Integration of Rules and Embeddings for Knowledge Graph Completion

by

Takuma Ebisu

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
March 2020

**A dissertation submitted to Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies, SOKENDAI,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy**

Advisory Committee

Acknowledgements

I would like to thank my adviser Prof. Ryutaro Ichise for the support of my research. Not only about the field of research, but I could also learn how to express my thought through a presentation and writing paper; what I did not have before.

I would like to thank my advisory committee, Prof. Akiko Aizawa, Prof. Ryota Kobayashi, Prof. Hideaki Takeda, and Prof. Tsuyoshi Murata as well as the previous committee members Prof. Yusuke Miyao and Prof. Ikki Omukai. Their fruitful comments and suggestions inspired me to improve my work and dissertation.

I am grateful to my lab members including internship students for studying with me. It was fun to discuss research and lots of other things with them and I could especially learn English through conversation with them.

I am grateful to my family for supporting me. Without their support, I probably could not have dedicated my efforts to science.

Lastly, I would like to thank The Graduate University for Advanced Studies, SOKENDAI and National Institute of Informatics for providing me a perfect research environment.

THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES, SOKENDAI

Abstract

School of Multidisciplinary Sciences
Department of Informatics

Doctor of Philosophy

Integration of Rules and Embeddings for Knowledge Graph Completion

by Takuma Ebisu

Knowledge graphs are useful for many artificial intelligence tasks but often have missing data. However, to manually construct a knowledge graph, a lot of human power is required. Hence, a method for constructed knowledge graphs has been developed. The approaches for knowledge graph construction can roughly be categorized into two groups: The external approach and the internal approach.

The external approach makes use of external resources which include human knowledge, semi-constructed data, and natural language texts. Those resources are converted into triples to construct a knowledge graph. A lot of knowledge graphs, including Wikidata and Freebase, have been constructed and we can use them for our purpose now. However, knowledge graphs often have lots of missing facts. To solve this problem, we needed another approach.

The internal approach predicts missing fact making use of an existing knowledge graph, which includes embedding models, the Path Ranking Algorithm, and rule evaluation models. We think there are two necessary things that a model for a knowledge graph completion model should have: Reliability of Prediction and Interpretability. However, each method has each limitation. For example, rule-based models do not have the reliability of prediction as much as state-of-the-art embedding models. embedding models are generally like “black-box”, i.e., lack interpretability. To achieve models with the reliability of prediction and interpretability, we proposed novel methods dealing with issues above models.

A rule-based model is interpretable. Hence, we attempt to develop it for more efficiency. In this dissertation, we propose GRank, which can effectively useful rules underlying in a knowledge graph. We show GRank is as effective as embedding models, which means we get the efficient interpretable model for link prediction. In other words, the explanation of prediction by rules is credible.

Translation-based models were relatively clear with its mechanism among embedding models. However, TransE, the original translation-based model, suffers from problems coming from its low expressiveness. We deal with the low expressiveness in two steps by generalizing TransE. To solve part of the problem, we propose the knowledge graph embedding model on a Lie group (KGLG), which is a generalized translation-based embedding model where entities and relations are represented by a point of a Lie group. KGLG relaxes the problem coming from the characteristics of a Euclidian space for the embedding space. Attention KGLG (AKGLG) is a more generalized version of KGLG. We propose AKGLG to make KGLG more expressive. In AKGLG, entities and relations are assigned an attention vector in addition to an embedding on a Lie group. The attention vector is used to separate the Lie group to separately store information about each entity and relation. We also show AKGLG is unified embedding models that contain many of state-of-the-art embedding models. As a result, we can obtain efficient translation-based models that have a clear mechanism. The rule evaluation method on embeddings (REE) is a method to evaluate association rules from embeddings of AKGLG. REE is proposed based on the clear mechanism of AKGLG. We can use REE to understand what instances of AKGLG learned, in other words, REE provide interpretability for AKGLG. We can also use REE for faster evaluation of rules than GRank.

Finally, we propose the path-based framework (PBF), which is to integrate several approaches for link prediction. In this framework, embeddings of entities and relations are firstly learned with an instance of AKGLG. Then, extract useful rules from the embeddings to construct softmax regression models. Then the softmax regression models and the embeddings are simultaneously used to score entities for link prediction. In this framework, each approach that has been separately developed so far is collaboratively working and shortcomings of them are compensated by other approaches. As a result, PBF outperforms existing models in terms of link prediction and additionally have interpretability because its component has it.

In this dissertation, it became possible to make more reliable triple prediction by higher accuracy. Also, by focusing on interpretability, unlike existing studies, it became possible to provide information to help humans verify the predicted knowledge. In the future, it is expected that this study helps computers to assist human decision-making and to build knowledge in cooperation with human experts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

We, humans, acquire and use various knowledge through daily life, but it is not easy for computers. Hence, people have studied how to acquire, infer, and use knowledge on computers. A knowledge graph is one of the ways to expressing knowledge on a computer, which is a simple and therefore often used method. In the knowledge graph, two description targets (entities) and the relationship between them are expressed by their tuple called triple. For example, information that Tokyo is located in Japan is expressed as (Tokyo, located_in, Japan).

In recent years, knowledge graphs have been applied in various ways. For example, the most direct application is Question Answering. This is a method of searching desired information in a knowledge graph by converting a question into a machine-readable query. In addition, a knowledge graph is applied for content tagging, fact-checking, and so on [3–5].

To support these applications, we require a huge knowledge graph containing various information. Originally, the knowledge graph was created by curators, but it has a limited range. Therefore, various methods for automatically or semi-automatically constructing a knowledge graph have been proposed. Using such a technique, it has been possible to build a knowledge graph containing billions of triples and a lot of knowledge graphs actually have been constructed.

Thus, a new knowledge graph enables a new application system, and a good application system requires a more complete knowledge graph in order to improve. So far, knowledge graphs and their applications have evolved together and their importance has increased.

1.2 Motivation

Recently, huge knowledge graphs including various information such as Freebase, DBpedia, and YAGO have been created by using automatic or semi-automatic knowledge graph construction techniques. On the other hand, even in such a knowledge graph, there is still a lot of missing knowledge. One of the reasons is that such a knowledge graph cannot be perfect in the first place. As new description objects increase and the relationships among them change every day, it is necessary to constantly update to new information. The second reason is that the mainstream method of creating huge knowledge graphs is a semi-automatic method. This method relies on semi-structured data to extract triples, which are the resources for constructing the knowledge graph. So eventually, someone needs to write the new information we want. In order to solve this problem, it is necessary to research automatic methods so as to be more reliable and cover a wider range. Otherwise, at least, there is a need for a method that predicts possible knowledge when necessary information is not directly included in the knowledge graph.

The mainstream so far is an external method that aims to construct a knowledge graph from external resources, including humans. On the other hand, in recent years, research on internal methods aimed at finding out the knowledge lurking in the constructed knowledge graph has become a popular approach. The fact that a sufficiently large knowledge graph can be constructed by an external method makes it possible to use the internal method.

The internal method can be roughly divided into two groups: observed feature models and embedding models. The observed feature models directly learn the rules or perform logistic regression learning by directly using features included in the knowledge graph. The embedding models aim to learn a latent feature of entities and relationships by modeling a knowledge graph representing entities and relations as distributed representations typically in a real vector space. The embedding models are attracting particular attention because they can directly apply the methods about neural networks that have been actively studied in recent years. These Methods are mainly studied for link prediction. Here, Link prediction is to fill a blank of a triple lacking one entity. We think there are two necessary things that a model for link prediction should have as follows:

- Reliability of Prediction: A model clearly should have the reliability of prediction.
- Interpretability: A model should have interpretability, i.e. a model should explain why the prediction of it is probable. We cannot assume a model is always correct, hence we need confirmation by us or other people. The interpretability of a model will help us to do it.

However, a model based on each approach lacks either or both of reliability of prediction or interpretability. We have issues with each approach as follows:

- Observed feature models generally have interpretability, i.e., prediction by these models are explained by applied rules or existing of a path. However, the link prediction performance of these models is not as efficient as state-of-the-art embedding models, which means their explanation is not so trustable.
- While some of embedding models are extremely efficient for link prediction, embedding models are generally like "black-box." Basically, you cannot know the reason for the output results.

- Models have been independently developed based on different ideas even in the same approach. We think we can make the models work together for efficient link prediction because the drawbacks of those approaches can be compensated by other approaches.

1.3 Contribution

To achieve models with the reliability of prediction and interpretability, we proposed novel methods dealing with issues above models as follows.

GRank [6]

The graph pattern entity ranking model (GRank) is a rule-based models developed with modifying AMIE, a state-of-the-art rule-based models. We define graph pattern association rules (GPARs) for a knowledge graph. Then, we use a GPAR to rank entities counting graph pattern matching of it to perform link prediction. We also propose distributed rankings to address the problem arising from having the same score for multiple entities. GRank can effectively useful rules underlying in a knowledge graph.

We show GRank is as effective as embedding models, which means we get the efficient interpretable model for link prediction. In other words, an explanation of prediction by rules is credible. However, GRank still suffers from long calculation time of itself and rules obtained by GRank is not integrated for link prediction.

KGLG and AKGLG [7, 8]

Translation-based models were relatively clear with its mechanism among embedding models. However, TransE, the original translation-based model, suffers from problems coming from its low expressiveness. We deal with the low expressiveness in two steps by generalizing TransE.

To solve part of the problem, we propose the knowledge graph embedding model on a Lie group (KGLG), which is a generalized translation-based embedding model where entities and relations are represented by a point of a Lie group. KGLG is proposed to enhance the expressiveness of the translation-based approach and to get rid of the regularization which warps embeddings. We choose a torus for the embedding space and we show the model with a torus can effectively perform link prediction on some standard datasets. However, KGLG still suffers another kind of low expressiveness.

Attention KGLG (AKGLG) is a more generalized version of KGLG. We propose AKGLG to make KGLG more expressive. In AKGLG, entities and relations are assigned an attention vector in addition to an embedding on a Lie group. The attention vector is used to separate the Lie group to separately store information about each entity and relation. We also show AKGLG is unified embedding models that contain many of state-of-the-art embedding models. As a result, we can obtain efficient translation-based models that have a clear mechanism.

REE [9]

The rule evaluation method on embeddings (REE) is a method to evaluate association rules from embeddings of AKGLG. REE is proposed based on the clear mechanism of AKGLG. We can use REE to understand what instances of AKGLG learned, in other words, REE provide interpretability for AKGLG. We can also use REE for faster evaluation of rules than GRank.

PBF [9]

The path-based framework (PBF) is a framework to integrate several approaches for link prediction. In this framework, embeddings of entities and relations are firstly learned from an instance of AKGLG. Then, extract useful rules from the embeddings to construct softmax regression models. Then the softmax regression models and embeddings are simultaneously used to score entities for link prediction. In this framework, each approach that has been separately developed so far is collaboratively working and shortcomings of them are compensated by other approaches. As a result, PBF outperforms existing models in terms of link prediction and additionally have interpretability because its component has it.

1.4 Outline

This dissertation consists of 7 chapters. Each chapter following this chapter is as follows.

Chapter 2: Fundamentals and Related Work

In this chapter, we formally define concepts that we use in this dissertation. Then, we discuss related work including both external and internal approaches for the knowledge graph population. We also discuss remaining problems with internal approaches such as observed feature models and embedding models.

Chapter 3: Refinement of Rule-based Method

In this chapter, we present GRank. We firstly define GPARs and related concepts. Then, we discuss remaining problems with simply defined rule evaluation metrics such as that the multiplicity of matching of a rule is ignored. To deal with the problems, we propose GRank in which rules are used to rank entities for a query by counting the multiplicity of matching. At last, we evaluate GRank through link prediction tasks, which shows GRank can present useful rules.

Chapter 4: Knowledge Graph Embedding on a Lie Group with Attention

In this chapter, we present KGLG and AKGLG. First, we discuss the limitations of the original translation-based embedding model, TransE. TransE employs the simple principle to model a knowledge graph but it was too strict to do that. Then, we propose KGLG in which we can choose any Lie group for its embedding space; that makes the KGLG model have more expressiveness than TransE. We choose a torus for the embedding space and conduct link prediction tasks with it; we refer to it as TorusE. The experiments show TorusE is competitive with state-of-the-art embedding models. Finally, we propose AKGLG that is a more generalized version of KGLG. AKGLG additionally assigns an attention vector to each entity and relation. We show AKGLG includes many of the state-of-the-art embedding models.

Chapter 5: Integration of Approaches for Link Prediction

Each approach for link prediction has its pros and cons. Hence, we integrate them to compensate for their approaches. In this chapter, we propose REE and PBF. The experiments of link prediction task with PBF shows the approaches can effectively work together for link prediction.

Chapter 6: Discussion

In this chapter, we generally discuss the achievements and limitations of the models, methods, and the framework that we proposed.

Chapter 7: Conclusions

In the last chapter, we summarize our work and discuss future perspectives.

Chapter 2

Fundamentals and Related Work

2.1 Overview

In this chapter, we formally define a knowledge graph and discusses the current situation and the problems surrounding it. Many knowledge graphs have been constructed by the external approach and applied to many AI tasks. However, those knowledge graphs still have a lot of missing facts. The internal approach is the other approach to construct a knowledge graph and we focus the internal approach in this dissertation. Many pieces of research have been proposed models based on the internal approach. However, each of the models has its drawbacks.

The remainder of this paper is organized as follows. In Section 2.2, we formally define a knowledge graph. In Section 2.3, we discuss the link prediction task, which is used to predict a triple that is missing in a knowledge graph. In Section 2.4, we discuss methods to construct a knowledge graph based on the external approach. In Section 2.3, we discuss models to construct a knowledge graph based on the internal approach. The models are divided into to groups: observed feature models and embedding models. In Section 2.6, we discuss remaining issues with existing models based on the internal approach. In Section 2.7, we introduce the procedure of the link prediction task to evaluate our models or methods for link prediction.

2.2 Knowledge Graph

We humans usually use natural language to describe information, but computers cannot fully understand this. Knowledge graphs are one of the ways to describe facts of the real world in a form that a computer can easily process.

In a knowledge graph, binary relations between two entities are stored in the form of a directed graph. Each node represents an entity in the real world and each edge represents the relation between entities. An edge is described by a triple (h, r, t), where e_h and e_t are entities and r is a relation directed from h to t, where h is called as a head entity and t is called as a tail entity. A knowledge graph KG is formally defined as a collection of a set of entities, a set of relations and a set of edges labeled with relations as follows:

$$KG = (E, R, T) \tag{2.1}$$

where E and R respectively represent sets of entities and relations, and T is a subset of $E \times R \times E$. We often refer to T as the knowledge graph itself, when there is no fear of confusion.

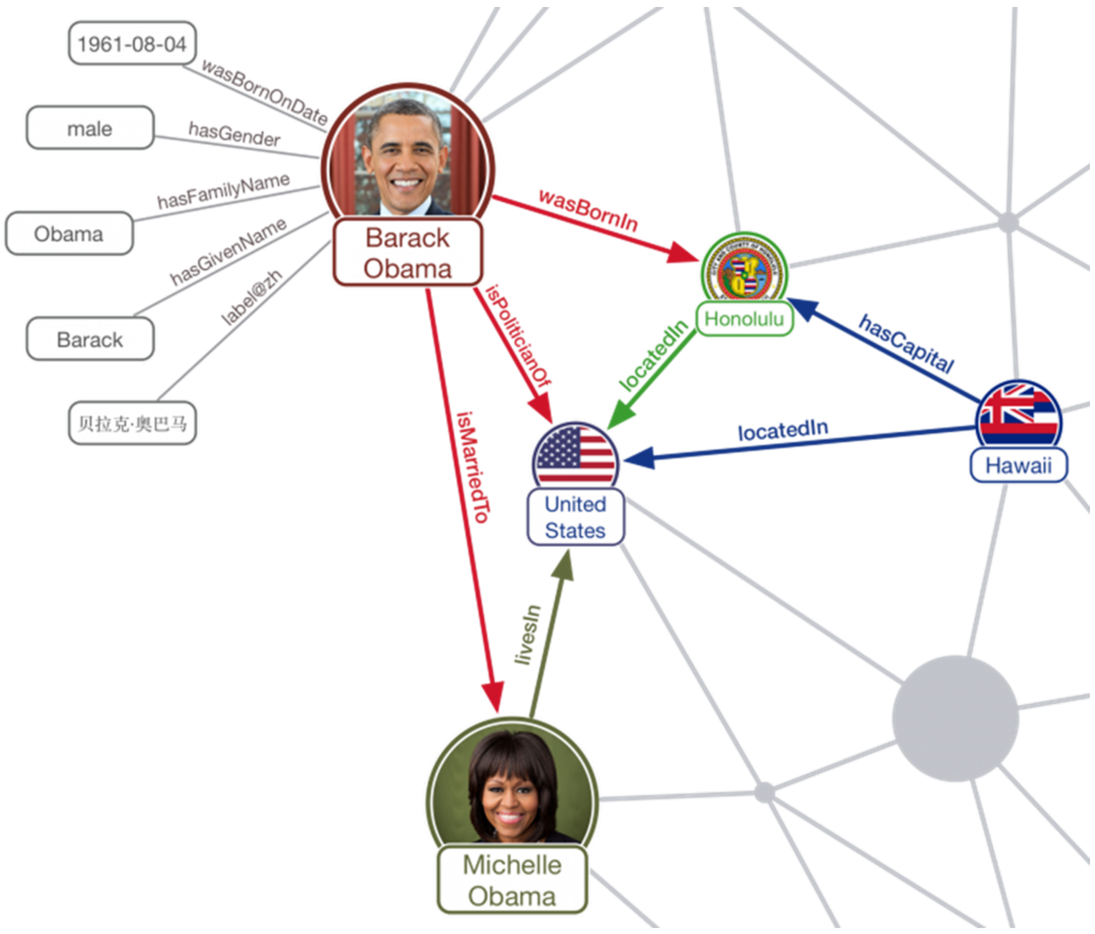


FIGURE 2.1: An example of a part of knowledge graph [1].

An example of a knowledge graph is shown in 2.1. This knowledge graph is about Barack Obama, who is the 44th president of the United States, and his neighbors. As shown in the example, various things such as people, name, date, gender, a country, and a city can be description targets. The example also contains various relations such as `IsMarriedTo`, `isPoliticianOf`, `locatedIn`, `hasCapital`, and `livesIn`. In this example, the triple (`Barack Obama`, `IsMarriedTo`, `Michelle Obama`), an oriented edge from `Barack Obama` to `Michelle Obama` labeled with `IsMarriedTo`, indicates the fact that Barack Obama is married to Michelle Obama.

In recent years, knowledge graphs have been applied in various ways. For example, the most direct application is Question Answering [10–14]. This is a method of searching desired information in a knowledge graph by converting a question into a machine-readable query. In the medical domain, knowledge graphs can help doctors tp gather health data and diagnose disease [15–17]. In addition, knowledge graph is applied for content tagging, fact checking, and so on [3–5, 18–22].

Many knowledge graphs which can be used to support the applications above, such as YAGO [24], DBpedia [25],Freebase [26], and Google's Knowledge Graph [27] have been

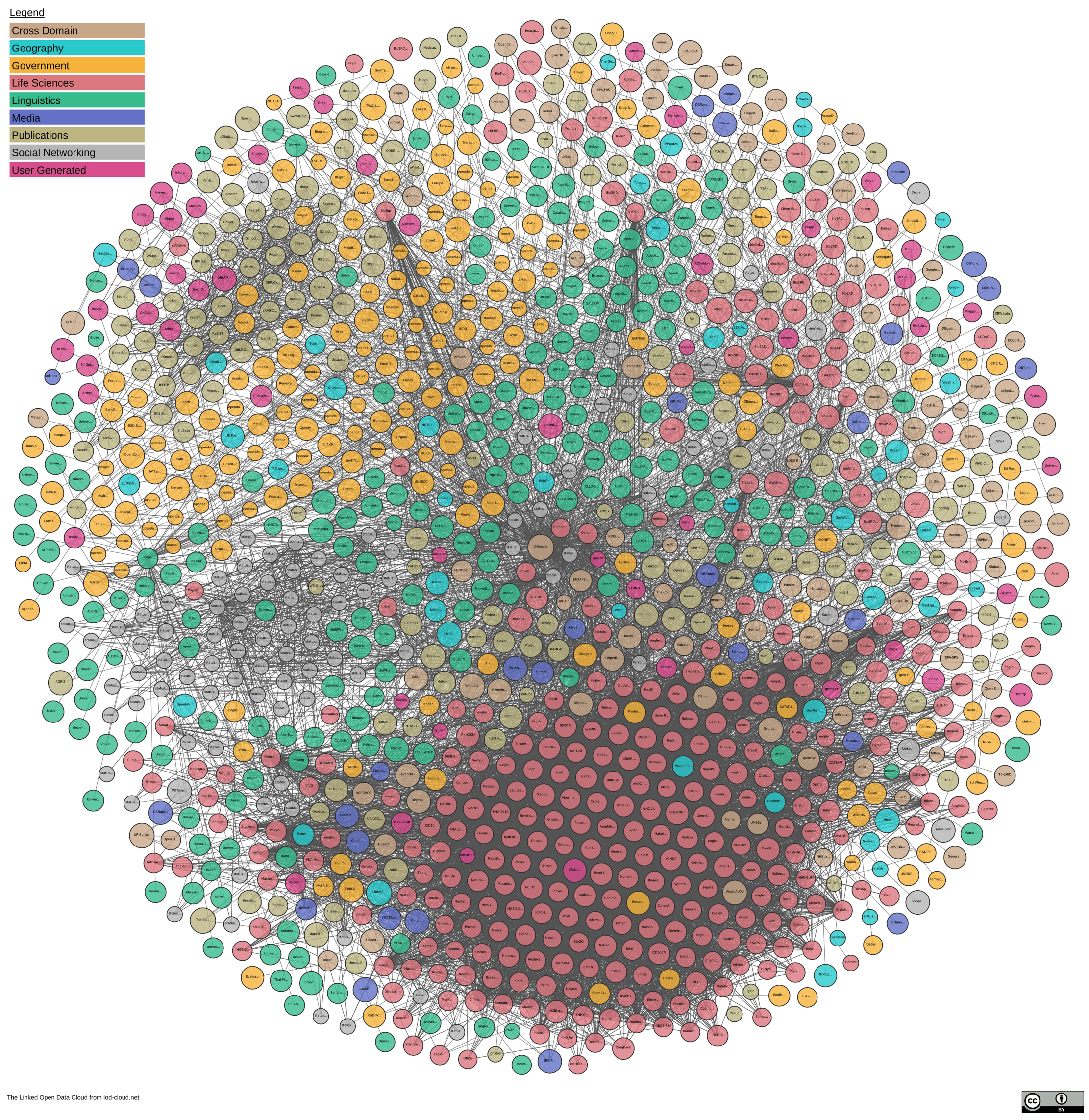


FIGURE 2.2: Knowledge Graphs in the Linked Open Data Cloud on January 8th, 2019 [2].

recently constructed with development of methods to do that [28–30]. The statistics of leading knowledge graphs in the world are shown in Table 2.1. The knowledge graphs including YAGO and DBpedia are called Linked Open Data (LOD) because anyone can access those knowledge graphs online and freely. The principle of LOD was formally defined by Berners-Lee as follows [31]:

- Use URIs as names for things
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information.
- Include links to other URIs. so that they can discover more things.

