Type-Embodied Representation Learning for Knowledge Graph Embedding

by

Md Mostafizur Rahman

Dissertation

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

September 2020

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

Advisory Committee

1. Prof. Atsuhiro Takasu (Supervisor)	National Institute of Informatics SOKENDAI
2. Prof. Hideaki Takeda	National Institute of Informatics SOKENDAI
3. Assoc.Prof. Kenro Aihara	National Institute of Informatics SOKENDAI
4. Assoc.Prof. Ryutaro Ichise	National Institute of Informatics SOKENDAI
5. Prof. Tomonari Masada	Rikkyo University

“If you want to shine like a sun, first burn like a sun."

— A. P. J Abdul Kalam

Acknowledgments

There are many people without whom I would never have achieved this goal. All of them contributed in their own way and proved to be essential to my academic and professional growth, to my personal development, and to my mental health. Here, I want to thank them all.

My sincere and deep gratitude goes first and foremost to my supervisor, Professor Atsuhiro Takasu. He provided me with inestimable advice and guidance, he also showed great patience, and always understood the issues that each Ph.D. candidate had including the personal ones. The path to obtaining a Ph.D. is long and difficult, I think that working in a caring environment helped me a lot in dealing with all the difficulties I faced.

I am grateful to my advisory committee, Prof. Kenro Aihara, Prof. Hideaki Takeda, Prof. Akiko Aizawa, Prof. Ryutaro Ichise, and Prof. Tomonari Masada. I highly appreciate their insightful and helpful comments.

I would like to thank all the members and internship students in our laboratory, specially our lab secretary Akiko Takenaka, she is a nice and cheerful person. I am also thankful to Makoto Takenaka for making my life easy and enjoyable here at Japan. Special thanks to Dr. Gianluca Dermatini, University of Queensland, Australia for his invaluable advices in my research.

Finally, I owe eternal gratitude to my only son Ayan, my parents, Md Mokbular Rahman and Rahima Rahman, my sister Litu and her husband Mahbub, and to my longtime friend and wife Munne for supporting me unconditionally. They inspired me and helped me becoming what I am by totally ignoring my odds.

Tokyo, June 30, 2020 Md Mostafizur Rahman

Abstract

A Knowledge Graph (KG) is a knowledge base containing facts about real world entities represented as a graph. We have witnessed rapid growth in knowledge graph construction and application in recent years. A large number of KGs, such as Google Knowledge Vault, Wikidata, DBpedia and Freebase have been served several fields of real world applications from semantic parsing and named entity disambiguation to information extraction and question answering. In this thesis we develop new methods for effectively using entity types in KG embedding and ranking entity types for better understand the entities.

We start this thesis by presenting novel techniques for using entity types in knowledge graph embedding for knowledge graph completion. Popular KG embedding methods focus on the structured information of triples and maximize the likelihood of them. However, they completely ignore the semantic information contained in most knowledge graphs and the prior knowledge indicated by the semantic information. To overcome this shortcoming of the embedding methods, we propose an approach that integrates the structured information and entity types in relational context which describe the categories of entities. With the type-based prior distributions, our approach generates multiple embedding representations of each entity in different relational contexts and compute the plausibility for triples using the sate-of-the-art methods. Extensive experiments show that entity types exhibit useful semantic information to describe the entities.

Entity type information plays an important role in knowledge graphs (KGs). In a KG, an entity usually holds multiple type properties. In our second contribution, we address the entity type ranking problem by means of knowledge graph embedding models. We try to show how entity type ranking can exhibit the corresponding

entity's characteristics. In our work, we show that entity type ranking can be seen as a special case of the KG completion problem. Our proposed approach outperforms the state-of-the-art type ranking models while, at the same time, being more efficient and scalable.

Finally, we discuss the problem of assigning relevance scores for triples from type-like relations. We can employ those scores as a fundamental ingredient for ranking results in entity search. We propose a joint semantic relevance learning approach using the text and structured data. The results of the triple scoring work indicate important directions for the future work.

Contents

List of Figures

List of Tables

1 Introduction

Knowledge is the core power in the age of data, incorporating the knowledge in graphical representation has been around us for decades now. It is an interesting point to make about how "important" and "strategic" data is and how to extract "knowledge" from raw data and store it. To serve that purpose knowledge graph concept comes to action, which is a way of storing entity-centric data modeled as a graph.

