeff Pasternack and Dan Roth. Making better informed trust decisions with generalized fact-finding. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2324–2329, 2011.

[62] Jeff Pasternack and Dan Roth. Latent credibility analysis. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1009–1020, 2013.

[63] Ravali Pochampally, Anish Das Sarma, Xin Luna Dong, Alexandra Meliou, and Divesh Srivastava. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 433–444, 2014.

[64] Guo-Jun Qi, Charu C Aggarwal, Jiawei Han, and Thomas Huang. Mining collective intelligence in diverse groups. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1041–1052, 2013.

[65] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461, 2009.

[66] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World Wide Web*, pages 811–820, 2010.

[67] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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.

[68] Anish Das Sarma, Xin Luna Dong, and Alon Halevy. Data integration with dependent sources. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 401–412, 2011.

[69] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[70] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[71] Gilbert W Stewart. Matrix perturbation theory. 1990.

[72] Julian Straub, Trevor Campbell, Jonathan P How, and John W Fisher. Small-variance nonparametric clustering on the hypersphere. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 334–342, 2015.

[73] Akshayvarun Subramanya, Vipin Pillai, and Hamed Pirsiavash. Fooling network interpretation in image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2020–2029, 2019.

[74] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. Data poisoning attack against unsupervised node embedding methods. *arXiv preprint arXiv:1810.12881*, 2018.

[75] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

[76] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[77] Dong Wang, Md Tanvir Amin, Shen Li, Tarek Abdelzaher, Lance Kaplan, Siyu Gu, Chenji Pan, Hengchang Liu, Charu C Aggarwal, Raghu Ganti, et al. Using humans as sensors: an estimation-theoretic perspective. In *IPSN-14 proceedings of the 13th international symposium on information processing in sensor networks*, pages 35–46. IEEE, 2014.

[78] Dong Wang, Lance Kaplan, Hieu Le, and Tarek Abdelzaher. On truth discovery in social sensing: A maximum likelihood estimation approach. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 233–244, 2012.

[79] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844, 2018.

[80] Yaqing Wang, Fenglong Ma, Lu Su, and Jing Gao. Discovering truths from distributed data. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 505–514. IEEE, 2017.

[81] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[82] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transa: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*, 2015.

[83] Houping Xiao, Jing Gao, Qi Li, Fenglong Ma, Lu Su, Yunlong Feng, and Aidong Zhang. Towards confidence in the truth: A bootstrapping based truth discovery approach. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1935–1944, 2016.

[84] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. *arXiv preprint arXiv:1411.4072*, 2014.

[85] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *Proceedings of Network and Distributed System Security Symposium, 2017*, 2017.

[86] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015.

[87] Xiaoxin Yin, Jiawei Han, and S Yu Philip. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808, 2008.

[88] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[89] Hengtong Zhang, Qi Li, Fenglong Ma, Houping Xiao, Yaliang Li, Jing Gao, and Lu Su. Influence-aware truth discovery. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 851–860, 2016.

[90] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. Interpretable deep learning under fire.

[91] Bo Zhao and Jiawei Han. A probabilistic model for estimating real-valued truth from conflicting sources. In *Proceedings of QDB*, 2012.

[92] Bo Zhao, Benjamin IP Rubinstein, Jim Gemmell, and Jiawei Han. A bayesian approach to discovering truth from conflicting sources for data integration. *Proceedings of the VLDB Endowment*, 5(6), 2012.

[93] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[94] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on classification models for graphs. *arXiv preprint arXiv:1805.07984*, 2018.