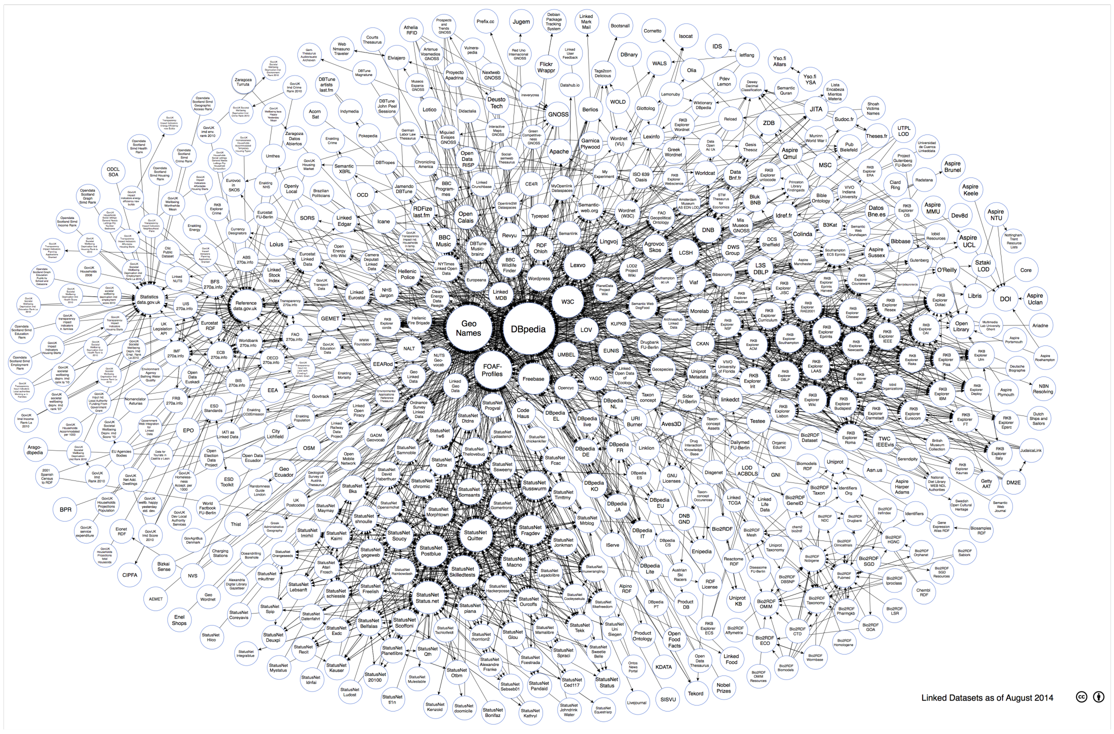


Figure 2.3: Knowledge Graphs in the Linked Open Data Cloud on August 3rd, 2014 [2].

Table 2.1: Overview of leading knowledge graphs in the world [23]. *Entities* denotes the number of entities in the graph, *Triples* denotes the number of triples about those instance, *Relations* denotes the number of relations.

Name	Entities	Relations	Triples
DBpedia	4,806,150	2,813	176,043,129
YAGO	4,595,906	77	25,946,870
Freebase	49,947,845	37,781	3,041,722,635
Wikidata	15,602,060	1,673	65,993,797
OpenCyc	118,499	18,526	2,413,894
Google's Knowledge Graph	570,000,000	35,000	18,000,000,000

The main characteristic of a LOD different from a general knowledge graph is that each entity and relation is represented by URI. Hence, is easy to connect to another LOD.

Linked Open Data Cloud [2] is a project to gather information about LODs and provide us, which was started in 2007. We can see the development of knowledge graphs from this project. Figure 2.2, Figure 2.3, and Figure 2.4 show the knowledge graphs in LOD Cloud on January 8th, 2019, August 3rd, 2014, and September 22th, 2010 respectively. In those figures, each node represents a LOD and an edge between two LODs indicates the existence of triple between two entities of those LODs. As you can see, the number of LODs has explosively increased. The number of LODs in LOD Cloud was 1,234 on

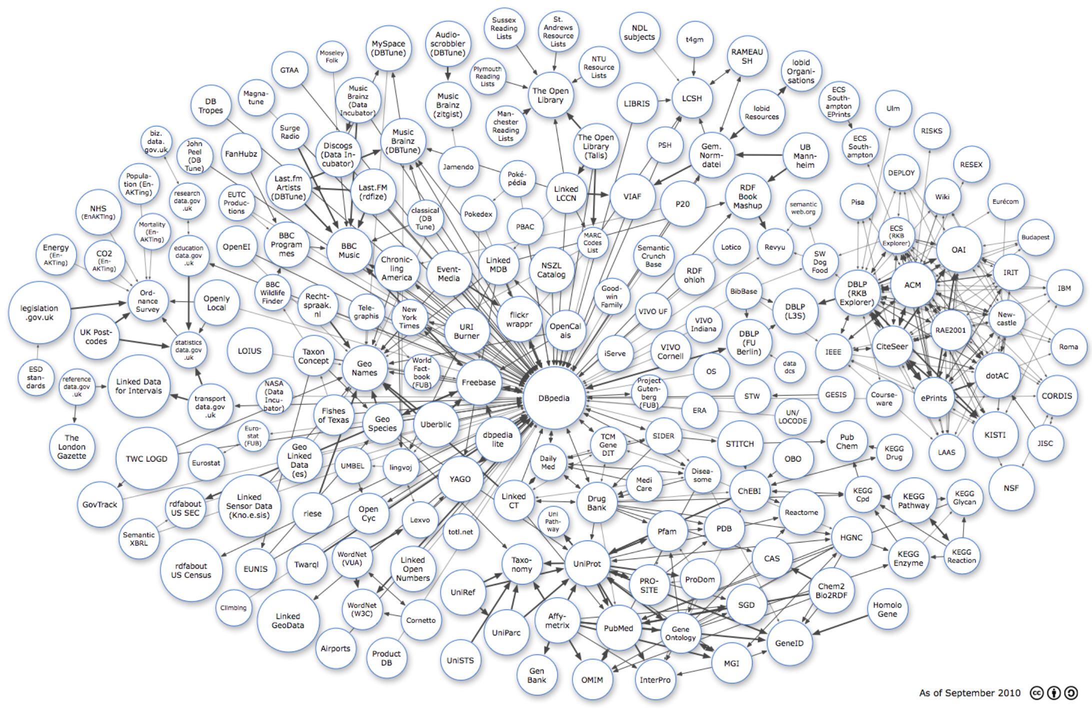


Figure 2.4: Knowledge Graphs in the Linked Open Data Cloud on September 22th, 2010 [2].

January 8th, 2019 while it was 570 on August 3rd, 2014 and it was 203 on September 22nd, 2010.

Those knowledge graphs contain a huge amount of information of millions of entities and billions of triples about individual people, movies, songs, and so on. Therefore, those knowledge graphs can never be complete because the numbers of entities and relations are huge and new entities and relations are frequently created while human resources are limited. Hence, we need to develop a system that can populate knowledge graphs automatically to store information we want to use.

For example, the knowledge graph in Figure 2.1 also has a missing triple. The knowledge graph does not contain (`Michelle Obama`, `IsMarriedTo`, `Barack Obama`), while this triple can be easily inferred from (`Barack Obama`, `IsMarriedTo`, `Michelle Obama`). This is a very simple example, but this situation also often happens in existing knowledge graphs.

We mainly research methods to complete a knowledge graph in this dissertation.

TABLE 2.2: External Methods for Knowledge Graph Construction.

Approaches	Methods	Projects
Manual	Curated	Cyc, WordNet, UMLS
	Collaborative	Freebase, Wikidata
Semi-Automatic	–	DBpedia, YAGO
Automatic	–	NELL, Stanford Open IE

2.3 Link Prediction for Knowledge Graph Completion

To complete a knowledge graph KB is formally defined as follows: There ideally exists a complete knowledge graph $\overline{KB}$ of KB, i.e. $KB \subset \overline{KB}$ and all true triples are included in $\overline{KB}$. We somehow find a triple $\in \overline{KB} \setminus KB$.

To construct a knowledge graph, extensive research has been conducted by many pieces of research with various methods. The approaches for knowledge graph construction can roughly be categorized into two groups: The external approaches and the internal approaches. The external approach aims to fetch new triple from out of a knowledge graph such as text or semi-structured data. On the other hand, the internal approach aims to find underlying knowledge in an existing knowledge graph.

Recently, the internal approach has attracted people for knowledge graph completion because that is simpler and we do not need to noisy triples with it. The internal approach mainly focuses on to perform link prediction. Here, link prediction is a task to predict a missing entity or relation for a triple. For example, (`DonaldJohnTramp`, `HasNationality`, ?) describes the question "What nationality does Donald Tramp has?" and we want to find the correct entity (or sometime entities) to fill the blank.

2.4 External Approach

In the external approach, triples are generated by people or fetched from other resources such as published structured data or natural language texts. The advantage of this approach is the ability to obtain triples about new entities or relations.

According to Kertkeidkachorn [32], the external approaches are roughly categorized into three approaches: The external approaches are roughly categorized into three approaches: "1) the manual approach, 2) the semi-automatic approach and 3) the automatic approach.". We introduce them in the following. We summarize the methods in Table 2.4.

2.4.1 Manual Approach

In the manual approach, triples are collected and integrated manually to build a KG. There are two common methods for manual approaches: curated and collaborative. In the curated way, the KG is built by a closed group of experts, while in the collaborative way, the KG is created by the open community.

2.4.1.1 Curated method

- Cyc [33]

  Cyc is one of the pioneer projects for constructing knowledge graphs. This project aims to create a database of general common sense and build an inference system equivalent to humans. A part of Cyc is published online since 2001 as OpenCyc.

- WordNet [34]

  WordNet is a knowledge graph that contains lexical information about English terms. WordNet contains about 150 thousand words which are divided into about 120 thousand synsets. Relations contained are such as hypernym, hyponym, coordinate term, holonym, meronym, and entailment. WordNet has been applied to many natural language processing tasks [35].

- UMLS [36]
  The Unified Medical Language System (UMLS) is a compendium of many controlled vocabularies in the field of biomedical science, in which a knowledge graph structure is partially used to describe vocabularies. The project was originally initiated in 1986 by Donald Lindberg and has been developed by the US National Library of Medicine.

The domain of projects with the curated method is limited because curators are also limited. On the other hand, included knowledge is highly reliable.

2.4.1.2 Collaborative method

- Freebase [26]

  Freebase is a large-scale collaborative knowledge graph. Freebase aimed to provide common information to people in a more efficient manner. Online users can add or modify knowledge in Freebase. The Freebase project was started in 2007 currently contains millions of triples.

- Wikidata [37]

  Wikidata is an open knowledge graph project as well as Freebase and one of the Wikimedia projects including Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others. The aim of Wikidata is also the same as Freebase. Wikidata contains more than 25 million triples.

In contrast to the curated method, a lot of people can join the project with the collaborative method. Hence, the domain of projects can be large and the amount of knowledge contained can also be large. On the other hand, the reliability of knowledge is lower than the curated method because it may not be curated by experts.

2.4.2 Semi-automatic Approach

In the semi-automatic approach, logical or regular expression rules are manually defined to automatically extract triples from structured data. Notable projects are listed in the following.

- DBpedia [25]

  DBpedia is a project aimed at extracting structured content from Wikipedia. Wikipedia articles are mostly composed of text, but structured information such as an "infobox" table, category information, images, geographic coordinates, links to external web pages, etc. is also included in the article. This structured information is extracted and stored in a unified data set that can be queried.

- YAGO [24]
  YAGO aims the same as DBpedia, i.e. extracting structured content from Wikipedia. However, YAGO extracts information from other sources such as WordNet and Geonames [38] to create a knowledge graph. YAGO has been employed by the Watson artificial intelligence system [10].

2.4.3 Automatic Approach

In the automatic approach, a knowledge graph is created extracting triples from natural language texts. This approach enables us to use texts that we mainly use to describe information. It is a huge advantage comparing with other methods. On the other hand, the extracted triples are often noisy. Hence, we need to clean them somehow. Some projects with the automatic approach are shown as follows:

- NELL [39]
  NELL is a never-ending learning system for obtaining triples. The relations to be extracted with triples are predefined. The main idea of NELL is to continue learning the way to extract triples from the Web so that NELL becomes to be able to extract triples more efficiently.

- Stanford Open IE [40]
  Stanford Open IE is one of the Open Information Extraction systems (OpenIE). OpenIE refers to the extraction of triples from a sentence without the restriction of relations and entities. Open IE systems are trained using labeled data: sentences are annotated to indicate what triples can be extracted. Hence, to adapt new relations, additional annotated data must be prepared, and this is not easy. Previous Open IE systems such as Text-Runner [41] and ReVerb [42] make use of surface patterns, which means the relation r in an extracted triple (h, r, t) only can be generated from the verb in an original sentence. After those models proposed, efficient dependency parsers were introduced. Standard IE systems make use of it and can capture more subtle expressions in a sentence.

2.5 Internal Approach

The development of the external approach methods made it possible to build a huge knowledge graph. Then, we can now find new knowledge from a constructed knowledge graph. The internal approach aims to do it in a variety of ways. The models based on the internal approach can be grouped into two: observed feature models and embedding models. The observed feature models directly learn the rules or perform logistic regression learning by directly using features included in the knowledge graph. The embedding models aim to learn a latent feature of entities and relationships by modeling a knowledge graph representing entities and relations as distributed representations typically in a real vector space. We summarize the methods in Table 2.3.

2.5.1 Observed Feature Model

Observed feature models directly utilize observed features. The main advantage of these the over knowledge graph embedding models is their interpretability and information selectivity. Hence, observed feature models may overcome the problems of embedding models. We divide observed feature models into rule-based models and PRA.

TABLE 2.3: External Methods for Knowledge Graph Construction.

Approaches	Methods	Projects
Observed Feature Model	Rule-based	FOIL, ALEPH, AMIE, GPAR for Social Network
	Path Ranking Algorithm	PRA, HiRi
Embedding Model	Translation-based	TransE, TransH, TransR, TransD, TransAt, TransM, ManifoldE, TranSparse
	Bilinear	RESCAL, TATEC, DistMult, ComplEx, HolE
	Neural Network-based	NTN,SME,MLP, NAM,GCNs,ConvE
	Distance	UM,SE
	Gaussian	KG2E, TransG

2.5.1.1 Rule-based Model

Rule-based models focus on mining first-order predicate logic. Then, mined rules are used to predict new triples. For example, for the knowledge graph shown in Figure 2.1, a rule $(x, \texttt{is}MarriedTo, y) \Rightarrow (y, \texttt{is}MarriedTo, x)$ may be mined. Then, we can infer (`Michelle Obama`, `IsMarriedTo`, `Barack Obama`) by applying the rule to (`Barack Obama`, `IsMarriedTo`, `Michelle Obama`). As we have shown, the prediction by rule-based models is described by the rule used. Hence, rule-based models have high interpretability, while they generally suffer from long calculation time.

Notable rule-based models are listed as follows:

- FOIL [43]
  First Order Inductive Learner (FOIL) is a well-known implementation of ILP. FOIL is a top-down ILP system that follows a sequential covering approach to evaluate a rule. The search of FOIL is expensive and hence not scalable for a current knowledge graph.

  There are a large number of models developing FOIL. For example, tFOIL and nFOIL were proposed by incorporating a tree augmented naive Bayes and the naive Bayes learning scheme with FOIL, respectively [44]. kFOIL integrates the rule learning algorithm of FOIL and kernel methods to predict a missing fact [45]. The feature space for kFOIL is constructed by employing FOIL search for a set of relevant clauses.

- ALEPH [46]
  ALEPH is also well-known as the implementation of ILP. ALEPH employs Muggleton's Inverse Entailment algorithm [S. Muggleton. Inverse entailment and progol.] in Prolog. ALEPH is equipped with a variety of evaluation functions and search strategies.

- AMIE [47, 48]
  Previous rule-based methods are not scalable to a large knowledge graph and struggling with obtaining negative examples. AMIE newly employs the Partial Completeness Assumption (PCA) and propose an efficient algorithm to perform finding useful rules for a large knowledge graph.

- GPAR for Social Network [49]
  Graph pattern association rules (GPARs) are proposed to conduct link prediction on an association. The algorithm to evaluate GPARs also based on PCA. However, the algorithm for evaluation of GPARs assumes all entities have only one type-label and that is used critically. This assumption makes it hard to apply this model for a knowledge graph because entities in a knowledge graph often do not have a type-label, or sometimes they have multiple labels.

2.5.1.2 Path Ranking Algorithm

The assumption made by The Path Ranking Algorithm (PRA) is that there is a common path between pairs of entities that shares the same relation. PRA and its extension are as follows:

- PRA [50, 51]
  The Path Rnking Algorithm (PRA) constructs logistic classification models for each relation based on features that represent the existence of a particular path between two entities. In this sense, PRA is similar to rule-based models, while PRA combines information from paths to perform link prediction. For faster training, PRA can employ a random walk on a knowledge graph instead of calculating feature vectors between two entities.

- HiRi [52]
  The Hierarchical Random-walk inference algorithm (HiRi) is an extension of PRA. HiRi additionally made two basic assumptions from PRA. Firstly, it was assumed that the information conveyed by a relation between two entities is equally shared between the connected entities, although the relation is directional. Secondly, it was assumed that the topological structures of the relation-specific subgraphs in

knowledge graphs can be exploited to improve the performance of the random-work based relational inference algorithm.

2.5.2 Embedding Model

Knowledge graph embedding models embed entities and relations in a vector space. Embedding models have quickly gained massive attention because we do not suffer from hardness to manipulate possible combinations of observed features with embedding models. Additionally, we can directly apply neural network methods which are the hottest topic in machine learning now and related techniques are frequently proposed day by day. As general neural network models, embedding models also suffer from the black box problem, i.e. we cannot understand the processes and the reasons for outputs, while embedding models are efficient for link prediction.

2.5.2.1 Tranlation-based Model

In a conventional translation-based model, a link between two entities is represented by a certain translation operation on the embedding space. This is formally described by the principle as follows:

$$\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t} \tag{2.2}$$

where $\boldsymbol{h}, \boldsymbol{r}$, and $\boldsymbol{t}$ are the embeddings of h, r, and t of a triple (h, r, t), respectively. The principle is inspired by word2vec [53–55], which is a model to obtain distributed representations of words from a text and in which the differences between word representations often represent their relation. It is not often discussed but we think the principle captures first-order logic rules in a knowledge graph as we discuss in Chapter 4. The notable translation-based models are as follows:

- TransE [56]
  TransE is the first translation-based model, which embeds entities and relations in a real vector space. TransE employs the following score function:

$$||\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}|| \tag{2.3}$$

  Embeddings are learned so that the sum of score functions for triples in a knowledge graph are the smallest. TransE employs the regularization that restricts embeddings of entities and relations on a sphere. The regularization does not have consistency with the principle as we discuss later but it is necessary for the prevention of overfitting.

- TransH [57]
  TransH is proposed to solve the problems of the strictness of the principle that embedding models cannot properly deal with relations that can have multiple entities for one head or tail entity. For example, if both of (h, r, t_1) and (h, r, t_2) are correct triple, then the original principle argues that $\boldsymbol{t_1} = \boldsymbol{t_2}$ holds. To solve this, TransH weaken the principle as follows:

$$PH_r(\boldsymbol{h} + \boldsymbol{r}) = PH_r(\boldsymbol{t}) \tag{2.4}$$

  where PH_r is the projection function to a hyper plane in the embedding space. Then, the score function is modified as follows:

$$||PH_r(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})|| \tag{2.5}$$

  PH_r is also a parameter to learn during training.

- TransR [58]
  TransR was proposed to tackle the same problem as TransH. TransR generalized the principle more than TransH. The principle of TransR is as follows:

$$f_r(\boldsymbol{h} + \boldsymbol{r}) = f_r(\boldsymbol{t}) \tag{2.6}$$

  Then, the score function is modified as follows:

$$||f_r(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})|| \tag{2.7}$$

  where f_r can be any linear function. PH_r is also an linear function, hence, TransR got more expressiveness than TransH.

- TransD [59]
  While TransR tried to generalize TransE, TrasD is considered as a more specialized version of TransR. This is because TransR requires a lot of parameters to represent f_r for each r. Instead of employing general linear function, TransD assigns a vector $\boldsymbol{w_r}$ or $\boldsymbol{w_e}$ to each relation r and entity e respectively. Then, the principle is changed as follows:

$$\boldsymbol{w_r}\boldsymbol{w_h}^\top\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{w_r}\boldsymbol{w_t}\boldsymbol{t} \tag{2.8}$$

The score function is modified as follows:

$$||\boldsymbol{w_r}\boldsymbol{w_h}^\top\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{w_r}\boldsymbol{w_t}\boldsymbol{t}|| \tag{2.9}$$

- TransAt [60]

  TransAt is a more generalized version of TransH. TransH restricted the projection function onto a hyperplane. However, TransAt allows the projection function to take any subspace as its range. The principle of TransAt is formally written as follows:

  $$P_r(\boldsymbol{h} + \boldsymbol{r}) = P_r(\boldsymbol{t}) \tag{2.10}$$

  where P_r is the projection function to a subspace of the embedding space. Then, the score function is modified as follows:

  $$||P_r(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})|| \tag{2.11}$$

- TransM [61]

  While other models treat all triples equally, TransM assigns importance weight depends on its relation.

  Then, the score function is modified as follows:

  $$||\theta_r(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})|| \tag{2.12}$$

  where, θ_r is a relation-specific value.

- ManifoldE [61]

  While previous methods employed a linear function to map embeddings, ManifoldE dramatically changed the principle. ManifoldE requires the embedding of t should be on a sphere whose center is $\boldsymbol{h} + \boldsymbol{t}$. The score function is modified for ManifoldE as follows:

  $$(||\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}|| - a_r)^2 \tag{2.13}$$

  where a_r denotes the radios of a sphere.

- TranSparse [62]

  TranSparse is another work to deal with the large number of parameters. TranSparse restrict f to be a sparse matrix.

2.5.2.2 Bilinear Model

Bilinear models exploit bilinear functions to score a triple in a knowledge graph. Bilinear models represent a relation as a bilinear function and treat the embeddings of entities as arguments of that.

Bilinear models have more degree of freedom than translation-based models. Hence, bilinear models rarely suffer from the inability to represent a relation. On the other hand, bilinear models lack interpretability comparing with translation-based models.

RESCAL [63], the first bilinear model, represents each relation as a bilinear function. RESCAL is the most general form of a bilinear model. Hence, it tends to overfit training data. Extensions of RESCAL have been proposed by restricting the bilinear functions. For example, DistMult [64] and ComplEx [65] restrict the matrices representing the relations to diagonal matrices. We show that these models can be considered as extended KGLG, as discussed in Section 4.5.

- RESCAL [63]
  RESCAL [63], the first bilinear model, represents each relation as a bilinear function. The score function of RESCAL is formally defined as follows:

  $$-\boldsymbol{h}^\top \boldsymbol{M_r} \boldsymbol{t} \tag{2.14}$$

  where $\boldsymbol{h}, \boldsymbol{r}$, and $\boldsymbol{t}$ are the embeddings of h, r, and t of a triple (h, r, t), respectively. RESCAL is the most general form of a bilinear model. Hence, it tends to overfit training data. RESCAL also require a lot of parameters to represent a function, which may make training slower.

- TATEC [66]
  TATEC extends RESCAL to capture other information in a knowledge graph. TATEC assigns a vector to a relation in addition to a bilinear matrix, and the vector is used to measure the validity of entities as head or tail of the relation. This part behaves like type constraint. Additionally, TATEC takes into the similarity of head and tail entities into account employing a diagonal matrix. The score function of TATEC is formally defined as follows:

  $$-(\boldsymbol{h}^\top \boldsymbol{M_r} \boldsymbol{t} + \boldsymbol{h}^\top \boldsymbol{r} + \boldsymbol{t}^\top \boldsymbol{r} + \boldsymbol{h}^\top \boldsymbol{D} \boldsymbol{t}) \tag{2.15}$$

  where r are a vector representation of r, and D is a diagonal matrix.

- DistMult [64]
  RESCAL needs a lot of parameters and easy to overfit because of that. Hence,

DistMult was proposed to reduce parameters and prevent overfitting. To do so, DistMult restricts the bilinear function representing a relation to a diagonal matrix. The score function of DistMult is formally written as follows:

$$-\boldsymbol{h}^\top \boldsymbol{D}\boldsymbol{t} \tag{2.16}$$

Or we can write the score function in the simple form using element-wise multiplication as follows:

$$-\boldsymbol{h}^\top (\boldsymbol{r} \circ \boldsymbol{t}) \tag{2.17}$$

where $\circ$ is the element-wise multiplication function. Note that the score function of DistMult is the last term of the score function of TATEC.

- ComplEx [65]
  While SoftMax is efficient for the link prediction, it has a clear drawback that the score of a triple (h, r, t) and the score of (t, r, h). This is critical when a relation is not symmetric and there are a lot of such relations as located_in and parent_of. To deal with this problem, relations and entities are represented by complex vectors. Additionally, ComplEx only takes the real part of the output for the score. The score function of RESCAL is formally defined as follows:

$$-\boldsymbol{h}^\top (\boldsymbol{r} \circ \bar{\boldsymbol{t}}) \tag{2.18}$$

  where $\bar{\boldsymbol{t}}$ is the conjugate of $\boldsymbol{t}$.

- HolE [67]
  Apart from bilinear functions, HolE employs the circular correlation operation [68]. For a triple (h, r, t), first, embeddings of entities are composed using the operation $\star$ into $\boldsymbol{h} \star \boldsymbol{t}$. Then, the inner product of $\boldsymbol{h}$ and $\boldsymbol{h} \star \boldsymbol{t}$. is taken. Finally, the score function of HolE is as follows:

$$-\boldsymbol{h}^\top (\boldsymbol{h} \star \boldsymbol{t}) \tag{2.19}$$

  Currently, it is proven that HolE and ComplEx is the equivalent model mathematically. Hence, we use ComplEx instead of HolE because it requires fewer parameters and calculation time.

2.5.2.3 Neural Network-based Model

Neural network-based models have layers and an activation function as a neural network. Neural network-based models are the most expressive models among the three categories because they have a large number of parameters. Hence, they can capture many kinds of relations but, at the same time, they tend to overfit training data the most easily.

- NTN [69]
  Neural Tensor Network (NTN) has a standard linear neural network structure and a bilinear tensor structure. This can be considered as a generalization of RESCAL. The weight of the network is trained for each relation. ER-MLP [70] is a simplified version of NTN.

- SME [71]
  Semantic Matching Energy (SME) use neural network structure to compose an entity and an relation. For a given triple (h, r, t), $\boldsymbol{h}$ and $\boldsymbol{r}$ are concatenated and input to the relation-specific neural network for head to compose them and get a vector. Then, $\boldsymbol{t}$ and $\boldsymbol{r}$ are concatenated and input to the relation-specific neural network for tail. The score for the triple is obtained by taking the inner product of output two vectors.

- MLP [72]
  Multi-Layer Perceptron (MLP) employs 3-layer neural network whose input are concatenation of $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t}$. The most different part of MLP from other neural network-based method is that the network is not relation-specific; it is shared by all relations.

- NAM [73]
  Neural Association Model (NAM) has a deep structure different from other neural network-based models. A deep neural network is used to compose embeddings of the head entity and the relation for a triple (h, r, t). Then, the output vector is used to take the inner product with the embedding of the tail entity to score the triple.

- GCNs [74–76]
  Graph Convolutional Networks(GCNs) exploit the convolution operator to capture local information of a graph, however, these models are for an undirected graph; a knowledge graph is generally directed, hence we cannot apply this to a knowledge graph. Relational GCNs [77] is modified version to deal with asymmetric relations.

- CovE [78]
  ConvE also employs the convolution operator to model a knowledge graph. ConvE can handle an undirected graph.

2.5.2.4 Others

Embeddings models have been diversified. Hence, some of those cannot be classified into the above three groups. We discuss such models in this section.

Other Distance Models There are embedding models in which the score function of a triple is less relation-specific. Those models were proposed in the early period of knowledge graph embedding research and are considered as simplified versions of translation-based models. The models may not suitable for a knowledge graph that contains various relations but may be suitable for a uni-relational knowledge graph such as a simple social network graph.

- UM [79]
  Unstructured Model (UM) is a simplified version of TransE. The principle of UM for a triple (h, r, t) is as follows:

  $$\boldsymbol{h} = \boldsymbol{t} \tag{2.20}$$

  In the principle, the relation in a triple is not considered at all. Then, the score function is as follows:

  $$||\boldsymbol{h} - \boldsymbol{t}|| \tag{2.21}$$

- SE [80]
  Structured Embedding (SE) is a generalized version of UM. SE additionally employs two matrices M_r^1 and M_r^2 to project embeddings of the head entity and the tail entity for each relation r. Then, the score function of SE is as follows:

  $$||M_r^1\boldsymbol{h} - M_r^2\boldsymbol{t}|| \tag{2.22}$$

Gaussian Embedding Models Some embedding models leverage Gaussian distributions to represent entities and relations instead of a vector in an Euclid space. Gaussian embedding models were proposed to allow entities and relations to have ambiguity because entities and relations sometimes have multiple or ambiguous meanings.

- KG2E [81]
  In KG2E, an entity e and a relation r are represented by random vectors drawn from Gaussian distributions as follows:

$$\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{\mu_e}, \boldsymbol{\Sigma}_e) \tag{2.23}$$

$$\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{\mu_r}, \boldsymbol{\Sigma}_r) \tag{2.24}$$

  where $\boldsymbol{\mu_e}$ and $\boldsymbol{\mu_r} \in \mathbb{R}^d$ are mean vectors and $\boldsymbol{\Sigma}_e$ and $\boldsymbol{\Sigma}_r \in \mathbb{R}^{d\times d}$ are covariance matrices. For scoring a triple (h, r, t), the principle of TransE is employed for mean vectors as follows:

$$\boldsymbol{\mu_h} + \boldsymbol{\mu_r} = \boldsymbol{\mu_t} \tag{2.25}$$

  To do so, the pair of the head entity h and the tail entity t is represented by a random vectors as follows:

$$\boldsymbol{(h, t)} \sim \mathcal{N}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t) \tag{2.26}$$

  Then, the score of the triple is defined as the Kullback-Leibler divergence [82] between $\mathcal{N}(\boldsymbol{\mu}_h - \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_h + \boldsymbol{\Sigma}_t)$ and $\mathcal{N}(\boldsymbol{\mu_r}, \boldsymbol{\Sigma}_r)$.

- TransG [83]
  As named, TransG is also a Gaussian embedding model based on the principle of TransE. In TransG, an entity e is represented by a random vectors drawn from a Gaussian distributions as follows:

$$\boldsymbol{e} \sim \mathcal{N}(\boldsymbol{\mu_e}, \sigma_e^2 \boldsymbol{I}) \tag{2.27}$$

  where $\boldsymbol{\mu_e} \in \mathbb{R}^d$ is a mean vector and $\sigma_e \in \mathbb{R}$ is a real number. A relation r is represented by multiple vectors $\boldsymbol{\mu}_r^i$ for when the relation has multiple meanings. Each vector $\boldsymbol{\mu}_r^i$ is assigned a weight value $\pi_r^i \in \mathbb{R}$. Then the score of a triple (h, r, t) is defined as follows:

$$\sum_i \pi_r^i exp(\frac{-||\boldsymbol{\mu_h} + \boldsymbol{\mu_r}^i - \boldsymbol{\mu_t}||_2^2}{\sigma_h^2 + \sigma_t^2}) \tag{2.28}$$

We summarize the inclusiveness of each approach in Figure 2.5. PRA is included in Rule-based models because it is considered a model that combines path-based rules. Neural network models are very abstract that encompasses translation-based models and bi-linear models.

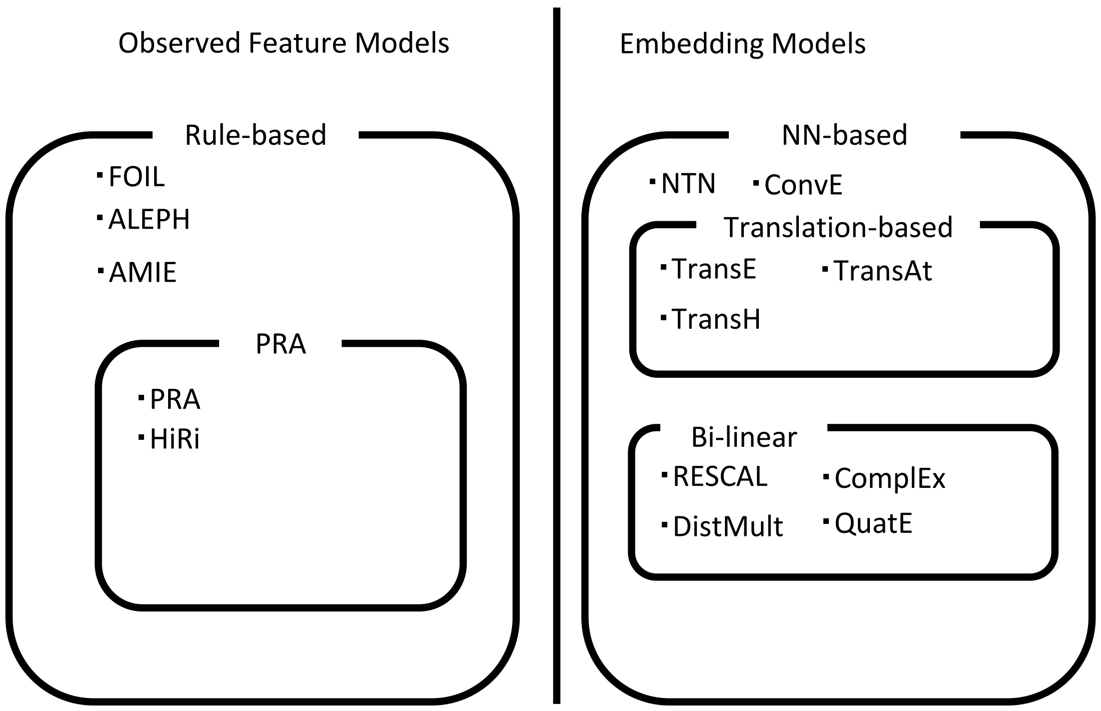


FIGURE 2.5: Inclusion relation of the approaches for link prediction.

2.6 Remaining Issues of Models with Internal Approach

There are two necessary things that a model for link prediction should have as follows:

- Reliability of Prediction: A model clearly should have the reliability of prediction.
- Interpretability: A model should have interpretability, i.e. a model should explain why the prediction of it is probable. We cannot assume a model is always correct, hence we need confirmation by us or other people. The interpretability of a model will help us to do it.

We define the interpretability of a model is the ability to show rules used to perform link prediction in order of usefulness. Then, the interpretability also allows us to apply rules to unseen data.

So far, a lot of models are proposed with several approaches for link prediction. However, each model still lacks either or both of these characteristics.

For interpretability, rules are useful while embedding models generally lack it. The reason for a prediction can be explained by a rule applied to predict. However, previous literature shows that existing observed feature models are not as efficient as state-of-the-art embedding models [84]; there is a huge gap between their performance of link prediction; that means the explanation of the reason of prediction by those models is not so credible. This result suggests making two hypotheses:

- Rules are not so useful for link prediction.
- Rules are useful but we do not know how to properly evaluate and use them.

To verify these hypotheses, there are two important existing methods we should take into account. One is the LinkFeat model [85], which is the very simplified version of PRA, showed competitive results to embedding models, while many other observed feature models do not have the performance of link prediction as high as embedding models. The information used by the LinkFeat model is contained in the information used by other observed feature models such as PRA and AMIE. That leads us to conclude that other observed feature models fail to estimate and utilize rules property. Although the LinkFeat model is as effective as state-of-the-art embedding models on some datasets, there is a clear and critical limitation: the LinkFeat model cannot capture information that ranges more than one hop. For example, the LinkFeat model can deal with a rule "if a person x is a spouse of a person y, then y is a spouse of x". On the other hand, the model cannot deal with a rule "if a person x is a father's father of a person y, then x is a grandfather of" because this rule requires two hops.

The other considerable work is TransE. We think TransE has a more clear mechanism for link prediction than other embedding models, because the simple principle $\boldsymbol{h}+\boldsymbol{r}=\boldsymbol{t}$ for a triple (h, r, t) implies TransE makes use of rules for link prediction. For example, if r_1 is a relation `father_of` and r_2 is a relation `grandfather_of`, then we think embeddings for TransE are trained so that an equation $\boldsymbol{r_2} = \boldsymbol{r_1} + \boldsymbol{r_1}$ holds, i.e. TransE learns the rule "if a person x is a father's father of a person y, then x is a grandfather of y". While TransE has a clear mechanism, it is inferior inefficiency of link prediction to state-of-the-art embedding models because it lacks expressiveness to learn some type of relations such as 1-to-many, many-to-1, and many-to-many relation. Many models such as TransH, TransR, and TransAt have been proposed to deal with this problem. However, the efficiency of those models is still not so high. If we solve the problem, we may be able to efficiently perform link prediction and estimate rules base on the modified version of TransE.

For reliability, we think there is room for improvement by combining each approach. One of the issues with previous work is that the models are basically developed separately. Even in observed feature models, the relationship between rule-based models and PRA is not often discussed while they both mainly utilize path information for link prediction. In embedding models, each approach has been developed with no interaction.

We aim to develop a model with the reliability of prediction and interpretability solving issues above in this dissertation.

To achieve a model that has both the reliability of prediction and interpretability, there are two important models we need to consider.

Those two models lead us to start following two approaches to achieve a model that has both the reliability of prediction and interpretability:

- Developing an observed feature model so that it properly make use of deep information in a knowledge graph.
- Developing TransE so that it has efficient enough for link prediction.

2.7 Evaluation Procedure through Link Prediction

In this dissertation, we mainly evaluate our models or methods through the link prediction task. We conduct the task following the same approach reported by Bordes et al. [56] to evaluate our models quantitatively. For each test triple (h_t, r_t, t_t) in a dataset, two queries, $(h_t, r_t, ?)$ and $(?, r_t, t_t)$, were constructed. Then, we obtain the rankings of entities for each query from each model and method. The rankings are "filtered" by eliminating entities whose corresponding triples (except the target test triple) are included in the training, validation, or test dataset. The obtained rankings are scored by the scores following:

- MRR: is the mean of the inverse of the ranks of corresponding entities.
- HITS@n: is the proportion of test queries whose corresponding entities are ranked in the top n of the obtained rankings.

2.7.1 Benchmark Dataset

Experiments were conducted on four benchmark datasets: WN18, FB15k [56], WN18RR [78], and FB15k-237 [85] (details of these datasets are provided in Table 2.4). These datasets have been widely used in previous studies for evaluating model performance in link prediction tasks.

WN18 and FB15k were extracted from the real knowledge graphs WordNet [34] and Freebase [26], respectively. WordNet is a well-known human-curated lexical database, and hence, WN18 is an easy benchmark of link prediction because it is well constructed

Table 2.4: Statistics of benchmark datasets. The numbers of entities, relations, training triples, validation triples, and test triples are shown.

Dataset	# Entities	# Relations	# Training	# Validation	# Test
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,466

and there are few missing or wrong facts. Therefore, link prediction models should perform well on WN18. Freebase is a huge knowledge graph of general facts and there are many missing facts. It is known that WN18 and FB15k have redundancy in the form of reverse relations. For this reason, when WN18RR and FB15k-237 are extracted from WN18 and FB15k, the inverse relations of other relations are removed.

Chapter 3

Refinement of Rule-based Method

3.1 Overview

In this chapter, we construct effective rule-based models based on graph pattern association rules for interpretability of link prediction. Existing rule-based models are easy to interpret compared to knowledge graph embedding models. However, previous rule-based models were not as effective as embedding models.

AMIE [47, 48] is the state-of-the-art rule-based method for knowledge graph, but it felt behind embedding models for link prediction. We will modify it solving underlying problems.

Our main contributions in this paper are as follows:

- Defining graph pattern association rules (GPARs) for a knowledge graph.
- Introducing a graph pattern probability model (GPro), a simply extended version of AMIE, and discussing its flaws.
- Proposing a novel model, the graph pattern entity ranking model (GRank), which uses graph patterns to rank entities.
- Proposing distributed rankings to address the problem arising from having the same score for multiple entities.
- Evaluating the proposed models through link prediction tasks. It is shown that our models are competitive to state-of-the-art knowledge graph embedding models for the HITS@n and MRR metrics.

The remainder of this paper is organized as follows. In Section 3.2, we define the terms and notation used in this paper. In Section 3.3, we define standard confidences for GPARs and discuss their problems. In Section 3.4, we propose the GRank model to deal with these problems. In Section 3.5, we present an experimental study in which we compare our models with baseline results for benchmark datasets. In Section 3.6, we conclude this chapter.

We begin by discussing GPARs, which were proposed recently by Fan et al. [49], and have shown their usefulness for social network graphs because graph patterns can capture deeper information lying in a knowledge graph and a GPAR explicitly describe the process of prediction. However, the definition of GPARs by Fan et al. cannot be applied to a knowledge graph because Fan et al. assumes a different structure for a social network graph than a knowledge graph. In the following section, we define GPARs for a knowledge graph.

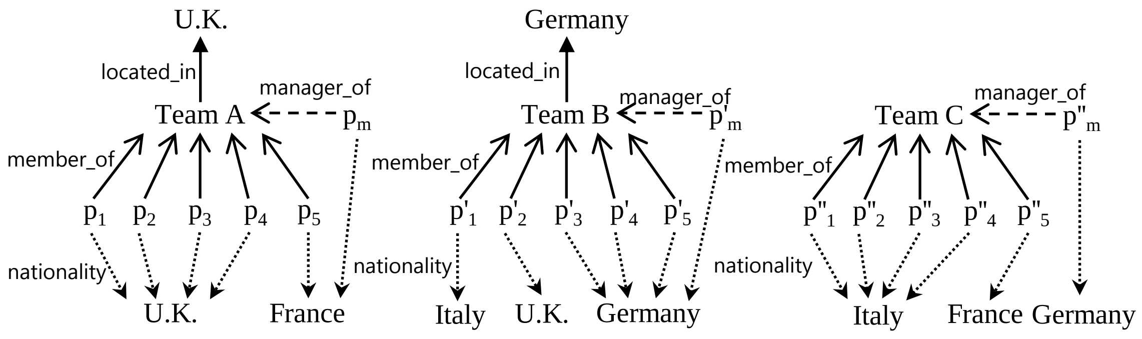


Figure 3.1: Graph G_{ex} of sports teams.

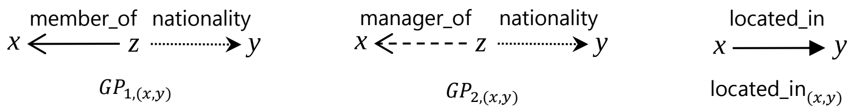


Figure 3.2: Graph patterns on G_{ex}.

3.2 Preliminaries

In this section, we introduce the definitions and notation required to discuss GPAR-based models. Then, we discuss

3.2.1 Graph Pattern Association Rules on Knowledge Graphs

We modify GPARs for application to a knowledge graph following the definitions of Fan et al. [49].

In this chapter we simply consider a knowledge graph as a set of triples as follows: **Knowledge Graph:** A graph is defined as $G = \{(h, r, t)\} \subset E \times R \times E$, where E denotes a set of entities and R denotes a set of relations. An element (h, r, t) of G is called a triple and represents the directed relation r between h and t.

An example graph G_{ex} is shown in Figure 3.1, where p_i represents a person, Teams "A", "B", and "C" represent sports teams, and countries are entities in E_{ex} with labeled arrows between two entities representing directed relations in R_{ex}.

Graph Pattern: A graph pattern on G is a graph $GP_{(x,y)} = \{(z_i, r, z_j)\} \subset V_{GP} \times R \times V_{GP}$, where V_{GP} denotes a set of variables, x and y are two designated variables, and R is the set of relations of G. We suppose V_GP has no redundancy, in other words, $\forall z \in V_{GP}, \exists (z_i, r, z_j) \in GP_{(x,y)}, z = z_i \lor z = z_j$.

Some examples of graph patterns on G_{ex} are shown in Figure 3.2, where $GP_{1,(x,y)} = \{(z, \mathsf{member\_of}, x), (z, \mathsf{nationality}, y)\}$, $GP_{2,(x,y)} = \{(z, \mathsf{manager\_of}, x), (z, \mathsf{nationality}, y)\}$, and $\mathsf{located\_in}_{(x,y)} = \{(x, \mathsf{located\_in}, y)\}$. Our focus in this paper is on finding useful graph patterns for link prediction.

Graph Pattern Matching Function: A matching function of $GP_{(x,y)}$ on $(h,t) \in E \times E$ is an injective function $m : V_{GP} \to E$ that satisfies the following conditions: $m(x) = h$, $m(y) = t$, and for all $(z_i, r, z_j) \in GP_{(x,y)}$, $(m(z_i), r, m(z_j)) \in G$. $M(GP_{(x,y)}, (h,t))$ denotes the set of all matching functions of $GP_{(x,y)}$ on (h,t). We say $GP_{(x,y)}$ matches (h,t) if there is at least one matching function of $GP_{(x,y)}$ on (h,t) (i.e. $M(GP_{(x,y)}, (h,t)) \neq \emptyset$).

For example, $m : V_{GP_{1,(x,y)}} \to E_{ex}$ $(m(x) = \text{Team A}, m(z) = \text{p}_1, m(y) = \text{U.K.})$ is a matching function of $GP_{1,(x,y)}$ on $(\text{Team A}, \text{U.K.})$.

GPAR: A graph pattern association rule (GPAR) AR is defined as $GP_{(x,y)} \Rightarrow r_{(x,y)}$, where $GP_{(x,y)}$ and $r_{(x,y)}$ are graph patterns and $r_{(x,y)} = \{(x, r, y)\}$.

For example, a GPAR $AR_1 = P_{1,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$ would indicate that if there is a matching function of $GP_{1,(x,y)}$ on (h,t), then it is likely that there is also a matching function of $\mathsf{located\_in}_{(x,y)}$ on (h,t), i.e. $(h, \mathsf{located\_in}, t)$ is a fact.

3.2.2 Reconstruction of Knowledge Graph to Queries and Answers

Our task is the link prediction of a knowledge graph, i.e., to predict the missing entity of a query, which is formally defined as follows:

Query: A query is a triple which is missing an entity: $(h, r, ?)$ or $(?, r, t)$.

We divide a knowledge graph G into queries and answers to use as training data for our model. Let $Q_{r,\text{head}}$ ($Q_{r,\text{tail}}$) denote the set of training queries missing a head (tail) entity for a relation r obtained from G; then $Q_{r,\text{head}}$ ($Q_{r,\text{tail}}$) is defined as follows:

$$Q_{r,\text{head}} = \{(?, r, t) \mid (h, r, t) \in G\} \tag{3.1}$$

$$Q_{r,\text{tail}} = \{(h, r, ?) \mid (h, r, t) \in G\} \tag{3.2}$$

In this case, the answers of training queries are defined as follows:

$$a_{(?,r,t)} = \{h \mid (h, r, t) \in G\} \tag{3.3}$$

$$a_{(h,r,?)} = \{t \mid (h, r, t) \in G\} \tag{3.4}$$

We additionally need negative examples to evaluate rules, however, a knowledge graph usually contains only positive triples. We also do not want to employ Closed World Assumption because we suppose a knowledge graph is not complete. Hence, we need to employ another assumption.

3.2.2.1 Balanced Partial Completeness Assumption

Instead of CWA, we want to adopt the partial completeness assumption (PCA) [47, 48] to generate negative answers.

Partial Completeness Assumption: if (h, r, t) is in G, then

$$\forall t' \in E, ((h, r, t') \notin G \Rightarrow (h, r, t') \text{ is negative}) \tag{3.5}$$

However, this assumption connote an imbalance of negative examples, i.e. only the tail entity is changed to generate negative examples, while we can consider changing the head entity. One of the problems of AMIE is a rule is evaluated based on this unbalanced PCA. We solve the problem by balancing PCA as follows:

Balanced Partial Completeness Assumption: if (h, r, t) is in G, then

$$\forall t' \in E, ((h, r, t') \notin G \Rightarrow (h, r, t') \text{ is negative}) \tag{3.6}$$

$$\forall h' \in E, ((h', r, t) \notin G \Rightarrow (h', r, t) \text{ is negative}) \tag{3.7}$$

We added the second equation because we also need to allow negative answers for $Q_{r,\text{head}}$. Under balanced PCA, negative answers for each question are defined as follows:

$$n_{(?,r,t)} = E \setminus a_{(?,r,t)} \tag{3.8}$$

$$n_{(h,r,?)} = E \setminus a_{(h,r,?)} \tag{3.9}$$

We use this assumption and training examples through this dissertation.

3.3 Standard Confidence and Problems

3.3.1 AMIE with GPARs

An association rule is essentially a binary classifier, i.e. the antecedent of an association rule matches or does not match, and an association rule is thus evaluated. Following this idea, we suggest the most straightforward way to define the confidence, which indicates the reliability of an association rule, is the conditional probability, which is the probability of the consequent given the antecedent for a GPAR. The conditional probability $Pr_{tail}(r_{(x,y)} \mid GP_{(x,y)})$ of a GPAR $GP_{(x,y)} \Rightarrow r_{(x,y)}$ to predict a tail is defined as follows:

$$\begin{aligned} conf_{tail}(GP_{(x,y)} \Rightarrow r_{(x,y)}) &= Pr_{tail}(r_{(x,y)} \mid GP_{(x,y)}) \\ &= \frac{\sum_{(h,r,?)\in Q_{r,\mathrm{tail}}} |\{t \in a_{(h,r,?)} \mid M(GP_{(x,y)}, (h,t)) \neq \emptyset\}|}{\sum_{(h,r,?)\in Q_{r,\mathrm{tail}}} |\{t' \in E \mid M(GP_{(x,y)}, (h,t')) \neq \emptyset\}|} \end{aligned} \tag{3.10}$$

For each query, the candidate entities found by the graph pattern are counted for the denominator while only correct entities are counted for the numerator. This confidence is used to evaluate GPARs only to answer queries with a missing tail because $Q_{r_{tail}}$ and its answers are used to define it.

Interestingly, GPARs with this confidence are equivalent to AMIE [47, 48], which was proposed to find horn clauses for a knowledge graph, although AMIE was proposed before the appearance of GPARs. However, AMIE originally has only one confidence value for a GPAR because AMIE employs unbalanced PCA. Hence, we introduce the following alternative definition for the confidence value to answer a query missing a head entity.

3.3.2 Standard Link Prediction Model and Problems

We define another confidence to deal with a query with a missing head entity as follows:

$$\begin{aligned} conf_{head}(GP_{(x,y)} \Rightarrow r_{(x,y)}) &= Pr_{head}(r_{(x,y)} \mid GP_{(x,y)}) \\ &= \frac{\sum_{(?,r,t)\in Q_{r,\mathrm{head}}} |\{h \in a_{(?,r,t)} \mid M(GP_{(x,y)}, (h,t)) \neq \emptyset\}|}{\sum_{(?,r,t)\in Q_{r,\mathrm{head}}} |\{h' \in E \mid M(GP_{(x,y)}, (h',t)) \neq \emptyset\}|} \end{aligned} \tag{3.11}$$

Additionally, we restrict matching functions to injective functions as defined in Section 3.2.1, which is different from AMIE, because the restriction avoids redundant matching functions which map multiple variables to the same entity and gives a good bias for real-world knowledge. For example, an GPAR $GP_{3,(x,y)} \Rightarrow \mathsf{sibling\_of}_{(x,y)}$, where $GP_{3,(x,y)} = \{(z, \mathsf{parent\_of}, x), (y, \mathsf{child\_of}, z)\}$, is helpful to predict siblings. However, let p represent a person, $GP_{3,(x,y)}$ matches (p, p) although p is not a sibling of p. The above restriction omits such concerns. For another example, a GPAR $\{(z_1, \mathsf{manager\_of}, x),$ $(z_1, \mathsf{manager\_of}, z_2), (z_2, \mathsf{located\_in}, y)\} \Rightarrow \mathsf{located\_in}_{(x,y)}$ on the graph G_{ex} in Figure 3.1 should not be considered helpful because $m(x) = m(z_2)$ holds for a matching function m

of the antecedent pattern and as a result, the GPAR is almost tautological. We consider two confidence values for GPARs, $conf_{tail}$ and $conf_{head}$, referred to as the graph pattern probability model (GPro).

However, GPro cannot deal with queries where counting the number of matching functions is crucial. An example where the number of matching functions is important is shown in Figure 3.1. In G_{ex}, the country that Team C is located in is missing. One might guess that Team C is located in Italy because most of the Team C players have Italian nationality and the nationality of a player often matches the country that the team is located in. However, GPro underestimates the GPAR $AR_{1,(x,y)} = GP_{1,(x,y)} \Rightarrow \textsf{located\_in}_{(x,y)}$, which is equivalent to one's guessing process: $conf_{tail}(AR_{1,(x,y)}) = 2/5$, while $conf_{tail}(AR_{2,(x,y)}) = 1/2$, where $AR_{2,(x,y)} = GP_{2,(x,y)} \Rightarrow \textsf{located\_in}_{(x,y)}$. Hence, GPro judges $AR_{2,(x,y)}$ is more useful than $AR_{1,(x,y)}$, and as a result, GPro predicts Team C is located in Germany rather than Italy.

This problem is caused by considering a GPAR as a binary classifier, i.e. the matching number is not taken into account. For example, if we apply $AR_{1,(x,y)} = P_{1,(x,y)} \Rightarrow \textsf{located\_in}_{(x,y)}$ to a query $(\text{Team A}, \textsf{located\_in}, ?)$ in the traditional way (as a binary classifier), the output will contain two entities with equal weighting, the U.K. and France, because $P_{1,(x,y)}$ matches $(\text{Team A}, \text{U.K.})$ and $(\text{Team A}, \text{France})$. Then, one of the output entities is correct and the other is incorrect. This is the reason why $AR_{1,(x,y)}$ is underestimated.

To deal with this problem, in this paper, we consider a GPAR as an entity ranking system by counting the number of matching functions of the antecedent graph pattern rather than considering as a binary classifier.

3.4 GPAR as Entity Ranking System and Evaluation Metrics

As well as considering a GPAR as a binary classifier, we consider it as an entity ranking system. Entities are ranked according to a score, based on their number of matching functions.

Moreover, we introduce the distributed rankings for entities, which are proposed to deal with situations where multiple entities have the same score. Then, we define the evaluation metrics for the distributed rankings to evaluate GPARs for link prediction.

These approaches overcome the problems shown in Section 3.3.2.

3.4.1 GPAR as Entity Ranking System

We consider a GPAR as a ranking system in this section to rank queries for which counting the number of matching functions of the antecedent is helpful, as shown in Section 3.3.2.

First, we define a scoring function whose arguments are a graph pattern $GP_{(x,y)}$ and a pair of entities (h, t). The scoring function returns the number of matching functions of a pattern on a pair, which is formally defined as follows:

$$score(GP_{(x,y)}, (h, t)) = |M(GP_{(x,y)}, (h, t))| \tag{3.12}$$

Given a pattern $GP_{(x,y)}$ and a query $(h, r, ?)$, we can obtain the $score(GP_{(x,y)}, (h, t'))$ for each candidate tail entity t'. Then we obtain the rankings of the tail entities in descending order of the scores. The head entity rankings for a query $(?, r, t)$ are also obtained in this way. This ranking method gives us new perspective when we apply GPARs to answer a query. For example, if we apply $AR_{1,(x,y)} = P_{1,(x,y)} \Rightarrow \mathsf{located\_in}_{(x,y)}$ to a query $(\text{Team A}, \mathsf{located\_in}, ?)$ the U.K. will be ranked first and France second. In this situation, we can say that $AR_{1,(x,y)}$ works because the correct entity ranks higher than the wrong entity. We can basically evaluate a GPAR as an entity ranking system by evaluating output rankings by an evaluation metric for an ranking system such as the mean average precision. However, often multiple entities have the same score and traditional metrics cannot deal with the situation. To deal with this problem, we propose a new concept, called distributed rankings, and the corresponding metrics in the following sections.

3.4.2 Distributed Rankings

We propose distributed rankings where each entity can distribute over multiple ranks and each rank can have multiple entities, to deal with situations where multiple entities have the same score.

Traditional rankings of entities are represented by a matrix $Rank = (rank_{i,j}) \in \{0, 1\}^{n \times n}$, where n is the number of entities, and for each column and row there is one 1 element. In this matrix, columns represent entities and rows represent ranks. For example, $rank_{i,j} = 1$ means that the entity j has rank i. On the other hand, distributed rankings of entities are represented by a matrix $dRank = (drank_{i,j}) \in [0, 1]^{n \times n}$, where the summation of a column or a row is equal to 1. Different from traditional rankings, the value of each element is continuous and multiple elements can be greater than 0 in a column or a row. For example, $rank_{i,j} = 0.5$ means that half of the entity j has rank i. Note that a traditional ranking matrix is a distributed ranking matrix.

Given a pattern $GP_{(x,y)}$ and a query $(h, r, ?)$, We obtain distributed rankings of entities, $dRANK(GP_{(x,y)}, (h, r, ?))$, according to their scores as follows. Let a be the number of entities whose scores are greater than the entity represented by j and let b be the number of entities whose scores are the same as the entity represented by j. Then, $drank_{i,j}$, an element of $dRANK(GP_{(x,y)}, (h, r, ?))$, is determined to be $1/b$ for $a + 1 \leqq i \leqq a + b$ and 0 otherwise. Distributed rankings of head entities for a query $(?, r, t)$ are obtained in the same way, and we refer to them as $dRANK(GP_{(x,y)}, (?, r, t))$. Unlike traditional rankings, distributed rankings are uniquely determined from the scores of entities.

Traditional rankings can be evaluated by metrics such as the average precision or the cumulative gain. However, distributed rankings cannot be evaluated by these metrics. Hence, we require a different evaluation metric for distributed rankings.

3.4.3 Evaluation of GPARs as Entity Ranking System

We use a GPAR to obtain distributed entity rankings as shown in Section 3.4.1. In this section, we define a metric to evaluate distributed rankings of entities by generalizing the average precision to evaluate a GPAR.

For a pattern $GP_{(x,y)}$ and a training query $(h, r, ?)$, the distributed precision at k, $dPre_k$, of $dRANK(GP_{(x,y)}, (h, r, ?))$ is defined as follows:

$$dPre_k(GP_{(x,y)}, (h, r, ?)) = \frac{\sum_{i=1}^{k} \sum_{t_j \in a_{(h,r,?)}} drank_{i,j}}{k} \tag{3.13}$$

where t_j is an entity represented by j and $drank_{i,j}$ is an element of $dRANK(GP_{(x,y)}, (h, r, ?))$. The elements related with correct entities ranked higher or equal to k are summed up as the traditional precision at k.

Then, the distributed average precision, dAP, is defined for a pattern $GP_{(x,y)}$ and a training query $(h, r, ?)$ as follows:

$$dAP(GP_{(x,y)}, (h, r, ?)) = \frac{\sum_{t_j \in a_{(h,r,?)}} \sum_{k=1}^{n} dPre_k(GP_{(x,y)}, (h, r, ?))) \times drank_{k,j}}{|a_{(h,r,?)}|} \tag{3.14}$$

where t_j is an entity represented by j, $drank_{i,j}$ is an element of $dRANK(GP_{(x,y)}, (h, r, ?))$, and n is the number of entities. The numerator of the average precision for traditional rankings is the summation of the precision at k for relevant entities. However, a relevant entity represented by j is distributed over multiple ranks in $dRANK$ so that the precision at k multiplied by $drank_{k,j}$ is summed over k where a relevant entity j is distributed. $dAP(GP_{(x,y)},(?,r,t))$ for a training query with a missing head can be defined in the same way. The distributed mean average precision for a GPAR $GP_{(x,y)} \Rightarrow r_{(x,y)}$ is defined as follows:

$$dMAP_{head}(GP_{(x,y)} \Rightarrow r_{(x,y)}) = \sum_{(?,r,t)\in Q_{r,\text{head}}} \frac{dAP(GP_{(x,y)},(?,r,t))}{|Q_{r,\text{head}}|} \tag{3.15}$$

$$dMAP_{tail}(GP_{(x,y)} \Rightarrow r_{(x,y)}) = \sum_{(h,r,?)\in Q_{r,\text{tail}}} \frac{dAP(GP_{(x,y)},(h,r,?))}{|Q_{r,\text{tail}}|} \tag{3.16}$$

We also define dMAP with for the “ filtered” [56] rankings which are obtained from original rankings by eliminating entities whose corresponding triples (except the target triple) were included in the training dataset. ”Filtered” dMAP (fdMAP) is the mean of the dAP of ”filtered” rankings for each answer of queries.

We refer to GPARs considered as entity ranking systems with these dMAPs or fdMAPs as the graph pattern entity ranking model (GRank).

	WN18				FB15k				WN18RR				FB15k-237			
	MRR	HITS@			MRR	HITS@			MRR	HITS@			MRR	HITS@		
Model		1	3	10		1	3	10		1	3	10		1	3	10
AMIE	0.911	0.895	0.922	0.942	0.665	0.617	0.694	0.750	–	–	–	–	–	–	–	–
PRA	0.458	0.422	–	0.481	0.336	0.303	–	0.392	–	–	–	–	–	–	–	–
LinkFeat	0.938	–	–	0.939	0.779	–	–	0.804	–	–	–	–	0.073	–	–	0.087
RESCAL	0.890	0.842	0.904	0.928	0.354	0.235	0.409	0.587	–	–	–	–	–	–	–	–
DistMult	0.922	0.891	0.952	0.956	0.840	0.802	0.865	0.906	0.460	0.416	0.472	0.548	0.354	0.260	0.389	0.543
ComplEx	**0.951**	0.945	**0.955**	**0.962**	**0.856**	**0.827**	**0.872**	**0.909**	**0.476**	**0.429**	**0.493**	**0.564**	**0.365**	**0.269**	**0.401**	**0.555**
ANALOGY	0.942	0.939	0.944	0.947	0.725	0.646	0.785	0.854								
R-GCN	0.814	0.686	0.928	0.955	0.651	0.541	0.736	0.825	–	–	–	–	0.248	0.153	0.258	0.417
ConvE	0.942	0.935	0.947	0.955	0.745	0.670	0.801	0.873	0.46	0.39	0.43	0.48	0.316	0.239	0.350	0.491
GPro	0.950	0.946	0.954	0.959	0.793	0.759	0.810	0.858	0.467	0.430	0.485	0.543	0.229	0.163	0.250	0.360
GRank (dMAP)	0.950	**0.946**	0.953	0.957	0.841	0.814	0.855	0.890	0.466	0.434	0.480	0.530	0.312	0.233	0.340	0.473
GRank (fdMAP)	0.950	**0.946**	0.954	0.958	0.842	0.816	0.856	0.891	0.470	0.437	0.482	0.539	0.322	0.239	0.352	0.489

Table 3.1: Mean Reciprocal Rank (MRR) and HITS@n scores obtained for the link prediction tasks on the WN18, FB15k, WN18RR, and FB15k-237 datasets. The highest result for each column is shown in bold. the results of RESCAL were reported by Nickel et al. [86], the results of R-GCN and ConvE were reported by Dettmers et al. [78], the results of PRA were reported by Liu et al. [52], and the results of Node+LinkFeat were reported by Toutanova and Chen [85].

By using a graph pattern to rank entities, GRank is able to properly estimate GPARs where the number of matches is important as shown in Section 3.3.2, unlike GPro. For example, $dMAP_{tail}(AR_{1,(x,y)}) = 1$, is the maximum value, while $dMAP_{tail}(AR_{2,(x,y)}) = 1/2$ in Figure 3.1. Hence, GRank can answer the query $(\text{Team C}, \textsf{located\_in}, ?)$ by applying $AR_{1,(x,y)}$.

3.5 Experiments

Our proposed models, GPro (Section 3.3.2) and GRank (Section 3.4), are evaluated through link prediction tasks in this section.

3.5.1 Evaluation Protocol

We describe how to obtain rankings from models. We restricted antecedent graph patterns of GPARs to connected and closed [47, 48] patterns whose size $|GP_{(x,y)}|$ was less than or equal to L to restrict the search space. A connected and closed patterns is a pattern connecting x and y without branches, as shown in Figure 3.2. L was chosen for each model among $\{1, 2, 3\}$ by MRR from the validation triples of each dataset. It took about four days to evaluate all candidate GPARs for GRank with dMAPs in FB15k using an Intel Xeon Gold 6154 (3.00 GHz, 18 cores).

We now explain how we obtained the rankings for queries with missing heads. For each relation r, we chose the top 1,000 GPARs in descending order of the standard confidence, the dMAP, or the fdMAP to predict the heads. Let $GP_{i,(x,y)} \Rightarrow r_{(x,y)}$ be the obtained GPAR, where i denotes the rank. We defined the ordering for two entities for query $(?, r_t, t_t)$ as follows: for entities e_1 and e_2, we define $e_1 > e_2$ if there exists i' for which $score(GP_{i,(x,y)}, (e_1, t_t)) = score(GP_{i,(x,y)}, (e_2, t_t))$ for $i > i'$ and $score(GP_{i',(x,y)}, (e_1, t_t)) > score(GP_{i',(x,y)}, (e_2, t_t))$. We obtained the entity rankings with this ordering for each query. Rankings for queries with missing tails were obtained in the same way.

3.5.2 Results

The results of the link prediction tasks for our proposed models, GPro, GRank with dMAP, and GRank with fdMAP, are shown in Tables 3.1, where the results reported in previous studies are included for comparison.

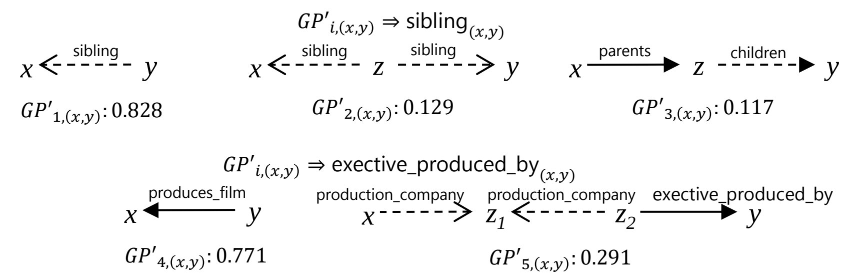


Figure 3.3: Examples of antecedent patterns for $dMAP_{tail}$ which were given high ranks by GRank for the FB15k dataset.

Table 3.1 shows the effectiveness of the LinkFeat model [85] on WN18 and FB15k among the previous observed feature models, although this model is very simple (high MRRs imply that the model also has high HITS@1s or HITS@3s). The LinkFeat model performed well on WN18 and FB15k because these datasets often contain the reverse relations of other relations. Our proposed models, GPro and GRank, generally yield competitive results to the knowledge graph embedding models and results which are better than or comparable to LinkFeat, which means that our models can also handle such redundancy. In particular, GRank with dMAP and fdMAP yielded better results on FB15k than the LinkFeat model and AMIE. This indicates that taking the multiplicity of matchings and deeper information into account is important for knowledge graphs such as FreeBase that contain miscellaneous relations and are not well curated like WordNet. As a result, GRank performed well.

On FB15k-237, The LinkFeat and GPro do not yield good results because FB15k-237 has less redundancy while GRank could performed such less redundancy datasets.

For FB15k-237, the performance of the LinkFeat model falls far behind embedding models and our models is comparable with most of the other more sophisticated knowledge graph models. That is natural because FB15k-237 is designed so that the LinkFeat model cannot perform well on it. GPro does not also yield good results because GPro mainly relies the omitted redundancy from FB15k. GRank performs better than GPro for the WN18RR dataset and the FB15k-237 dataset for the same reason as the FB15k dataset. However, GRank is overperformed by the state-of-the-art embedding models. This result suggests that there exists the information in a knowledge graph that GRank cannot utilize.

Quality of Obtained Paths The examples of antecedent patterns ranked high by GRank with $dMAP_{tail}$ for FB15k are shown in Figure 3.3. The patterns shown for predicting the sibling relation are all correct as the antecedents of GPARs; however, the MAP of $GP'_{2,(x,y)}$ and $GP'_{3,(x,y)}$ are low. The reason for this is that $GP'_{2,(x,y)}$ works when an individual has more than two siblings. The MAP of $GP'_{3,(x,y)}$ is low because individual's parents are often missing in FB15k. However, they are still ranked higher than other patterns.

The $\mathsf{produces\_film}$ relation is the inverse relation of the $\mathsf{exective\_produced\_by}$ relation in FB15k. Such patterns are very helpful when performing link prediction tasks, and GRank is able to find them. However, the MAP is not as high because of missing facts. GRank is able to use majority rules such as $GP'_{5,(x,y)} \Rightarrow \mathsf{film\_produced\_by}(\mathsf{x}, \mathsf{y})$ instead in such cases. This rule can be interpreted as stating that a particular film was likely to have been produced by a person who produced many films in the same production company.

Output triples of GRank (and GPAR-based models) are described by antecedent patterns unlike knowledge graph embedding models as shown here.

3.6 Conclusions

In this chapter, we first defined GPARs for a knowledge graph and the standard confidence measures of GPARs for link prediction. Then, we pointed out the problems with the standard confidence measures and we introduced a new perspective using GPARs to rank entities to overcome these problems. We also proposed distributed rankings for situations where multiple entities have the same scores and defined metrics for them. This idea led us to propose the GRank model. GRank is easy to interpret because outputs are described by GPARs, unlike knowledge graph embedding models, and so efficient that it competes the state-of-the-art knowledge graph embedding models in link prediction tasks, which supports the usefulness of interpretation.

Chapter
4

Knowledge Graph Embedding on Lie Group with Attention

4.1 Overview

TransE [56], the original translation-based model for link prediction tasks, is well known because of its simplicity. We additionally think TransE may utilize rules and, therefore may be interpretable. TransE embeds triples and relations on a real vector space with the principle $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$, where $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$ are embeddings of h, r and t, respectively, if the triple (h, r, t) is stored in the knowledge graph used as training data. On the other hand, TransE does not match state-of-the-art embedding models at the efficiency of link prediction; that implies we may not get useful information even though we interpret TransE. In this chapter, we attempt to the efficiency of TransE for link prediction.

The main problem with TransE is low expressivity which is not enough to represent some relations. In this chapter, we deal with this problem by generalizing TransE in two steps. Firstly, we generalize TransE to take any Lie group for the embedding space. Secondly, we generalize it for separating space to store apart the information of unrelated entities and relations, which solves a part of the problem coming from the strictness of the principle.

The remainder of this chapter is organized as follows. In Section 4.2, we briefly introduce TransE. In Section 4.2, we discuss the main limitation of TransE: the low expressiveness. In Section 4.3, we propose a comprehensive model for knowledge graph embedding: knowledge graph on a Lie group (KGLG). We also propose an instance of KGLG which embeds a knowledge graph on a torus. This approach overcomes a part of the limitation coming from the characteristics of the embedding space. In Section 6, we present an experimental study in which we compare our method with baseline results of benchmark datasets. In Section 7, we conclude this chapter.

4.2 Mechanism of TransE

The algorithm of TransE consists of three main parts as follows:

- *Principle*: TransE learns embeddings so that $\boldsymbol{h}+\boldsymbol{r} = \boldsymbol{t}$ holds if $(h, r, t) \in \Delta$, where Δ denotes the set of true triples. To measure how much a triple embedding follows the principle, a scoring function f is used. Usually L_1 or the square of the L_2 norm of $\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}$ is used as $f(h, r, t)$. In this case, $f(h, r, t) = 0$ means $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ holds completely.
- *Negative Sampling*: With only the principle, TransE learns the trivial solution that all entity embeddings are the same and all relation embeddings are 0. Hence,

negative triples are required. Usually, a knowledge graph contains only positive triples, so TransE makes a negative triple by changing the head or the tail entity at random for each true triple. This is called negative sampling. TransE learns embeddings so that $f(h', r, t')$ gets larger if $(h', r, t') \in \Delta'_{(h,r,t)}$, where $(h, r, t) \in \Delta$ and $\Delta'_{(h,r,t)} = \{(h', r, t) | h' \in E,\ h' \neq h\} \cup \{(h, r, t') | t' \in E,\ t' \neq t\}$.

- *Regularization*: To not allow embeddings to diverge unlimitedly, regularization is needed. TransE employs normalization as regularization. Embeddings of entities are normalized so that their magnitude becomes 1 in each step of learning. That is, for every entity $e \in E$, $\boldsymbol{e} \in S^{n-1} \subset \mathbb{R}^n$, where S^{n-1} is an *n-1* dimensional sphere.

TransE originally exploits margin loss. The objective function is defined as follows:

$$\mathcal{L} = \sum_{(h,r,t) \in \Delta} \sum_{(h',r,t') \in \Delta'_{(h,r,t)}} [\gamma + f(h, r, t) - f(h', r, t')]_+ \tag{4.1}$$

where $[x]_+$ denotes the positive part of x and $\gamma > 0$ is a margin hyperparameter. TransE is trained by using stochastic gradient descent.

The principle allows TransE to utilize first-order rules based on a path. If a rule $(r_1, r_2, ..., r_n) \Rightarrow r$ holds and there are enough matching in a knowledge graph, then the embeddings of these relations are trained to follow the equation:

$$\sum_{k=1}^{i} r_k := r_1 + r_2 + \cdots + r_k = r \tag{4.2}$$

We can get this equation by the sequential application of the principle.

We think TransE deal with rules employing this equation. However, it seems that TransE does not have the ability to compatibly embed various entities and relations, in other words, TransE suffers from its low expressiveness. We think this problem is caused by two different parts of TransE.

One of them is the embedding space, $\mathbf{R}^n$. $\mathbf{R}^n$ is an open space, hence, we need the regularization to prevent that embeddings diverge and get sparse infinitely. However, the principle and regularization conflict during training, because for each $e \in E$ and $r \in R$, $\boldsymbol{e} + \boldsymbol{r} \notin S^{n-1}$ almost always holds. To be more precise, for each triple (h, r, t), $\boldsymbol{h}$ should be in $(\boldsymbol{r}^{\perp} - \boldsymbol{r}/2) \cap S^{n-1} \simeq S^{n-2}$, $\boldsymbol{t}$ should be in $(\boldsymbol{r}^{\perp} + \boldsymbol{r}/2) \cap S^{n-1} \simeq S^{n-2}$ to realize the principle $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ exactly, where $\boldsymbol{r}^{\perp} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \cdot \boldsymbol{r} = 0\}$. This limitation makes it hard to get exact embeddings of a knowledge graph. For example, if a knowledge graph contains triples such as (h, r, e) and (e, r, t) sharing a entity as head and tail in the same

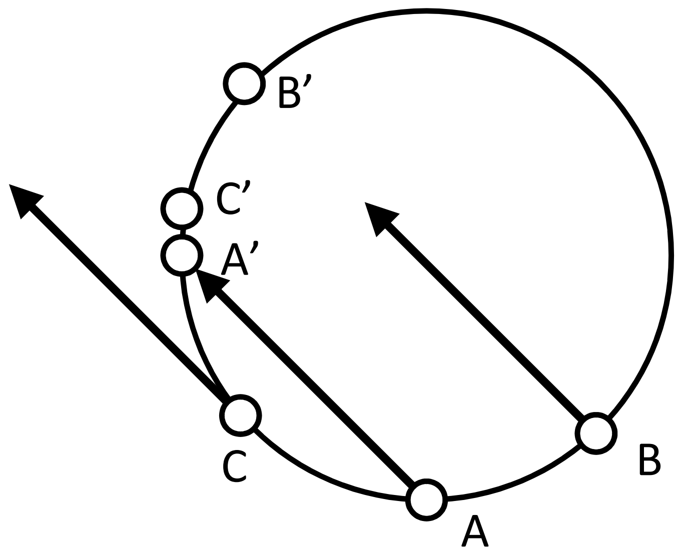


FIGURE 4.1: The image of embeddings obtained by TransE when n is 2. It is assumed that $(A, r, A'), (B, r, B')$ and (C, r, C') hold.

relation, it cannot be exactly embedded because $\boldsymbol{e}$ should be in $(\boldsymbol{r}^{\perp} - \boldsymbol{r}/2) \cap S^{n-1}$ and $(\boldsymbol{r}^{\perp} + \boldsymbol{r}/2) \cap S^{n-1}$ while $(\boldsymbol{r}^{\perp} - \boldsymbol{r}/2) \cap (\boldsymbol{r}^{\perp} + \boldsymbol{r}/2) = \emptyset$ Hence, the principle $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ is rarely realized in most cases, as shown in Fig. 4.1. In this figure, it is assumed that $(A, r, A'), (B, r, B')$ and (C, r, C') hold. The points represent the entity embeddings and the arrows represent the embedding of r. Embeddings of (A, r, A') are obtained so that they follow the principle completely. However, $\boldsymbol{B} + \boldsymbol{r}$ and $\boldsymbol{C} + \boldsymbol{r}$ are out of the sphere and $\boldsymbol{B}'$ and $\boldsymbol{C}'$ are regularized on it. The regularization warps embeddings and they do not satisfy the principle. As a result, it becomes difficult to predict new triples more accurately. This problem has not been explicitly mentioned before but it has been known TransE yields better results by changing its regularization. For example, Lin et al. and Trouillon et al. employed L_1 or L_2 regularization for TransE in their papers [65, 87]. Nickel et al. employed no regularization but AdaGrad for training instead [86]. However, their TransEs are still not competitive to state-of-the-arts. We think that is because L_1 and L_2 regularizations decay what a model has learned, and AdaGrad fixes embeddings after some epochs of training so that a model cannot continue to learn new facts with AdaGrad. Hence we need to get rid of the regularization from a translation-based model with stochastic gradient descent.

The embedding space $\mathbf{R}^n$ also make TransE hard to deal with symmetrical rules. if r is a symmetrical rule and (h, r, t) is a true triple, then (t, r, h) is also a true triple. If we apply the principle for those triples, then an equation, $\boldsymbol{r} = 0$, is inferred, which is not unpreferable for us because we want to distinguish different relations.

The second cause of the problem is that the principle is too strict. For example, if a head entity/relation pair have multiple valid tail entities and the embeddings perfectly follow the principle, then all of the tail entities have to be represented by the same point; this is undesirable because we need to distinguish different entities.

4.3 Knowledge Graph Embedding on a Lie Group

In this section, we introduce the knowledge graph embedding on a Lie group model (KGLG), which is more comprehensive than previous translation-based models. The KGLG model allows us to choose any Lie group as embedding space for a knowledge graph and it helps to solve the regularization problem in following Section 4.4 while employing the same principle used in TransE. We employes the weighted negative sampling part method (WNP) for the objective function of KGLG to solve another problem of the unchangeable number of negative sampling. We first consider the required conditions for an embedding space. Then, a Lie group is introduced as candidate embedding spaces and we propose the KGLG model.

4.3.1 Required Conditions for Embedding Space

Some conditions are required for an embedding space according to the embedding strategy of TransE.

First, an embedding space has to be equipped with an operation between two points on it to define the principle. Hence, we choose an embedding space from groups because a group has the group operation which is suitable to define the principle. To construct a scoring function for a model, a distance function is required to measure the difference between two points. Additionally, we require an embedding space to be a differential manifold because we employ gradient descent to get optimized embeddings.

Embeddings are restricted on S^n for TransE but we cannot use S^n does not fill the conditions because S^n cannot be equipped with a proper group operation except for special cases, n is 0, 1, or 3.

Actually, a Lie group fills all conditions required for embedding spaces with the TransE strategy. We explain the Lie group in the next section.

4.3.2 Lie Group

The foundation of the theory of Lie groups [88–91] was established by Sophus Lie. Lie groups, which play various roles in physics and mathematics, are defined as follows.

Definition 4.1. A Lie group is a group that is also a finite-dimensional smooth manifold, in which the group operation $+$ of multiplication and inversion are smooth maps.

We can use the group operation $+$ of a Lie group to define the principle on it. It is known that distance function d can be defined on any manifold. Hence, we can define a scoring function employing the distance function on a Lie group. It is also known that every smooth manifold can carry a Riemannian metric. Hence every Lie group can be a Riemannian manifold by defining a Riemannian metric. We can optimize a real-valued function on a Riemannian manifold through specific SGD for a Riemannian manifold such as RSGD[92] or RSVRG[93].

4.3.2.1 Examples of Lie Groups

Some Lie groups are simple and common. We give some examples of Lie groups here.

- $\mathbb{R}^n$: A real vector space $\mathbb{R}^n$ is the most common Lie group. Its group operation is the canonical addition operation.
- $(\mathbb{R}\backslash\{0\})^n$: $(\mathbb{R}\backslash\{0\})^n$ is also a Lie group and its group operation is the element-wise multiplication operation.
- $GL_n(\mathbb{R})$: A general linear group $GL_n(\mathbb{R})$ considered as a submanifold of $\mathbb{R}^{n^2}$ is a Lie group. Its group operation is the canonical multiplication operation of matrices.
- $U(n)$: A unitary group $U(n)$ considered as a submanifold of $\mathbb{C}^{n^2}$ is a Lie group. Its group operation is the canonical multiplication operation of matrices.

Note that if G and H are Lie groups then $G \times H$ is also a Lie group, where $G \times H$ is a product as manifolds and a direct product as groups of G and H.

4.3.3 The KGLG Model

TransE assumes embeddings of entities and relations are on $\mathbb{R}^n$. If (h, r, t) holds for TransE, embeddings should follow the principle $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$; otherwise, $\boldsymbol{h} + \boldsymbol{r}$ should be far away from $\boldsymbol{t}$, where $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t} \in \mathbb{R}^n$ represents h, r, and t, respectively. Our proposed method, Knowledge Graph Embedding on a Lie group (KGLG), follows the principle also, but the embedding space is changed from a vector space to any Lie group.

For KGLG, each entity $e \in E$ and each relation $r \in E$ are represented respectively by $g_e \in G$ and $g_r \in G$ where G is a Lie group. Then, the principle is rewritten as follows:

$$g_h + g_r = g_t \tag{4.3}$$

and embeddings are obtained by minimizing $f_l(h, r, t)$ for $(h, r, t) \in \Delta$ and maximizing $f_l(h, r, t)$ for $(h', r, t') \in \Delta'_{(h,r,t)}$, where f_l is a scoring function, for example $f_l = d(g_h + g_r), g_t)$ where d is a given distance function on G.

Stochastic gradient descent cannot be generally used for KGLG, hence KGLG is optimized by RSGD[92] or RSVRG[93] with a given riemmanian metric g on G.

TransE is an instance of KGLG. $\mathbb{R}^n$ is the embedding space of TransE and it is a Lie group. Additionary, it is equipped with the canonical Euclidean metric.

4.4 TorusE

In this section, we introduce "compactness" which is a property of a topological space, especially a manifold. We show a regularization is not required when we employ a compact manifold as an embedding space of KGLG. Then, we propose a novel model, TorusE, which is an instance of KGLG and whose embedding space is a torus, a compact Lie group.

4.4.1 Compactness

Compactness is a property of a topological space. The definition of compactness is as follows.

Definition 4.2. A topological space is compact if each of its open covers has a finite subcover.

For example, a subspace U of $\mathbb{R}^n$ is compact if and only if U is bounded and closed. In the case of a manifold, compactness can be equivalently defined as follows.

Definition 4.3. A manifold M is compact if there exists a bounded closed submanifold M' of $\mathbb{R}^n$ such that M and M' are diffeomorphic.

It is known that compactness is preserved under a continuous map, i.e. let X be a compact space, let Y be a topological space, and let f be a continuous map from X to Y, then $f(X)$ is a compact subspace of Y. Hence, if an embedding space of KGLG is compact, then the objective function L has the minimum value and embeddings never diverge without a regularization. That means a regularization is not needed any longer.

S^n is compact so that it was employed by TransE. However, we cannot define a summation well on it because it is not a Lie group.

4.4.2 Torus

We introduce a torus, which is a compact Lie group, and define distance functions on a torus. The definition of a torus is as follows.

Definition 4.4. An n-dimensional torus T^n is a quotient space, $\mathbb{R}^n/\sim = \{[\boldsymbol{x}] | \boldsymbol{x} \in \mathbb{R}^n\} = \{\{\boldsymbol{y} \in \mathbb{R}^n | \boldsymbol{y} \sim \boldsymbol{x}\} | \boldsymbol{x} \in \mathbb{R}^n\}$, where $\sim$ is an equivalence relation and $\boldsymbol{y} \sim \boldsymbol{x}$ if and only if $\boldsymbol{y} - \boldsymbol{x} \in \mathbb{Z}^n$.

Through the natural projection $\pi : \mathbb{R}^n \to T^n, \boldsymbol{x} \mapsto [\boldsymbol{x}]$, the topology, the differential structure and the Riemannian metric g_{torus} of a torus is derived from the vector space. Note that $h : T^n \to \underbrace{S^1 \times S^1 \times \cdots \times S^1}_{n} \subset \mathbb{C}^n, [\boldsymbol{x}] \mapsto exp(2\pi i \boldsymbol{x})$ is a diffeomorphism. The group operation $+_{torus}$ is also derived from the original vector space: $[\boldsymbol{x}] +_{torus} [\boldsymbol{y}] := [\boldsymbol{x} + \boldsymbol{y}]$. A torus is a compact Abelian Lie group with these structures and group operation. We define distance functions in three ways:

- d_{L_1}: A distance function d_{L_1} on T^n is derived from the L_1 norm of the original vector space by defining $d_{L_1}([\boldsymbol{x}], [\boldsymbol{y}]) = \min_{(\boldsymbol{x}', \boldsymbol{y}') \in [\boldsymbol{x}] \times [\boldsymbol{y}]} ||\boldsymbol{x}' - \boldsymbol{y}'||_1$.
- d_{L_2}: A distance function d_{L_2} on T^n is derived from the L_2 norm of the original vector space by defining $d_{L_2}([\boldsymbol{x}], [\boldsymbol{y}]) = \min_{(\boldsymbol{x}', \boldsymbol{y}') \in [\boldsymbol{x}] \times [\boldsymbol{y}]} ||\boldsymbol{x}' - \boldsymbol{y}'||_2$.
- d_{eL_2}: T^n can be embedded on $\mathbb{C}^n$ by h. A distance function d_{eL_2} on T^n is derived from the L_2 norm of the $\mathbb{C}^n$ by defining $d_{eL_2}([\mathrm{x}], [\mathrm{y}]) = ||h([\mathrm{x}]) - h([\mathrm{y}])||_2$.

These distance functions are the most canonical ones on a torus because π is a local isomorphism when a torus is equipped with d_{L_1} or d_{L_2} and h is a local isomorphism when a torus is equipped with d_{eL_2}. These distance functions are used to define scoring functions for TorusE shown in the following section.

A torus has an advantage for an embedding space of knowledge graph other than compact compared with a real or compact vector space: A torus has a point $[x]$ such that $x \neq 0$ and $x +_{torus} x = 0$ hold. This enables symmetric relations such as "spouse" to be represented by other than 0.

TABLE 4.1: Scoring functions for triple (h, r, t), parameters and complexity of related work.

Model	Scoring Function	Parameters	$\mathcal{O}_{time}$	$\mathcal{O}_{space}$
TransE	$\|\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|\|_i$	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
TransH	$\|\|(\boldsymbol{h} - \boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{h}\boldsymbol{w}_r) + \boldsymbol{r} - (\boldsymbol{t} - \boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{t}\boldsymbol{w}_r)\|\|_i$	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}, \boldsymbol{w}_r \in \mathbb{R}^n$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
TransR	$\|\|\boldsymbol{W}_r\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{W}_r\boldsymbol{t}\|\|_i$	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r} \in \mathbb{R}^k, \boldsymbol{W}_r \in \mathbb{R}^{k\times n}$	$\mathcal{O}(kn)$	$\mathcal{O}(kn)$
RESCAL	$\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}_r\boldsymbol{t}$	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{W}_r \in \mathbb{R}^{n\times n}$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
DistMult	$\boldsymbol{h}^{\mathrm{T}}diag(\boldsymbol{r})\boldsymbol{t}$	$\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{R}^n$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
ComplEx	$Re(\boldsymbol{h}^{\mathrm{T}}diag(\boldsymbol{r})\bar{\boldsymbol{t}})$	$\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{C}^n$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
NTN	$\boldsymbol{u}_r^{\mathrm{T}}tanh(\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}_r^{[1:k]}\boldsymbol{t} + \boldsymbol{V}_{r,h}\boldsymbol{h} + \boldsymbol{V}_{r,t}\boldsymbol{t} + \boldsymbol{b}_r)$	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{u}_r, \boldsymbol{b}_r \in \mathbb{R}^k$, $\boldsymbol{W}_r^{[1:k]} \in \mathbb{R}^{k\times n\times n}, \boldsymbol{V}_{r,h}, \boldsymbol{V}_{r,t} \in \mathbb{R}^{k\times n}$	$\mathcal{O}(kn^2)$	$\mathcal{O}(kn^2)$
TorusE	$min_{(\boldsymbol{x},\boldsymbol{y})\in([\boldsymbol{h}]+[\boldsymbol{r}])\times[\boldsymbol{t}]}\|\|\boldsymbol{x} - \boldsymbol{y}\|\|_i$	$[\boldsymbol{h}], [\boldsymbol{r}], [\boldsymbol{t}] \in T^n$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

4.4.3 TorusE

TorusE is an instance of KGLG. The embedding space of TorusE is a Torus.

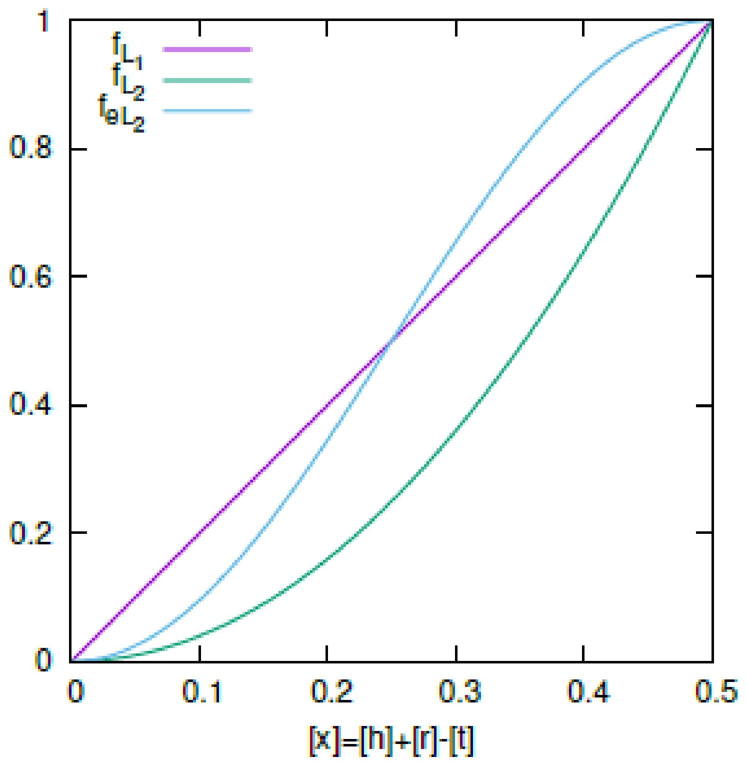
For TorusE, each entity $e \in E$ and each relation $r \in E$ are represented by $[\boldsymbol{e}] \in T^n$ and $[\boldsymbol{r}] \in T^n$, respectively. We define scoring functions in three ways that exploit the distance functions described in the previous section:

- f_{L_1}: We define a scoring function $f_{L_1}(h, r, t)$ as $2d_{L_1}([\boldsymbol{h}] +_{torus} [\boldsymbol{r}], [\boldsymbol{t}])$.
- f_{L_2}: We define a scoring function $f_{L_2}(h, r, t)$ as $4d_{L_2}^2([\boldsymbol{h}] +_{torus} [\boldsymbol{r}], [\boldsymbol{t}])$.
- f_{eL_2}: We define a scoring function $f_{eL_2}(h, r, t)$ as $d_{eL_2}^2([\boldsymbol{h}] +_{torus} [\boldsymbol{r}], [\boldsymbol{t}])/4$.

The scoring functions are normalized so that their maximum values are n. These scoring functions and their derivatives when $n = 1$ are illustrated in Fig. 4.2. f_{L_1}, f_{L_2} and f_{eL_2} look similar; however their derivatives are surprisingly different. f'_{L_1} is constant, f'_{L_2} has a vanishing point at $x = 0$, and f'_{eL_2} has two vanishing points at $x = 0$ and $x = 0.5$. These affect the obtained embeddings through RSGD or RSVRG.

TorusE does not require any regularization and calculation time for regularization, so it is expected to be more scalable than TransE. The image of embeddings obtained by TorusE is shown in Fig. 4.3.

The scoring functions and the complexity of related models are listed in Table 4.1. Although ComplEx is a bilinear model and TorusE is a translation-based model, they have strong similarity. By mapping $[\boldsymbol{h}], [\boldsymbol{r}]$ and $[\boldsymbol{t}]$ on $\mathbb{C}^n$ by h and identifying $h([\boldsymbol{r}])$ as a corresponding diagonal matrix, $-2f_{eL_2}(h, r, t) + 1 = f_{ComplEx}(h, r, t)$ holds. Bilinear models are trained to maximize the scores of triples while translation-based models are trained to minimize them. Hence, TorusE with f_{eL_2} can be considered as a more restricted and less redundant version of ComplEx on $T^n \subset \mathbb{C}^n$.



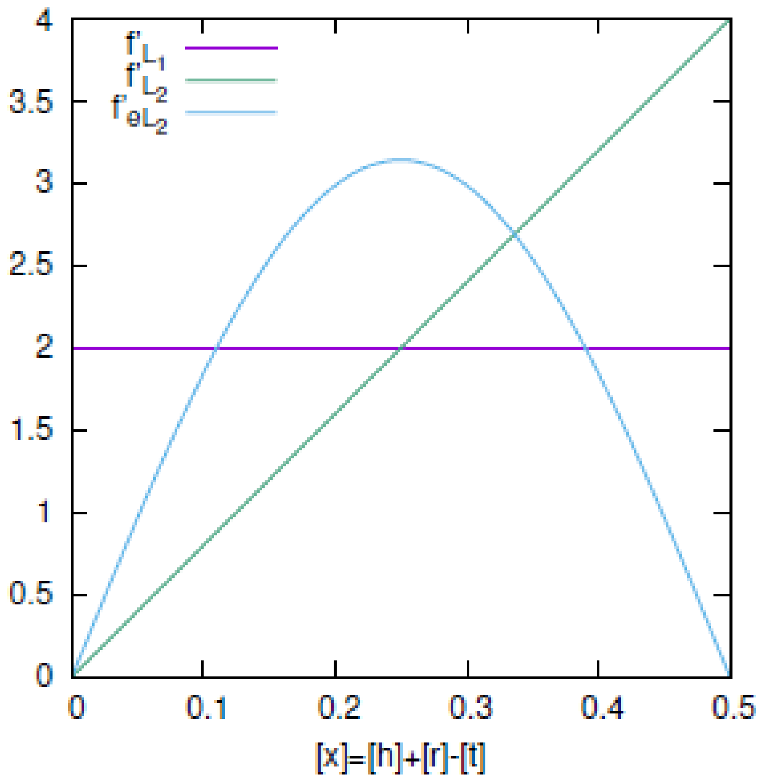


FIGURE 4.2: The graphs of normalized scoring functions and their derivatives for TorusE when $n = 1$. f'_{L_1}, f'_{L_2}, and f'_{eL_2} are derivatives of the scoring functions.

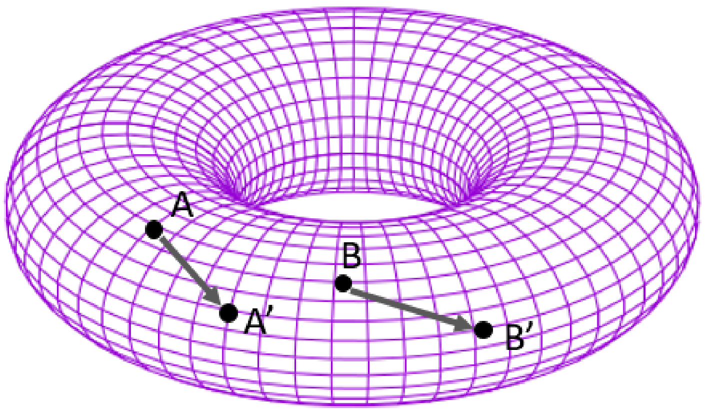


FIGURE 4.3: The image of embeddings on 2-dimensional torus obtained by TorusE. Embeddings of the triples (A, r, A') and (B, r, B') are illustrated. Note that $[\boldsymbol{A}'] - [\boldsymbol{A}]$ and $[\boldsymbol{B}'] - [\boldsymbol{B}]$ are similar on the torus.

Note that some extensions of TransE, such as TransG [94] and pTransE [87], can be applied directly to TorusE by changing the embedding space from a real vector space to a torus.

4.4.3.1 Calculation Technique on a Torus

Each embedding is represented by a point on a torus $[\boldsymbol{x}]$. Note that $\boldsymbol{x}$ itself is an n-dimensional vector and we use it to represent a point of the torus on a computer. By taking a fractional part of a vector, an embedding becomes one to one with a point of the torus and we can easily calculate the scoring functions. Let $p_{frac} : \mathbb{R} \to [0, 1)$ be the function taking a fractional part. Then, the distances are calculated as follows:

$$d_{L_1}([\boldsymbol{x}], [\boldsymbol{y}]) = \sum_{i=1}^{n} min(|p_{frac}(x_i) - p_{frac}(y_i)|, 1 - |p_{frac}(x_i) - p_{frac}(y_i)|) \tag{4.4}$$

$$d_{L_2}([\boldsymbol{x}], [\boldsymbol{y}]) = \sum_{i=1}^{n} min(|p_{frac}(x_i) - p_{frac}(y_i)|^2, (1 - |p_{frac}(x_i) - p_{frac}(y_i)|)^2) \tag{4.5}$$

$$d_{eL_2}([\boldsymbol{x}], [\boldsymbol{y}]) = \sum_{i=1}^{n} (2 - 2cos(2\pi(x_i - y_i))) \tag{4.6}$$

For example, let $\boldsymbol{x}$ and $\boldsymbol{y}$ be 3.01 and $0.99 \in \mathbb{R}^1$. Then $|p_{frac}(x_1) - p_{frac}(y_1)| = 0.98$ and $1 - |p_{frac}(x_i) - p_{frac}(y_i)| = 0.02$ hold. Hence we obtain $d_{L_1}([\boldsymbol{x}], [\boldsymbol{y}]) = 0.02$. As shown here, distance calculations on a torus are simple enough. In the case of d_{L_1}, just the *min* and p_{frac} functions are used in addition to the L_1 norm. TorusE can be trained through standard stochastic gradient descent using this technique and it is equivalent to RSGD with g_{torus}.

4.5 Attention KGLG

The concept of KGLG allows us to take any Lie group as the embedding space of a translation-based model. KGLG solves the problem caused by the embedding space. However, the problem with the strictness of the principle remains and it is difficult for KGLG to handle 1-N, N-1, and N-N relations. We propose a concept of embeddings called Attention Knowledge Graph Embedding on Lie Group (AKGLG) to solve this problem by generalizing KGLG. We show that state-of-the-art embedding models are instances of AKGLG.

4.5.1 Mechanism of Translation-based Models

In KGLG, relations and entities are represented by points on a Lie group G following the generalized principle $g_h +_G g_r = g_t$, where $+_G$ is the group operation of G and g_h, g_r, and g_t are embeddings on G of the head entity, relation, and tail entity, respectively, of an observed triple (h, r, t). G has a similarity function d_G that is used to score a triple (h, r, t) with $d_G(g_h +_G g_r, g_t)$. This principle allows KGLG to utilize first-order rules based on a path. If a rule $(r_1, r_2, ..., r_n) \Rightarrow r$ holds and there are enough groundings , i.e., a mapping from variables in the rule to E holding relations [6], in a knowledge graph, then the embeddings of these relations are trained to follow the equation:

$$\sum_{k=1}^{i} g_{r_k} := g_{r_1} +_G g_{r_2} +_G \cdots +_G g_{r_k} = g_r \tag{4.7}$$

We can get this equation by the sequential application of the principle. However, the principle seems too strict to compatibly embed various entities and relations. For example, if a head entity/relation pair have multiple valid tail entities and the embeddings perfectly follow the principle, then all of the tail entities have to be represented by the same point; this is undesirable because we need to distinguish different entities. TransR and TransD solve this problem by mapping entities to another space depending on the relation when the principle is applied, where the embeddings of relations are on codomains of these mappings. However, these models cannot utilize rules because the embeddings of relations are in different spaces and thus equation (1) has no meaning. Hence, we need to extend KGLG in a different way.

Table 4.2: List of existing embedding models that can be interpreted as instances of KGLG and AKGLG. Note that TransAt is more restricted than standard AKGLG.

Base Lie Group	KGLG	AKGLG
{-1,1}	Not proposed	DistMult
$\mathbb{R}$	TransE	TransH, TransAt
S^1	TorusE	ComplEx

4.5.2 Attentioned Knowledge Graph Embedding on Lie Group

The problem discussed in the previous section occurs because entities and relations are equally distributed throughout the embedding space of KGLG. We solve the problem by assigning an attention vector for each entity and relation to structuralize KGLG. The attention vector indicates the part of the embedding space where the information of the corresponding entity or relation is stored. We construct AKGLG on KGLG, whose embedding space is denoted by G. For AKGLG, entities e and relations r are represented by points $\boldsymbol{g}_e$ and $\boldsymbol{g}_r$, respectively, on $G' = G^n$. Then, we assign vectors w_e and $w_r \in [0, \infty)^n$ to each entity and each relation, respectively. The score of AKGLG of a triple (h, r, t) is formally defined as follows:

$$Score_{G'}(h, r, t) = (w_h \circ w_r \circ w_t) \cdot \boldsymbol{d}_G(\boldsymbol{g}_h +_G \boldsymbol{g}_r, \boldsymbol{g}_t)$$

where $\circ$ represents the element-wise multiplication operation, $\cdot$ represents the dot product operation, and $\boldsymbol{d}_G(\boldsymbol{g}_h +_G \boldsymbol{g}_r, \boldsymbol{g}_t)$ is an n-dimensional vector whose i-th element is equal to $d_G(h_i +_G r_i, t_i)$. Note that when the score of (h, r, t) is calculated, only the part of G' where the attentions of h, r, and t overlap, i.e., their attention values are simultaneously large enough, is considered. These attentions and the score function produce embeddings of entities and relations without conflict by properly separating the stored information.

4.5.3 Existing Embedding Models as Instances of AKGLG

Examples of AKGLG and KGLG are shown in Table 4.2.

We can consider a KGLG on a group $G_1 = \{-1, 1\}$, where the group operation is the standard real number multiplication operation. We define the similarity function on G_1 as $d_1(x, y) = 1(\text{if}\, y = x), -1(\text{if}\, x \neq y)$. Then, we can consider a KGLG on G_1^n that extends the similarity function of G_1, i.e., it takes the sum after the element-wise calculation. G_1^n is not an infinite set but can utilize simple rules. An AKGLG on G_1^n is equivalent to DistMult. Here, each entity or relation has its attention vector and its embedding on G_1^n. We can obtain a vector representation for each relation or entity element-wisely multiplying the attention vector and the embedding as a real vector. The triple score of a DistMult based on these vector representations is the same as the score of AKGLG, i.e.:

$$\begin{aligned} Score_{G_1^n}(h,r,t) &= (w_h \circ w_r \circ w_t) \cdot \boldsymbol{d}_{G_1}(\boldsymbol{g}_h +_{G_1} \boldsymbol{g}_r, \boldsymbol{g}_t) \\ &= (w_h \circ \boldsymbol{g}_h) \circ (w_r \circ \boldsymbol{g}_r) \cdot (w_t \circ \boldsymbol{g}_t) \end{aligned}$$

where the right side is the score of DistMult.

We can consider a KGLG on a circle S^1 as a subset of $\mathbb{C}$ whose elements have a magnitude of 1 where the group operation $+_{S^1}$ is the standard complex number multiplication operation. We define the similarity function as $d_{S^1}(x,y) = Re(x +_{S^1} \overline{y})$. Then, we can consider a KGLG on S^n that extends the score function of S^1; this is equivalent to TorusE [7]. An AKGLG on S^n is equivalent to ComplEx. We can obtain a complex vector representation for each relation or entity multiplying them. The triple score of ComplEx based on these vector representations is the same as the score of AKGLG, i.e.:

$$\begin{aligned} Score_{S^n}(h,r,t) &= (w_h \circ w_r \circ w_t) \cdot \boldsymbol{d}_{S^1}(\boldsymbol{g}_h +_{S^1} \boldsymbol{g}_r, \overline{\boldsymbol{g}_t}) \\ &= Re((w_h \circ \boldsymbol{g}_h) \circ (w_r \circ \boldsymbol{g}_r) \cdot (w_t \circ \overline{\boldsymbol{g}_t})) \end{aligned}$$

where the right side is the score of ComplEx.

We can consider a KGLG on $\mathbb{R}$ where the group operation is the standard summation operation. We define the similarity function as $d_{\mathbb{R}}(x,y) = -(x-y)^2$. Then, we can consider a KGLG on $\mathbb{R}^n$ that extends the score function of $\mathbb{R}$; this is equivalent to TransE. An AKGLG on $\mathbb{R}^n$ has not been proposed. However, its restricted versions are TransH and TransAt. In TransAt, the attention vectors for entities are fixed to the vector whose elements are all 1 and the attention vectors for relations are restricted to a vector whose elements are 0 or 1.

As we have shown, knowledge graph embedding models can be unified using the concept of AKGLG. These models work based on the translation principle.

4.6 Experiments

4.6.1 Experimental Settings

We evaluated KGLG and AKGLG, including TorusE, from two perspectives:

- Experiment 1: The link prediction tasks on standard datasets to evaluate embeddings obtained by TorusE and TransE.
- Experiment 2: Measuring the time it takes to train TorusE to see the scalability of TorusE.

4.6.1.1 Implementation Details

For our experiments, we employed two types of cost functions.

One of the cost functions we employed was based on the score function with margin loss. Because the datasets contained only positive triples, we employed the ”Bern” method [57] for negative sampling. Then, the score function is as follows:

$$\mathcal{L} = \sum_{(h,r,t)\in\Delta} \sum_{(h',r,t')\in\Delta'_{(h,r,t)}} [\gamma + f(h,r,t) - f(h',r,t')]_+ \tag{4.8}$$

This score function is traditionally used for translation-based model. Regularization for TransE was the restriction of entities on a sphere the same as the original version. Regularization is not required for TorusE, in contrast with the other embedding methods. We conducted experiments for TransE and TorusE with this cost function.

The different types of cost functions have been used for bilinear models from translation-based models. As we showed above, a torus is a sub manifold of a complex vector space. Hence, we could employ this cost function for TorusE. For this cost function, all possible negative triples are generated for a training triple changing the head entity or the tail entity of it. The cost function was defined as the total of cross-entropy loss with the scores of a training triple and its all possible negative triples applying the softmax function. We conducted experiments for TorusE, DistMult, and ComplEx with this cost function. For DistMult and ComplEx, we employed L_3 regularization. This cost function and the regularization was recently proposed by Lacroix et al. [95]. We could not apply this cost function to TransE, because TransE cannot efficiently utilize a GPU, hence, it was too slow to finish training.

For each epoch, we randomly separated training triples into one-hundred groups, and embedding parameters were updated for each group.

4.6.1.2 Optimization Details

It is known that the higher the dimension of an embedding model is, the more efficient the model is for link prediction. Hence, we set the dimension as high as possible. For a model with the margin loss cost function, we set the dimension as 10,000. For a model with the cross-entropy loss cost function, we set the dimension as 2,000. The other hyper parameters were determined by conducting a grid search for each dataset.

Table 4.3: MRR and HITS@n scores for link prediction tasks with WN18, FB15k, WN18RR, and FB15k-237 datasets. The highest result for each column is shown in bold. The results for ConvE were reported by Dettmers et al. [78], and those for PRA were reported by Liu et al. [52].

	WN18				FB15k				WN18RR				FB15k-237			
	MRR	HITS@			MRR	HITS@			MRR	HITS@			MRR	HITS@		
Model		1	3	10		1	3	10		1	3	10		1	3	10
PRA	0.458	0.422	–	0.481	0.336	0.303	–	0.392	–	–	–	–	–	–	–	–
GPro	0.950	0.946	0.954	0.959	0.793	0.759	0.810	0.858	0.467	0.430	0.485	0.543	0.229	0.163	0.250	0.360
GRank (fdMAP)	0.950	0.946	0.954	0.958	0.842	0.816	0.856	0.891	0.470	0.437	0.482	0.539	0.322	0.239	0.352	0.489
TransE	0.397	0.040	0.745	0.923	0.414	0.247	0.534	0.688	0.397	0.040	0.745	0.923	0.414	0.247	0.534	0.688
DistMult	0.922	0.891	0.952	0.956	0.840	0.802	0.865	0.906	0.460	0.416	0.472	0.548	0.354	0.260	0.389	0.543
ComplEx	**0.951**	0.945	**0.955**	**0.962**	**0.856**	**0.827**	**0.872**	**0.909**	0.476	0.429	**0.493**	**0.564**	**0.365**	**0.269**	**0.401**	**0.555**
ConvE	0.942	0.935	0.947	0.955	0.745	0.670	0.801	0.873	0.46	0.39	0.43	0.48	0.316	0.239	0.350	0.491
TorusE (margin)	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832	0.452	0.422	0.464	0.512	0.305	0.217	0.335	0.484
TorusE (cross entropy)	**0.951**	**0.947**	0.954	0.960	0.810	0.768	0.835	0.884	**0.477**	**0.439**	0.490	0.551	0.346	0.252	0.380	0.535

4.6.2 Experiment 1: Link Prediction Tasks

4.6.2.1 How to Rank Entities for a Query

We conduct the link prediction task in the same way reported in the paper of TransE [56]. For a given test query $(h, r, ?)$, the score (h, r, e) is calculated for each $e \in E$ and it is used to rank e as an answer of the query. For a given test query $(?, r, t)$, the score (e, r, t) is calculated for each $e \in E$ and it is used to rank e as an answer of the query.

4.6.2.2 Results

Overall Results The results of the link prediction tasks are shown in Table 4.3.

First of all, we can see the efficiency of a model of the cross-entropy by comparing TorusE (margin loss) with TorusE (cross-entropy). The main reason for this huge gap is that all possible negative examples generated by PCA for the cross-entropy loss cost function.

We can see the effect of changing the embedding space by comparing the results obtained by TransE with the results by TorusE (cross-entropy loss). Note that those models employed the same cost function. The results obtained by TorusE are generally better than The results obtained by TransE on all metrics. The results show the problem with less expressiveness of TransE is really relaxed by changing the embedding space from an Euclid space. However, we are not completely sure from these results which part is solved, because embedding on a torus can possibly solve two types of problems: much strictness of the principle and warping of embeddings by the regularization. This question is discussed in the following section with results in detail.

We can see the effect of the attention system of AKGLG by comparing TorusE (cross entropy) with ComplEx because their difference is just the attention system. For WN18 and WN18RR, TorusE (cross entropy) and ComplEx is competitive. On the other hand, the difference between the scores is noticeable on FB15k and FB15k-237. This is because FB15k and FB15k-237 contain a large amount of relations comparing with WN18 and WN18RR: WN18RR only contains thirteen relations but FB15k contains 1,345 of relations. We think if a knowledge graph contains more relations, then it becomes more difficult to embed the relations so that they fill the principle. Hence, the attention system of AKGLG efficiently allocates a part of the embedding space to each relation to realize them compatibly.

Detailed Results The details of the MRR for each relation on WN18 are shown in Table 4.4. We can clearly see the effect of changing the embedding space from a Euclid

Table 4.4: Details of "filtered" MRR on WN18. The results are listed separately for each relation contained in the dataset.

Relation name	Rel. Category	TorusE	TransE
similar_to	1-to-1	**1.000**	0.242
verb_group	1-to-1	**0.974**	0.283
hyponym	1-to-N	**0.956**	0.379
member_meronym	1-to-N	**0.931**	0.433
instance_hyponym	1-to-N	**0.961**	0.626
has_part	1-to-N	**0.944**	0.417
member_of_domain_topic	1-to-N	**0.944**	0.502
member_of_domain_usage	1-to-N	**0.917**	0.270
member_of_domain_region	1-to-N	**0.885**	0.358
hypernym	N-to-1	**0.957**	0.376
member_holonym	N-to-1	**0.942**	0.438
instance_hypernym	N-to-1	**0.961**	0.680
part_of	N-to-1	**0.947**	0.415
synset_domain_topic_of	N-to-1	**0.921**	0.536
synset_domain_usage_of	N-to-1	**0.940**	0.182
synset_domain_region_of	N-to-1	**0.919**	0.197
derivationally_related_form	N-to-N	**0.951**	0.362
also_see	N-to-N	**0.626**	0.257

Table 4.5: Detailed results by category of relationship. We compare "filtered" HITS@10 on FB15k classifying relations into 4 categories: 1-to-1, 1-to-N, N-to-1, and N-to-N. The results of TransH were reported by Wang et al. [57], and the results of TransR were reporetd by Lin et al. [58].

Tasks	Predicting Head				Predicting Tail			
Relation Category	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
TransE	**0.806**	**0.936**	0.312	0.735	**0.805**	0.421	**0.938**	0.772
TransH	0.668	0.876	0.287	0.645	0.655	0.398	0.833	0.672
TransR	0.788	0.892	0.341	0.692	0.792	0.374	0.904	0.721
TorusE	0.775	0.906	**0.479**	**0.859**	0.773	**0.592**	0.893	**0.890**

space to a torus here. TorusE generally overperforms TransE for all results. We think this happened because we could get rid of the regularization by changing the embedding space.

In the relations of WN18, "similar_to" and "verb_group" are symmetric relations and the results about those relations are especially low. On the other hand, TorusE could perform link prediction almost perfectly on those relations, which shows TorusE can learn symmetric relations while TransE cannot do it.

The details of the MRR for each relation on FB15k are shown in Table 4.5. TransH [57] and TransR [58] were developed because the authors thought the principle of TransE had a problem to predict triples with a relation which has multiple heads per tail or multiple tails per head. TorusE employs the same principle as TransE so that TorusE might have the same problem with TransE. Hence, we classified relations into four categories: 1-to-1, 1-to-N, N-to-1, and N-to-N relations to see detailed results and compare with published results. A relation is classified as 1-to-1 if each head entity appears with at most one tail entity, 1-to-N if each head entity appears with multiple tail entities, N-to-1 if multiple head entities appear with the same tail entity, and N-to-N if multiple head entities appear with multiple tail entities. We employ the criteria introduced by Bordes et al. [56] to classify relations: we compute the average number of heads per tails (hpt_r) and the average number of tails per head (tph_r) for each relation r. If $hpt_r,\ tph_r < 1.5$, r is considered as 1-to-1. If $hpt_r < 1.5$ and $tph_r \geqq 1.5$, r is considered as 1-to-N. If $hpt_r \geqq 1.5$ and $tph_r < 1.5$, r is considered as N-to-1. If $hpt_r,\ tph_r \geqq 1.5$, r is considered as N-to-N. We can see TorusE performs better than TransH and TransR on all categories of relations, which means their modification from TorusE is not so effective for 1-to-N, N-to-1, and N-to-N relations. Some of the results of TransE is better than those of TorusE. We think this is because the best hyperparameters are different depends on categories and the hyperparameters are tuned to get the highest MRR averaging the score of all relations.

On both dataset, TorusE performs well with 1-to-N, N-to-1, and N-to-N relations compared with other models. We think that is because the principle itself is not actually problematic as pointed before. It is definitely impossible to follow the principle completely on such relations, but to follow the principle completely is not necessary to deal with the link prediction task. Because the task employs rankings of entities, a model for the task is adequate when the correct entities are located in higher ranks than the incorrect entities are, even if the correct entities are not at the top ranks.

4.6.3 Experiment 2: Scalability of TorusE

4.6.3.1 Evaluation Protocol

To evaluate the scalability of TorusE, we measured the time it took to train TorusE for one epoch by changing the dimensions of the model on WN18 and FB15k with the margin loss cost function. Other hyperparameters were set to optimal configurations for Experiment 1.

4.6.3.2 Results

The calculation times of TorusE and TransE are shown in Fig. 4.4. They were measured by using a single GPU (NVIDIA Titan X). The complexities of TorusE and TransE are theoretically the same and the lowest among all models at $\mathcal{O}(n)$. For both models, the calculation time is considered a first-order equation of the dimension. However, a large gap existed between the empirical calculation times of these models.

For the WN18 dataset, TransE took 55.6 seconds to complete one epoch when the dimension was 10,000. On the other hand, TorusE took 4.0 seconds when the dimension was 10,000, and so TorusE is eleven times faster than TransE. This is mainly due to the regularization of TransE, because the normalizing calculations of all entity embeddings are time-consuming.

For FB15k, TransE took 29.4 seconds to complete one epoch and TorusE took 16.8 seconds when the dimension was 10,000, and so TorusE is faster than TransE. FB15k contains more triples than WN18 does. Hence, TorusE took more time for FB15k than WN18. However, TransE took less time. This was because of the number of entities contained in the datasets. The number of entities of WN18 is much more than the number of entities of FB15k.

These models were trained for 500 epochs in an experiment. So, the total time was about 30 minutes and 2 hours 30 minutes for TorusE to finish training on WN18 and FB15k respectively. We also measured the calculation times of ComplEx with the implementation by Trouilon et al. [96]. The calculation times of ComplEx were 1 hour 15 minutes and 3 hours 50 minutes on each dataset with 150 and 200 dimensions on the same GPU, and TorusE was faster than it.

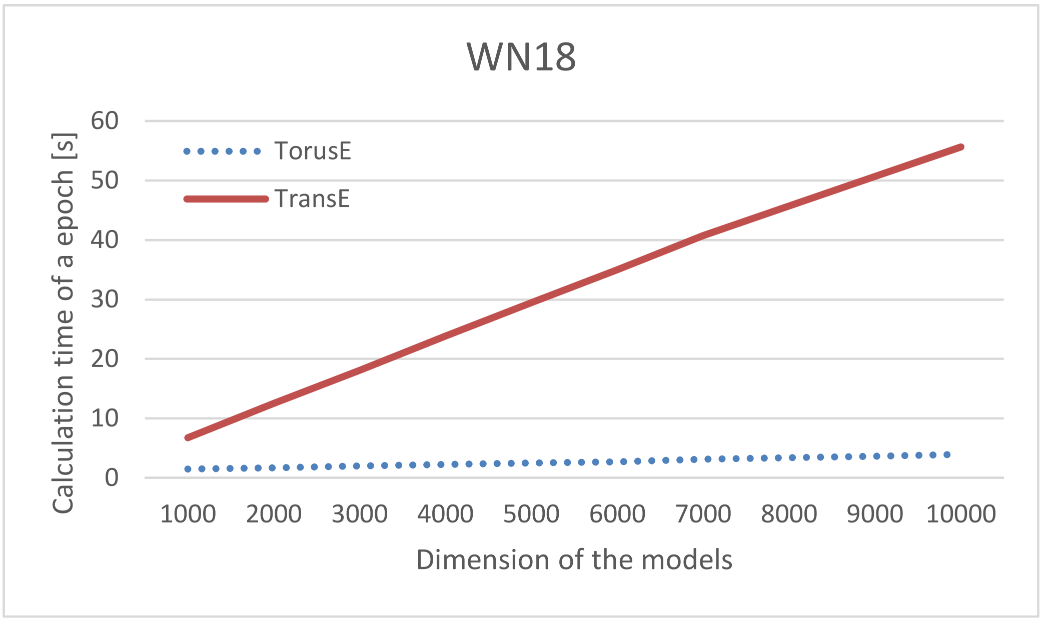
4.7 Conclusions

Our contributions in this chapter are as follows.

- We pointed out the new problems with the strictness of the principle of TransE: Regularization conflicts with the principle and TransE cannot deal with som relations such as symmetrical relations.
- We proposed the knowledge graph embedding on a Lie group (KGLG) model, which is the more comprehensive model than previous translation-based models. KGLG lets us choose any Lie group as an embedding space for translation-based knowledge graph embedding. We also proposed TorusE, which is an instance of

KGLG and embeds a knowledge graph on a torus. It is not bothered with the regularization problem because it does not require any regularization, unlike other models. TorusE also can deal with symmetrical relations. TorusE outperformed state-of-the-art models for link prediction tasks and it was experimentally shown to be faster than TransE.

- We further generalize KGLG to Attention KGLG (AKGLG) in which attention vectors are additionally assigned to entities and relations to indicate a part of the embedding space to store the information of themselves. AKGLG relaxes the strictness of the principle. We also showed that many state-of-the-art models are included in AKGLG, i.e. those models are actually translation-based models.



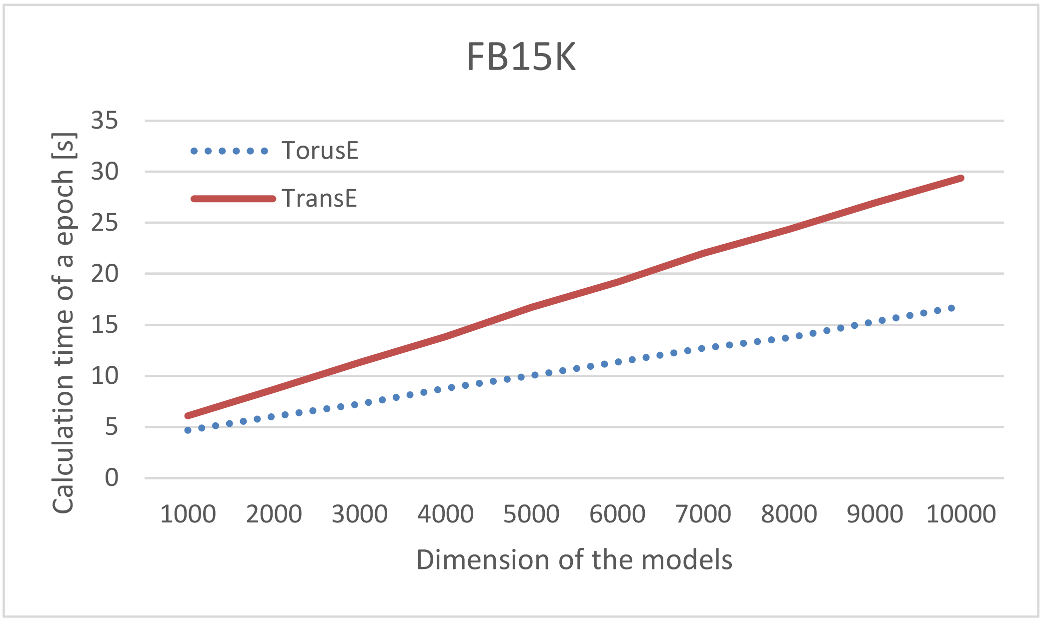


FIGURE 4.4: Calculation time of TorusE and TransE on WN18 and FB15k

Chapter 5

Integration of Approaches for Link Prediction

5.1 Overview

In this chapter, we integrate approaches for link prediction and make them efficiently work together. The main contributions in this chapter are as follows:

- We propose a method that evaluates rules based on embeddings of AKGLG.
- We propose a framework for combining approaches for link prediction to compensate for the disadvantages of each approach.
- We evaluate the proposed method and framework in terms of calculation time and link prediction accuracy with standard datasets. It is shown that our method can find useful rules faster than traditional rule evaluation models and that our framework outperforms other models in terms of link prediction.

The remainder of this section is organized as follows. In Section 5.2, we discuss the remaining problems with our work. In Section 5.3, we propose a method for evaluating and selecting useful rules for link prediction based on embeddings of AKGLG. In Section 5.4, we propose a framework for combining many approaches. In Section 5.5, we present an experimental study that compares our method and framework with baseline results for benchmark datasets. In Section 5.6, we present the conclusions.

5.2 Remaining Limitations of Approaches for Link Prediction

While we developed each of a rule-based model and an embedding model to solve problems with them, there are still some problems remain. We have shown instances of AKGLG are efficient and we know the mechanism of them but we still do not have a way to interpret the embeddings obtained by AKGLG. Additionally, embedding models generally suffer from and the mixing of all information in embeddings even though a certain relation may require only a few simple rules to precisely predict.

On the other hand, GRank describes the reason for output by showing filled rules. However, GRank has some problems such as slow calculation speed and non-integrated rules, i.e., rules are discrete and it is not clear that how we combine them. Another problem is the limited search space. More complex rules or those that include constants are not considered because such rules are often useless and they greatly expand the search space. However, such rules may useful for link prediction.

PRA [50, 51] constructs logistic classification models for each relation based on features that represent the existence of a particular path between two entities. However, the models lack an efficient way to select paths for features. This problem can be overcome using rule evaluation models.

The main problems of observed feature models are slowness and a limited rule search space. To solve these problems, we propose a method for evaluating rules based on embeddings in Section 5.3.2 and a framework for combining embedding models and observed feature models in Section 5.4. Some models [97, 98] employ rules extracted by traditional rule evaluation models to obtain better embeddings. In contrast, we refine the information in embeddings based on the observed features. This is done because embedding models already can capture rules but cannot order information.

5.3 Rule Evaluation on Embeddings

In the previous chapter, we proposed AKGLG. We showed that both KGLG and AKGLG utilize rules. In this section, we propose a method for evaluating rules based on this idea. This method allows us to interpret embeddings; that is, we can know what kind of rules are learned and used for link prediction. Additionally, the method evaluates rules faster than traditional rule evaluation methods.

5.3.1 Rule Evaluation on KGLG

In chapter 3, the rules are limited to the form $r_1(x_1, x_2) \wedge r_2(x_2, x_3) \wedge \cdots \wedge r_n(x_n, x_{n+1}) \Rightarrow r(x_1, x_{n+1})$, where x_i is a variable for reduce the searchspace of rules; hereafter, we refer to a rule as $(r_1, r_2, ..., r_n) \Rightarrow r$. We proposed GRank that assigns two confidence scores to a rule, where one is the confidence of predicting a tail entity and the other is that of predicting a head entity. Hence, we assume that the rule $(r_1, r_2, ..., r_n) \Rightarrow r$ is used to predict a tail entity given a head entity h, i.e. used to answer the query $(h, r, ?)$. The rule for predicting a head entity of r based on the same path is described by its inverse form: $(r_n^{-1}, r_{n-1}^{-1}, ..., r_1^{-1}) \Rightarrow r^{-1}$. We also suppose that a KGLG learns two embeddings for each relation, as proposed by Lacroix et al. [95]. One of the embeddings is trained and used to predict the tail entities of the corresponding relation; we denote this embedding as g_r. The other is trained and used to predict head entities; we denote this embedding as $g_{r^{-1}}$.

If $(r_1, r_2, ..., r_n) \Rightarrow r$ is really useful for predicting tail entities related with r, then the KGLG learns embeddings in such a way that $g_{(r_1, r_2, ..., r_n)} = \sum_{k=1}^{i} g_{r_k}$ and g_r are similar. Therefore, we propose to evaluate the rule by measuring the similarity between $g_{(r_1,r_2,...,r_n)}$ and g_r. KGLG has a score function d_G, which is used to estimate the validity of a triple, as discussed in Chapter 5. For example, TransE can have L1 norm or the square of L2 norm and TorusE has f_{L_1}, f_{L_2}, and f_{eL_1} as the score function. We can use the function to measure the similarity between $g_{(r_1,r_2,...,r_n)}$ and g_r. The confidence score of a rule $(r_1, r_2, ..., r_n) \Rightarrow r$ is formally written as follows:

$$conf((r_1, r_2, ..., r_n) \Rightarrow r) = -d_G(g_{(r_1,r_2,...,r_n)}, g_r) \tag{5.1}$$

However, the confidence score obtained using this equation is not so reliable because the embeddings obtained by KGLG may not be as good as those obtained by AKGLG. In the following section, we define the confidence score on AKGLG to evaluate a rule more properly.

5.3.2 Rule Evaluation on AKGLG

5.3.2.1 Path Representation

For AKGLG, each entity and relation is represented by its embedding on a Lie group and its attention vectors. We want to represent a path to compare it with the representation of a relation and evaluate a rule on AKGLG. For the embeddings on a Lie group, we can obtain the path embedding on a Lie group by group multiplying the relation embeddings on the path, as we did in the previous section. For the attention vectors, we take the element-wise geometric mean of the attention vectors on the path. We employ the geometric mean because we want the i-th element to be zero if one of the i-th elements of the attention vectors is 0 because the information does not propagate through the corresponding dimension. We then normalize the path attention vector because the magnitude of attention vectors likely represents the appearance frequency in the knowledge graph; we thus do not want to take the magnitude into account. The attention vector of a path $(r_1, r_2, ..., r_n)$ is formally defined as follows:

$$w_{(r_1,r_2,...,r_n)} = \frac{(w_{r_1} \circ w_{r_2} \circ \cdots \circ w_{r_n})^{1/n}}{||(w_{r_1} \circ w_{r_2} \circ \cdots \circ w_{r_n})^{1/n}||_2} \tag{5.2}$$

5.3.2.2 Evaluation of Rules

The similarity of the embeddings on a Lie group is calculated in the same way as that for KGLG. We take the attention vectors into account in a way similar to that used for calculating the score of a triple. The confidence score of a rule $(r_1, r_2, ..., r_n) \Rightarrow r$ on

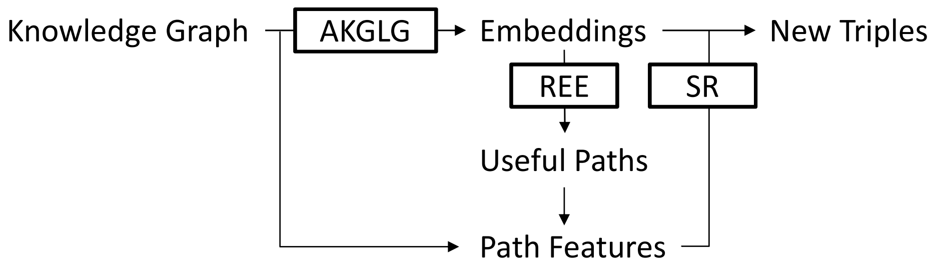


FIGURE 5.1: Procedure of PBF. In this figure, SR stands for softmax regression.

AKGLG is formally defined as follows:

$$conf((r_1, r_2, ..., r_n) \Rightarrow r) = -(w_{(r_1,r_2,...,r_n)} \circ w_r) \cdot \boldsymbol{d}_G(\boldsymbol{g}_{(r_1,r_2,...,r_n)}, \boldsymbol{g}_r)$$

where $\boldsymbol{d}_G(\boldsymbol{g}_{(r_1,r_2,...,r_n)}, \boldsymbol{g}_r)$ is the n-dimensional vector whose i-th element is equal to $d_G(g_{(r_1,r_2,...,r_n),i}, g_{r,i})$. We refer to this evaluation method as the rule evaluation based on embeddings (REE).

5.4 Framework for Combining Link Prediction Approaches

In this section, we propose a framework to exploit the advantages of various approaches for link prediction, as discussed in Section 5.2. The outline of the framework is as follows:

1. Obtain embeddings of entities and relations by employing AKGLG.
2. Extract useful paths for each relation for link prediction by evaluating corresponding rules with traditional rule evaluation models or REE.
3. Construct a softmax regression model for each relation in a way similar to PRA. The features used for training and prediction are obtained by counting the number of groundings of extracted paths, as done by GRank.
4. Perform link prediction by taking the weighted sum of the scores of the embedding model and the softmax regression models.

We refer to this framework as the path-based framework (PBF). The flowchart of PBF with REE is shown in Figure 5.1. We discuss the details of steps 2 and 3 below.

5.4.1 Path Extraction using REE

We can select useful paths for each relation using traditional rule evaluation models. However, traditional models are time-consuming because the number of candidate paths increases exponentially with their size, and so do their groundings. We expect that REE can evaluate rules faster because it does not need to consider groundings.

We first extract candidate rules by finding positive groundings for REE. This can be done much faster than evaluating rules with traditional methods because we do not need to find negative groundings of rules. We restrict groundings to “injective” , i.e., one entity can appear at most once, as in Chapter 3. This restriction allows us to find groundings on the converted simple graph (i.e., there are no multiple edges between entities) using the following procedure. In the explanation, we limit the path length to be an odd number $2k - 1$ for simplicity and we suppose that the entities are ordered (i.e., labeled by different integers).

1. Convert a knowledge graph $KG = \{(h, r, t)\}$ to the simple graph $\{(h, r) | (h, r, t) \in KG\}$.
2. For each entity e, find all cycles in which e is the smallest entity as follows:
   - Find all entity paths whose length is at most k under the condition that all entities except e on paths are larger than e in terms of the order.
   - For each pair of entity paths whose last entities are the same, make a cycle that concatenates them and ensure that there is no entity duplication.
3. For each path p, if there is a cycle that is the groundings of the rule $p \Rightarrow r$, then add the rule to the candidate rules to be evaluated by REE.

Note that we can find cycles with no duplication and conduct the computations using parallel processing because the procedure for each entity is independent. Then, we evaluate candidate rules with REE and select a fixed number of rules for each relation. The body paths of these rules are used to construct softmax regression models.

5.4.2 Softmax Regression Model

In this section, we construct a softmax regression model for each relation following PRA. We first construct training queries for each relation. Training queries for relation r are formally defined as $\{(h, r, ?) | (h, r, t) \in KG\}$. Each element of the feature vector of a query $(h, r, ?)$ and an entity e corresponds to one of the extracted paths and its value is equal to the multiplicity of the corresponding path (i.e., the number of groundings of the path that start from h and end at e). We employ multiplicity because it is efficient, as shown by GRank. However, the magnitude of path multiplicity greatly differs. Hence, we need to rescale feature vectors to obtain a good model. The i-th element of the final feature vector for a query $(h, r, ?)$ and an entity e is formally defined as follows:

$$v_{((h,r,?),e),i} = mul(h, e, p_i)/M \tag{5.3}$$

where p_i is the path indexed by i for r, $mul(h, e, p_i)$ is the number of groundings of p_i that start from h and end at e, which is equivalent to $score(p_i, (h, e))$ defined for GRank [6], and $M = max\{mul(h, e', p_i)|e' \in E\}$.

Next, we describe the softmax regression models. The softmax regression model for r has a parameter vector θ_r for training. The score of the triple (h, r, e) for query $(h, r, ?)$ is calculated by taking the dot product of θ_r and $v_{((h,r,?),e)}$. For each answer for a query, a fixed number of entities are randomly selected for negative examples. We employ cross-entropy for the loss function. The loss function for the model of a relation r is formally written as follows:

$$L_r = \sum_{(h,r,t)\in KG} -log(\frac{exp(\theta_r^\top \cdot v_{((h,r,?),t)})}{\sum_{e\in NE_{(h,r,t)}\cup\{t\}} exp(\theta_r^\top \cdot v_{((h,r,?),e)})}) \tag{5.4}$$

where $NE_{(h,r,t)}$ is the set of randomly selected negative entities. Note that $v_{((h,r,?),e)}$ can often be the zero vector. We ignore the positive triple (h, r, t) if $v_{((h,r,?),t)}$ is the zero vector and we do not select the entity e as a negative example if $v_{((h,r,?),e)}$ is the zero vector.

5.5 Experiments

In this section, we conduct experiments to evaluate REE and PBF. REE is directly compared with traditional rule evaluation models including GRank in terms of calculation time and link prediction accuracy. PBF is compared with other models including state-of-the-art embedding models in terms of link prediction accuracy.

5.5.1 Protocol of Link Prediction Task

Next, we describe how to obtain rankings using the methods. For PBF, we can get the rankings of entities for a query $(h_t, r_t, ?)$ by calculating the score of triples (h_t, r_t, e) for each $e \in E$. For REE, we follow the settings of experiments for GRank in Chapter 3 to obtain ranking entities from the extracted rules. We extract 1,000 rules for each relation (including the inverse of a relation) and each of the rules is used to obtain entity rankings for a query by counting its groundings. The final rankings of entities are obtained by concatenating the rankings from each rule.

5.5.2 Experimental Settings

We employed ComplEx in the experiments as an instance of AKGLG. ComplEx was used to obtain embeddings to extract rules in the first experiment and answer test queries in the second experiment. Note that each entity and relation is represented by a complex vector $\boldsymbol{c} \in \mathbb{C}^n$. This vector is decomposed into attention vector $\boldsymbol{abs}(\boldsymbol{c})$ and point on the torus $(\mathbf{1}/\boldsymbol{abs}(\boldsymbol{c})) \circ \boldsymbol{c}$, where $\boldsymbol{abs}(\boldsymbol{c})$ is a real vector whose i-th element is equal to $abs(c_i)$. The settings and hyperparameters of ComplEx are those given by Lacroix et al. [95], with a dimension of 2,000 and L3 regularization. We also conducted experiments on DistMult and TorusE with the same settings for a fair comparison.

We employed GRank with fdMAP [6] for comparison with REE and for use in PBF. The limit of the path length for rules was selected from $\{1, 2, 3\}$ based on the MRR of link prediction on the validation triples.

For the softmax regression models in PBF, we set the number of extracted paths for each relation to 100. We employed stochastic gradient descent for training. We used L_2 regularization and the coefficient of the regularization. The learning rate was selected from $\{0.1, 0.01, 0.001\}$ depending on the MRR of link prediction on the validation data . We set the batch size to 100 and trained each model using 500 batches.

The weight for the final step of PBF was selected from $\{0, 0.1, \cdots, 1\}$ based on the MRR of link prediction on the validation data .

5.5.3 Experimental Results for REE

5.5.3.1 Calculation Time

Table 5.1 shows the calculation times for REE, including the rule candidate selection method (see Section 5.1), and GRank for the relatively large datasets FB15k and FB15k-237. It took GRank less than one minute to finish for WN and WN18RR. Note that we used an Intel Xeon Gold 6140 CPU (18 cores) for running GRank and rule candidate selection and an Intel Xeon E5-1620 CPU (4 cores) and a GPU (Nvidia Titan X) for running REE. The maximum path size was 3 in this experiment. The results show that REE is more efficient than GRank, especially for FB15k. FB15k is a denser graph than FB15k-237 and thus the number of groundings of rules is larger. That makes GRank even slower. As a result, REE is 30 times faster than GRank.

Table 5.1: Calculation time of REE and GRank.

	FB15k	FB15k-237
GRank	181,352s	2,486s
REE	5,504s	544s

We did not take the computation time of ComplEx, which is relatively short, into account. In our experiment, one epoch of training for ComplEx on FB15k takes about 110 s and that 25 epochs are sufficient. For hyperparameter tuning, one epoch is sufficient. Hence, about 5,000 seconds are sufficient to obtain embeddings; this is far faster than rule evaluation using GRank. To estimate rules faster, we can employ low-dimensional space, as described in Section 5.5.3.3.

5.5.3.2 Link Prediction Task

The results of the link prediction tasks for REE are shown in Tables 5.2. The results reported in previous studies are included for comparison. We focus on the comparison of REE with traditional rule evaluation models GRank and GPro [6], which is a modified version of AMIE for link prediction.

The results show that GRank is generally better than the other rule evaluation models because it considers the groundings of rules in great detail, whereas our method evaluates them indirectly. REE is competitive with GPro; it has worse results for WN18 and WN18RR and better results for FB15k and FB15k-237. This shows REE can properly evaluate rules.

We believe that GRank is more reliable in evaluating rules, so we compare the rules obtained by GRank and REE based on GRank. On the WN18 dataset, the rules obtained for each relation by GRank consist of one relatively high-confidence rule and other relatively low-confidence rules. Among these 36 high-confidence rules with GRank, 34 of them also had the highest rating with REE. This result suggests the similarity of the evaluation by GRank and REE.

FB15k also showed similarity in obtained rules, but it was found that REE evaluation tended to overestimate the rule of "go and return" such as $(r', r'^{-1}, r) \Rightarrow r$. This is because r' is inverse of r'^{-1}, so the expression on the Lie group is $g_{r'} = -g_{r'^{-1}}$ and therefore $g_{r'} + g_{r'^{-1}} + g_r = g_r$ holds. We think that this problem can be dealt with by the attention system of AKGLG, but in the case of FB15k, we often see such rules because FB15k particularly contains many relations.

Table 5.2: MRR and HITS@n scores for link prediction tasks with WN18, FB15k, WN18RR, and FB15k-237 datasets. The highest result for each column is shown in bold. The results for ConvE were reported by Dettmers et al. [78], and those for PRA were reported by Liu et al. [52].

	WN18				FB15k				WN18RR				FB15k-237			
	MRR	HITS@			MRR	HITS@			MRR	HITS@			MRR	HITS@		
Model		1	3	10		1	3	10		1	3	10		1	3	10
GPro	0.950	0.946	0.954	0.959	0.793	0.759	0.810	0.858	0.467	0.430	0.485	0.543	0.229	0.163	0.250	0.360
GRank (fdMAP)	0.950	0.946	0.954	0.958	0.842	0.816	0.856	0.891	0.470	0.437	0.482	0.539	0.322	0.239	0.352	0.489
REE	0.942	0.940	0.944	0.946	0.819	0.801	0.828	0.852	0.437	0.403	0.452	0.504	0.288	0.215	0.316	0.432
TorusE	0.951	0.947	0.954	0.960	0.810	0.768	0.835	0.884	0.477	0.439	0.490	0.551	0.346	0.252	0.380	0.535
DistMult	0.922	0.891	0.952	0.956	0.840	0.802	0.865	0.906	0.460	0.416	0.472	0.548	0.354	0.260	0.389	0.543
ComplEx	0.951	0.945	0.955	0.962	0.856	0.827	0.872	0.909	0.476	0.429	0.493	0.564	0.365	0.269	0.401	0.555
ConvE	0.942	0.935	0.947	0.955	0.745	0.670	0.801	0.873	0.46	0.39	0.43	0.48	0.316	0.239	0.350	0.491
PRA	0.458	0.422	–	0.481	0.336	0.303	–	0.392	–	–	–	–	–	–	–	–
PBF with GRank	**0.953**	**0.948**	**0.956**	**0.963**	**0.870**	**0.845**	**0.882**	**0.915**	**0.494**	**0.453**	**0.509**	**0.576**	**0.376**	0.282	**0.413**	**0.564**
PBF with REE	0.952	0.947	0.955	0.962	0.868	0.844	0.880	0.914	0.491	0.450	0.506	**0.576**	**0.376**	**0.284**	0.411	0.558

Table 5.3: MRR scores obtained using REE for FB15k and FB15k-237 for various dimensions of the embedding space.

Dimension	50	100	500	1000	2000
FB15k	0.815	0.816	0.816	0.816	0.819
FB15k-237	0.245	0.259	0.276	0.283	0.288

5.5.3.3 Effect of Dimension

It is known that a higher dimension for the embedding space improves the accuracy of embedding models. The dimension may also affect the accuracy of rule evaluation. Hence, we conducted the link prediction task using REE for various dimensions of the embedding space. The results are shown in Table 5.3.

The results show that a higher dimension for the embedding space improved the accuracy of rule evaluation. This may explain why the dimension is important for link prediction: a space with a higher dimension can more properly distinguish paths and learn rules. Hence, a model with a higher dimension can better predict links.

The results also show that the MRR score obtained with a relatively low dimension is close to the maximum score. REE with a low-dimensional embedding space can thus be used when fast evaluation of rules is required.

5.5.4 Experimental Results for PBF

In this section, we discuss the results for PBF in terms of accuracy of link prediction to determine whether PBF can integrate different approaches. The results of the link prediction tasks for PBF are shown in Tables 5.2 with those of other models, where PBF employed GRank or REE to select paths.

The results show that ComplEx is really efficient. We can see the importance of the attention mechanism by comparing ComplEx with TorusE. The high accuracy of ComplEx is due to its ability to utilize a variety of rules and store information separately using the attention mechanism, as discussed and experimentally shown in previous sections. PBE obtained the best results for all datasets when it incorporated ComplEx and observed feature models. Especially, PBF improves the scores for HITS@1. These results show that PBF compensates for the disadvantage of ComplEx, namely mixed and messy information, with carefully selected information.

Particularly interesting observations are the results for PBF with REE. This combination is really competitive with PBF with GRank even though it did not evaluate rules directly;

instead, the rules were evaluated based on embeddings. The results suggest that REE is sufficient for PBF.

5.6 Conclusions

In this chapter, we proposed a method for evaluating rules based on the embeddings of AKGLG called REE. Then, we proposed a framework called PBF for incorporating AKGLG and observed feature models. PBF compensates for the disadvantages of different approaches, which all utilize path information for link prediction.

We conducted experiments to evaluate the proposed methods. REE was evaluated in terms of calculation time and link prediction accuracy. The results showed that REE can reliably evaluate rules and that its calculation time is lower than that for traditional rule evaluation models for some standard datasets. These results imply that existing models that are instances of AKGLG are really utilizing rules and we can understand what embeddings learned through REE . PBF also outperformed existing models. The results comprehensively show that AKGLG and observed feature models can effectively work together.

Chapter 6

Discussion

6.1 Overview

In this chapter, we discuss the achievement, and some limitations of our models, and framework comparing with previous work.

6.2 Discussion

To achieve a method that is effective and interpretable for link prediction, we proposed several models and a framework as follows:

- GRank

  We proposed GRank solving the problems with AMIE. We showed GRank efficiently chose rules and performed the link prediction tasks competitively to embedding models.

- KGLG, AKGLG, and REE

  We proposed KGLG and AKGLG solving the expressiveness problem with TransE, the original translation-based embedding model. To propose them, we generalized TransE maintaining clearness of the mechanism of TransE that learns rules directly. Hence, we can know the mechanism of KGLG and AKGLG; they actually learn rules in a differential manner from rule-based models. We also showed many of state-of-the-art embedding models are the instances of AKGLG, which means many state-of-the-art embedding models got the interpretability. Experiments about link prediction showed TorusE, an instance of KGLG on a torus, is competitive to state-of-the-art embedding models.

  We proposed REE to interpret embeddings of AKGLG (including KGLG). REE is a rule evaluation method based on the clear mechanism of AKGLG. The basic idea of REE is that measuring the similarity between the embedding of a relation and the representation of a path obtained by composing the embeddings of the relations in the path. In the experiments, we showed rules REE could efficiently mine useful rules for link prediction and calculation time is faster than GRank.

  We can ideally consider many instances of AKGLG as many as the number of Lie Groups, i.e. infinite. Some of these may precisely learn rules and more efficient than existing instances. However, to realize a Lie group on a computer and to train a model with the Lie group is not so easy.

- PBF

  PBF is the flamework for compensating the disadvantages that each approach has to achieve a method more faster and precise link prediction than GRank and AKGLG with REE. In PBF, AKGLG is firstly used to obtain latent features of entities and relations in a knowledge graph for link prediction. Using REE, we extract useful rules for each relation faster than GRank. We construct observed feature vectors like PRA from the extracted rules with the score function of GRank. Then, a softmax regression models are constructed and trained with the observed feature vectors. Finally, embeddings and the softmax regression models are ensembled to perform link prediction. As a result, PBF outperforms state-of-the-art models on all datasets. PBF also has interpretability because PBF inherits interpretability from GRank or AKGLG.

We showed GRank and REE could efficiently perform the link prediction tasks with extracted rules, in other words, GRank and REE have higher interpretability than existing models. GRank performed the link prediction tasks with rules slightly better than REE, which means GRank is more trustable than in terms of explaining the reason for prediction, while REE can estimate rules faster.

We also showed PBF could outperform the link prediction tasks, in other words, PBF has higher reliability of prediction than existing models.

The function of GRank and AKGLG (including KGLG) are similar. Then, we come up with questions such as “what are shared characteristics by these models?” and “Are there different characteristics between these models”. We discuss these questions as follows:

- Shared Characteristics by GRank and AKGLG
  We limited the bodies of rules to paths for GRank to reduce the search space of rules like AMIE and GRank. While there are a lot of other first-order predicate logic rules such as a rule with variable relations and a rule with multiple paths as the body, GRank does not consider them. We also think rules made use of by AKGLG is limited to this type because the principle of AKGLG (same as the principle of TransE) does not capture such complex rules. This limitation can be one of the disadvantages of these models comparing with traditional rule-based models such as FOIL and ALEPH. There ideally is a knowledge graph in which such complex rules are helpful for link prediction. However, we think this limitation also contributes to the high performance of GRank and AKGLG on link prediction because the limitation efficiently cuts off useless rules, while rule-based models tend to suffer from learning useful rules [28].
- Different Characteristics between GRank and AKGLG
  While GRank and AKGLG share some characteristics, the results with REE and

ComplEx implies the existence of different characteristics between Grank and AKGLG. In Experiment 5.5.3.2, the results of the link prediction showed extracted rules by REE is not efficient as rules extracted by GRank. However, we think this is natural because AKGLG is trained to minimize the general cost function about all triples while GRank carefully checks all groundings of rules. We would say GRank and AKGLG perform link prediction like reasoning and intuition for humans, respectively.

On the other hand, ComplEx perform the link prediction tasks better than AKGLG in Experiment 4.6.2.2. Hence, AKGLG must have an advantage against GRank. We think the advantage of AKGLG is that AKGLG can employ rules with negated literal while GRank cannot do that. For example, a rule "if a person x is a spouse of y, then x is not a child of y" is generally correct and described by formal rule with a negated literal, $(x, \texttt{spouse\_of}, y) \Rightarrow (x, \texttt{children\_of}, y)$, and this is useful to predict the relation `children_of`. AKGLG can integrate those rules for link prediction.

In summary, the different characteristics between Grank and AKGLG are as follows:

- GRank has an advantage in terms of accurate learning of rules against AKGLG.
- AKGLG has an advantage in terms of the range of rules employed for link prediction against GRank.

AKGLG additionally brings us two benefits as follows:

- In previous, we did not clearly know the mechanism of embedding models. This has occurred unfavorable situations in addition to the uninterpretability of models: researchers have been developed models by just modifying the score function of triples without considering mechanism. The efficiency of the modified models has been evaluated through the link prediction tasks. However, the performance on the link prediction tasks gets easily affected by the hyper-parameters. As a result, we now know many developed versions of models are not so efficient compared with the original model or sometimes worse. For example, many variants of TransE, such as TransH and TransR, is proposed by relaxing the principle to make it easy that embeddings easily fill the principle while it is not guaranteed that translation-based models with the relaxed principle can capture useful information for link prediction. However, it has been shown that their performances are not so different from TransE [84]. AKGLG suggests rules are really important for link prediction. Hence, we think we should always keep in mind the interaction between

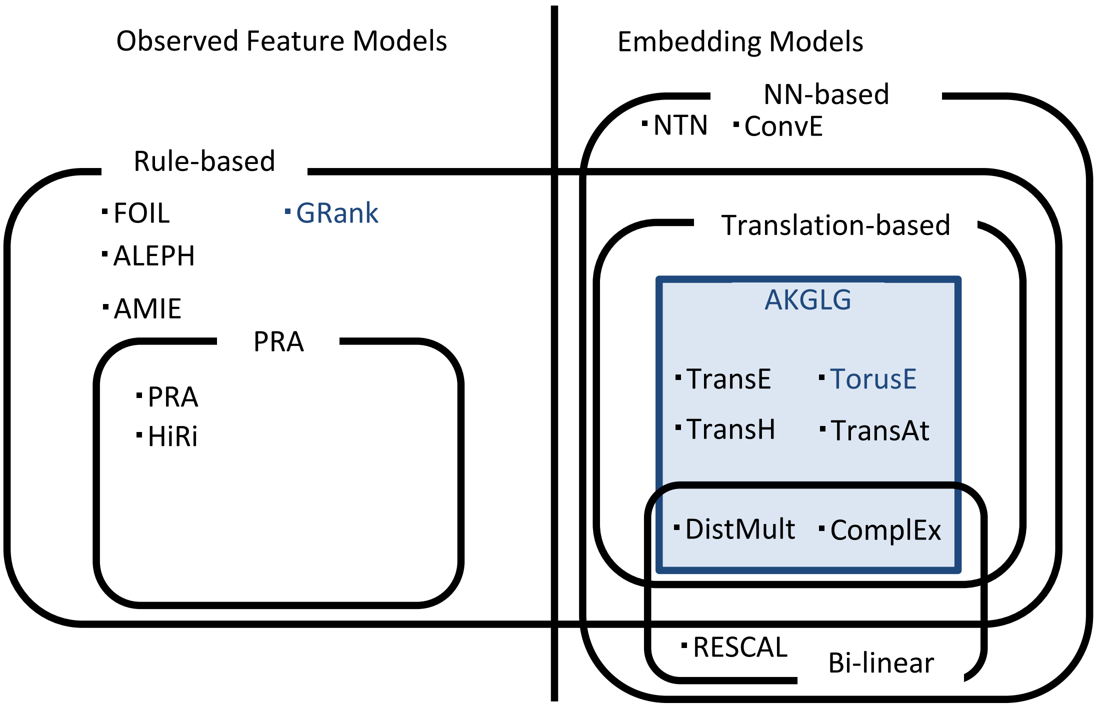


FIGURE 6.1: Inclusion relation of the approaches for link prediction after our study.

rules and embeddings for developing better models. The rule extraction task from embeddings, which we employed to evaluate REE, also may help to evaluate an embedding model from a different point of view.

- It is shown that AKGLG learns rules in a different way from traditional rule based models; AKGLG is trained in a numerical way using the stochastic gradient descent. Models of this type such as ∂ ILP [99, 100] have recently attracted much attention; we conclude AKGLG, including ComplEx, DistMult, TransE, and TorusE, are also included in those. This point of view helps us developing embedding models further; we need to consider the interaction between rules and embeddings for research of embedding models.

Our study shifted inclusion relations between each approach as shown in Figure 6.1. Now, rule-based models include translation-based models and most of the existing bi-linear models are extensions of translation-based models.

Methods to combine rules and embeddings were previously proposed [97, 98] by other researches as PBF does. Those methods basically aimed to use rules to help models to learn better embeddings. However, we think the methods are not efficient because the methods do not solve the problems each model in the methods has.

This study has enabled more accurate triple prediction. This allows us to more reliably use what our model predicts when we do not have the information we need in the knowledge graph. We also focused on interpretability, which is a major difference from recent existing researches. High interpretability of our model can assist human decision making with reasoning. This is particularly useful for knowledge graph construction because predicted triples ultimately require human confirmation. To achieve that, the technique called human in the loop has recently gathered researchers' attention. Here, "Human in the loop" is a method of making a computer efficiently learn things by intervening experts as necessary. My research is considered to play the following important roles in human in the loop of knowledge graph construction.

1. Increasing the accuracy of prediction reduces the frequency of expert intervention. In the future, we may need to predict new triples based on predicted triples like other semi-supervised machine learning methods. 2. The interpretability of our models facilitates expert judgment.

In this way, we think that our study especially enables efficient knowledge graph construction with human in the loop.

We think our work also improves the usefulness of the applications of knowledge graph embedding models to other fields giving interpretability. In the case of the recommender system for e-commerce, for example, we can add an explanation of why goods are recommended to a customer.

Chapter 7

Conclusions

7.1 Conclusions

Many knowledge graphs have recently been constructed and played important roles for various applications such as question answering, content tagging, fact-checking, and so on. Methods based on the external approach have helped us to construct those graphs, but we still need a more efficient way to do that because the information we want to describe in a knowledge graph covers many different topics. Hence, the internal approach got attention to find new knowledge hidden in an existing knowledge graph. The internal approach became possible for large enough knowledge graphs constructed by the external method. Various models have been proposed based on the internal approach, which can be classified into two groups: observed feature models and embedding models. Observed feature models generally have an advantage of interpretability and state-of-the-art observed feature models are far more effective than existing observed feature models. However, existing models lack either or both of reliability of prediction and interpretability; We think both characteristics are important for a model. In particular, the interpretability helps us to validate the prediction of a model applying extracted rules. In this dissertation, we aimed to achieve a model that has both characteristics. We developed models by solving problems with AMIE and TransE to achieve that. The brief explanation of models developed in these approaches are as follows:

- GRank
  GRank is a rule-based model with high accuracy for link prediction. We defined GPARs for a knowledge graph because GPARs were originally defined for a social network supposing strict conditions that are hard for a knowledge graph to fulfill. Then, we proposed the GRank model to evaluate GPARs modifying the problems with AMIE. First, we modified the unbalance usage of PCA. PCA gives for a triple: examples with a different head entity from the triple and examples with a different tail entity. Hence, we should define two types of confidence scores employing these examples while AMIE gives only one confidence for a GPAR. Using those confidence scores, a GPAR is estimated for head prediction or tail prediction respectively. Second, we restrict a matching function to be injective. This restriction reduces the search space by eliminating the matching of potential redundant rules. The restriction does not only prevent to overestimate meaningless rules, but also make the calculation time shorter. Third, we proposed to use a GPAR to rank entities for a query and to define the confidence score based on this idea. For a GPAR, the multiplicity of matching function is used to rank entities for a query, and we define the confidence score of the rule as a metric for rankings such as MAP. However, the idea often leads to an unpreferable situation: multiple entities have the same score and it is not clear how to treat them. We used the

distributed rankings to solve this situation and defined metrics for the distributed rankings.

GRank efficiently performed the link prediction tasks and the results showed GRank is competitive to embedding models; which means GRank has efficiently evaluated the usefulness of rules, i.e. GRank has interpretability.

- KGLG, AKGLG, and REE

  TransE, the original translation-based model, has a clearer mechanism comparing with other embedding models. However, TransE suffers from its limited expressiveness i.e. embeddings of entities and relations in a knowledge graph cannot be learned properly and, as a result, is low efficient for link prediction. We aimed to solve this limitation by generalizing TransE in two steps.

  First, we proposed the knowledge graph embedding model on a Lie group (KGLG) which is a generalized version of TransE which allows us to choose any Lie group for the embedding space. By choosing a proper Lie group, we can relax the limitation. In this dissertation, we choose a torus for the embedding space, because a torus was easy to implement and can relax the limitations in two way as follows: We do not need to use the regularization that warps embedding because a torus is a closed manifold, and a symmetric relation can be efficiently embedded on a torus while it was difficult on a real vector space. We referred to the instance of KGLG with the torus as TorusE. We also showed TorusE can be trained faster than TransE because it does not require regularization.

  Second, we proposed Attention KGLG (AKGLG) which assigns an attention vector to each relation and entity that indicate the part of the torus to store the information about the entity or the relation. This extension enables the model effectively to learn relations such as 1-to-N, N-to-1, and N-to-N relation while AKGLG inherits a clear mechanism of AKGLG while AKGLG still holds a clear mechanism for link prediction. We also showed many state-of-the-art embedding models are the instances of AKGLG, which means we can know the mechanism of many state-of-the-art embedding models.

  While we theoretically believed AKGLG learns rules and employ them for link prediction, it was not clear practically; AKGLG may work differently. Additionally, we could not interpret AKGLG. Hence, we needed a method to understand the embeddings obtained by AKGLG. We proposed a rule evaluation method based on the mechanism of AKGLG; we referred to the method as a rule evaluation method on embeddings (REE). The basic idea of REE is that measuring the similarity between the embedding of a relation and the representation of a path obtained by composing the embeddings of the relation in the path. Each relation has a representation of a Lie group and an attention vector. To obtain the representation of a path, we take the sum of the representations on a Lie group and take the geometric mean of the attention vectors regarding relations in the path. Then, the weighted similarity between the head relation and the body path of a rule to evaluate it. REE can tell us what a model learned from a knowledge graph in the form of a rule i.e., REE gives AKGLG interpretability.

We could show GRank and AKGLG (or KGLG) has interpretability. GRank and AKGLG share a characteristic that they mainly make use of rules in which the bodies are paths, and have different characteristics such as:

- GRank has an advantage in terms of accurate learning of rules against AKGLG.
- AKGLG has an advantage in terms of the range of rules employed for link prediction against GRank.

On the other hand, GRank and AKGLG have drawbacks as follows:

- GRank needs long computational time for learning rules as traditional rule-based models.
- AKGLG sometime fails to learn rules precisely because the cost function is too general to learn all rules.
- We do not have clear criteria for choosing a Lie group of AKGLG without practice. We do not even have one for choosing KGLG or AKGLG; KGLG sometime overperforms AKGLG. For example, TorusE, an instance of KGLG with a torus, overperformed ComplEx, AKGLG with a torus, on some datasets in our experiments.

Then, we proposed the framework to realize a more efficient method for link prediction combining GRank and AKGLG. We can compensate for the disadvantages of Grank and AKGLG by advantages of each other for faster and precise link prediction using this framework. The brief explanation of the framework is as follows:

- The path based framework (PBF) combines each approach for more efficient link prediction. In PBF, AKGLG is firstly used to obtain latent features of entities and relations in a knowledge graph for link prediction. Using REE, we extract useful rules for each relation. We construct observed feature vectors like PRA from the extracted rules with the score of entities in GRank. Then, a softmax regression models are trained with the observed feature vectors. Finally, embeddings and the softmax regression models are ensembled to perform link prediction.

PBF makes use of good points of each approach for link prediction. Hence, it can perform fast and accurate link prediction.

To provide details, models in PBF are cooperating explained as follows:

- The slow calculation of GRank is compensated by REE.
- The limited search space of GRank is compensated by AKGLG.
- The problem with the existence of multiple applicable rules for a query of GRank is compensated by the softmax regression models like PRA.
- The vague information in embeddings is compensated by GRank.

As a result, PBF can perform link prediction faster and precise.

7.2 Future Work

In this section, we discuss the future direction of research for knowledge graph completion and related field to this dissertation. We have ideas for two different directions as follows:

- Knowledge Graph Completion
  So far, rule-based models and embedding models have not been efficiently integrated, because we did not know the connection between them, hence, we did not know how the information in a knowledge graph is processed. In this dissertation, we reveal the connection, but we think there is still room for improvement. One of the approaches to develop a more efficient model is trying other instances of AKGLG with Lie groups as the embedding space. Each Lie group has each different characteristic. Hence, we may obtain a better embedding model that is more efficient for link prediction and extracting rules. However, to realize a Lie group on a computer and to train a model on it are really challenging; we need to effort to do that. There may also be a different approach to learn rules by embeddings.
- Explanable Artificial Intelligence
  So far, it is not easy to combine rules and a neural network, because rule learning has been done discretely while a neural network treats distributed representations. In this dissertation, we showed embedding models are rule learners in a numerical

manner. Hence, it may be possible to set an embedding model in a neural network for learning rules and we can understand what they are doing by an interpretation method such as REE.

Bibliography

[1] Ambiverse. Ambiverse, 2017. URL https://www.ambiverse.com/knowledge-graphs-encyclopaedias-for-machines/.

[2] Paul Buitelaar Anja Jentzsch Andrejs Abele, John P. McCrae and Richard Cyganiak. Linking open data cloud diagram. *URL http://lod-cloud.net/*.

[3] Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, pages 1–7, 2012.

[4] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124, 2013.

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620, 2014.

[6] Takuma Ebisu and Ryutaro Ichise. Graph pattern entity ranking model for knowledge graph completion. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 988–997, June 2019.

[7] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume abs/1711.05435, 2018.

[8] Takuma Ebisu and Ryutaro Ichise. Generalized translation-based embedding of knowledge graph. *IEEE Transactions on Knowledge and Data Engineering*, 01 2019.

[9] Takuma Ebisu and Ryutaro Ichise. Graph pattern entity ranking model for knowledge graph completion. *CoRR*, abs/1904.02856, 2019.

[10] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3): 59–79, 2010.

[11] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *HLT-NAACL*, pages 851–861, 2015.

[12] Christina Unger, André Freitas, and Philipp Cimiano. An introduction to question answering over linked data. In *Reasoning Web International Summer School*, pages 100–140. Springer, 2014.

[13] Sarthak Jain. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115, San Diego, California, June 2016. Association for Computational Linguistics.

[14] Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. Question answering over knowledge graphs: Question understanding via template decomposition. *Proc. VLDB Endow.*, 11(11):1373–1386, July 2018.

[15] K. Ashwin Kumar, Yashwardhan Singh, and Sudip Sanyal. Hybrid approach using case-based reasoning and rule-based reasoning for domain independent clinical decision support in ICU. *Expert System with Applications*, 36(1):65–71, 2009.

[16] Renata Saraiva Perkusich, João Nunes, Mirko Perkusich, Hyggo Almeida, and Clauirton Siebra. A hybrid approach using case-based reasoning and rule-based reasoning to support cancer diagnosis: A pilot study. volume 216, 08 2015.

[17] Marcos Martínez Romero, José Manuel Vázquez-Naya, Javier Pereira, Miguel Pereira Loureiro, Alejandro Pazos, and Gerardo Baños. The iosc3 system: Using ontologies and SWRL rules for intelligent supervision and care of patients with acute cardiac disorders. *Comp. Math. Methods in Medicine*, 2013: 650671:1–650671:13, 2013.

[18] Dominic Seyler, Mohamed Yahya, and Klaus Berberich. Knowledge questions from knowledge graphs. *arXiv preprint arXiv:1610.09935*, 2016.

[19] Richard Qian. Understand your world with bing. *Official Bing Blog, March*, 2013.

[20] François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716, 2008.

[21] Yang Liu, Qingguo Zeng, Huanrui Yang, and Adrian Carrio. Stock price movement prediction from financial news with deep learning and knowledge graph embedding. In Kenichi Yoshida and Maria Lee, editors, *Knowledge Management and Acquisition for Intelligent Systems*, pages 102–113, Cham, 2018. Springer International Publishing.

[22] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 285–294, 2019.

[23] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8, 2017.

[24] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, 2007.

[25] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, pages 722–735, 2007.

[26] 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.

[27] Amit Singhal. Introducing the knowledge graph: things, not strings. *URL https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html*, 2012.

[28] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, Jan 2016.

[29] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[30] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 09 2019.

[31] Tim Berners-Lee. Linked data-design issues (2006). *URL http://www. w3. org/DesignIssues/LinkedData. html*, 10:11, 2011.

[32] Natthawut Kertkeidkachorn. Knowledge graph population from natural language text, 2017.

[33] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, November 1995. ISSN 0001-0782.

[34] George A. Miller. Wordnet: A lexical database for English. *Commun. ACM*, 38 (11):39–41, November 1995.

[35] Jorge Morato, Miguel Angel Marzal, Juan Lloréns, and José Moreiro. Wordnet applications. In *Proceedings of GWC*, pages 20–23, 2004.

[36] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[37] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[38] GeoNames. Geonames, 2017. URL http://www.geonames.org/. [Online; accessed 5-April-2017].

[39] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, 2010.

[40] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, July 2015.

[41] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[42] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. ACL, 2011.

[43] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3): 239–266, Aug 1990.

[44] Niels Landwehr, Kristian Kersting, and Luc De Raedt. Integrating naïve bayes and FOIL. *J. Mach. Learn. Res.*, 8:481–507, 2007.

[45] Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi. Fast learning of relational kernels. *Machine Learning*, 78(3):305–342, 2010.

[46] Stephen Muggleton. Inverse entailment and progol, 1995.

[47] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference*, pages 413–422, 2013.

[48] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015.

[49] Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. Association rules with graph patterns. *VLDB J.*, 8(12):1502–1513, 2015.

[50] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.

[51] Ni Lao, Tom M. Mitchell, and William W. Cohen. Random walk inference and learning in A large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, 2011.

[52] Qiao Liu, Liuyi Jiang, Minghao Han, Yao Liu, and Zhiguang Qin. Hierarchical random walk inference in knowledge graphs. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–454, 2016.

[53] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[54] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

[55] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[56] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[57] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[58] 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.

[59] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, July 2015.

[60] Wei Qian, Cong Fu, Yu Zhu, Deng Cai, and Xiaofei He. Translating embeddings for knowledge graph completion with relation attention mechanism. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4286–4292, 7 2018.

[61] Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, December 2014.

[62] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 985–991, 2016.

[63] 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 Machine Learning*, pages 809–816, 2011.

[64] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2014.

[65] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080, 2016.

[66] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Effective blending of two and three-way interactions for modeling multi-relational data. In *Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECMLPKDD'14, pages 434–449, 2014.

[67] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961, 2016.

[68] Tony A. Plate. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CSLI Publications, Stanford, CA, USA, 2003.

[69] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[70] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610, 2014.

[71] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Mach. Learn.*, 94(2):233–259, February 2014.

[72] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, pages 601–610, 2014.

[73] Quan Liu, Hui Jiang, Zhen-Hua Ling, Si Wei, and Yu Hu. Probabilistic reasoning via deep learning: Neural association models. *CoRR*, abs/1603.07704, 2016.

[74] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[75] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[76] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[77] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103, 2017.

[78] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume abs/1707.01476, 2018.

[79] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 127–135, 2012.

[80] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 301–306, 2011.

[81] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 623–632, 2015.

[82] Solomon Kullback. *Information Theory and Statistics.* 1959.

[83] Han Xiao, Minlie Huang, and Xiaoyan Zhu. TransG : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2316–2325, Berlin, Germany, August 2016.

[84] Dat Quoc Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *CoRR*, abs/1703.08098, 2017.

[85] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*, 2015.

[86] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961, 2016.

[87] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, 2015.

[88] J.F. Adams. *Lectures on Lie Groups.* Midway reprints. University of Chicago Press, 1969.

[89] Nicolas Bourbaki. *Lie Groups and Lie Algebras*. Springer Publishing Company, Incorporated, 2008.

[90] B. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Graduate Texts in Mathematics. Springer, 2003.

[91] P.M. Cohn. *Lie Groups*. Cambridge tracts in mathematics and mathematical physics. Cambridge University Press, 1957.

[92] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Trans. Automat. Contr.*, 58(9):2217–2229, 2013.

[93] Hongyi Zhang, Sashank J. Reddi, and Suvrit Sra. Riemannian SVRG: fast stochastic optimization on riemannian manifolds. In *Advances in Neural Information Processing Systems*, pages 4592–4600, 2016.

[94] Han Xiao, Minlie Huang, and Xiaoyan Zhu. TransG : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[95] Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2863–2872, 10–15 Jul 2018.

[96] Théo Trouillon, Christopher R. Dance, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *CoRR*, abs/1702.06879, 2017.

[97] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Joint Conference on Empirical Methods in Natural Language Processing*, pages 192–202, 2016.

[98] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4816–4823, 2018.

[99] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Int. Res.*, 61(1):1–64, January 2018.

[100] William W. Cohen. Tensorlog: A differentiable deductive database. *CoRR*, abs/1605.06523, 2016.