The Knowledge Graph enables us to search anything related to our interest and context. But things were not that smooth before the KG era. For decades search has essentially been about matching keywords to queries that means matching the keywords and searching results were not much satisfactory as nowadays. Surprisingly, queries in search engines have increased awfully only because of user satisfaction. Google receives over 63,000 searches per second and 5.6 billion searches per day [1].

A KG is a multi-relational graph composed of entities and relations between the entities. Modern KGs have been successfully applied to many real-world applications, from semantic parsing and named entity disambiguation to information extraction and

[1] https://seotribunal.com/blog/google-stats-and-facts/

question answering and became an appealing topic in machine learning domain. We discuss the motivation, challenges and contributions of the thesis in this chapter.

1.1 Motivation

This dissertation focuses on the influences of entity types in knowledge graph applications. Representation learning models for knowledge graph embeddings typically encode entities or concepts of a knowledge graph in a continuous and low-dimensional vector space, where the relational inferences of entities are modeled as semantically meaningful vector. Hence, these models provide efficient and versatile methods to incorporate the symbolic knowledge of multi-relational data into machine learning.

Most of the currently proposed KG embedding models perform the embedding task solely on the basis of observed facts. Given a KG, such a model first represents entities and relations in a continuous vector space, and defines a scoring function on each fact to measure its plausibility. Those representation learning based models concentrate on structured information in triples and ignore the rich semantic information of entities and relations, which is contained in the KGs. Semantic information includes entity types, entity's text descriptions/summary, and other textual information. There is no doubt that recently proposed models have significantly improved the embedding representations and increased the prediction accuracy, there is still room for improvement by exploiting semantic information such as entity types. One of the main drawbacks of knowledge graph completion is the polysemy of entities or relations, i.e., each entity or relation may have different semantics in different triples based on the relational context. For example, in the triple (Tom Hanks, starredin, Terminal), the type of entity Tom Hanks can be considered as actor here, while in (Tom Hanks, director of, Larry Crowne), Hanks is a director. If we keep the same vector representation for the entity Tom Hanks for these two relations, it may increase confusion. The current popular models miss the rich sematic type information.

In the todays' world search queries are entity centric data oriented. One of the important piece of information associated with the entity is the type/category information. However, an entity is usually not associated to a single generic type but rather to a set of more specific types, which may be relevant or not given the

document context. Ranking between those types may help Search Engine Result Pages (SERPs) to display meaningful search results of user initiated queries. So efficient entity type ranking method is required to tackle such problems.

In this dissertation, we aim at investigating multi-relational representation learning methods leveraging entity type for knowledge graph completion, which capture complex properties of the real-world facts. We also address the problem of entity type ranking and propose some techniques to tackle that task.

1.2 Challenges

We address several key challenges in this thesis. First, we propose the study of learning embeddings for knowledge graphs. Models should characterize the heterogenous structures of knowledge graphs in the embedding space, while capturing the precise correspondences of entities and relations across graphs. In addition, this learning process is often subject to insufficient supervision, since the alignment information to learn the correspondence is provided to only a limited extent.

1.2.1 Heterogenous Structures of Knowledge Graphs

In knowledge graph, we have a pool of diverse and heterogeneous facts representing our knowledge about the instances [7]. Some facts link instances together (Trump, presidentOf, USA) and (Trump, starredIn, Home Alone 2), on the other hand some describe attributes of instances (Home Alone 2 was released on 1992). The way of storing the facts in a KG may vary widely. Some knowledge graphs often lead to heterogeneity in relation facts, as well as that in vocabularies of entities and relations. Not only that, KGs contain language-specific facts as well. As we all know that knowledge graphs are typically extracted from independently maintained corpora [8], which no doubt leads to all the abovementioned heterogeneity. Different KGs exploit different ontologies so it's very difficult to align them in general. For the above mentioned reasons, proposing ideal embedding models considering the challenges might be difficult than the theoretical aspects.

1.2.2 Various Relational Properties

In the relations in KGs, we generally notice the transitivity and symmetry properties. Sometimes relations are arranged hierarchically. Such relational facts lead to non-linearity of the embedding space in the KG that is hard to manage considering the regular relational facts. Most of the KGs are ontology based. We show a statistic about how different kind of relational properties are distributed in knowledge graphs. For example, Yago3 covers more than 92% of the relations are transitive or symmetric relations, and more than 95% of the relations are hierarchical [9]. Freebase contains more than 20% of transitive or symmetric relations [10]; ConceptNet [11] contains 70% of transitive or symmetric relations, and at least 26% of hierarchical relations. In knowledge graph embeddings, these problems always effect the performances.

1.2.3 How to Manage Too Much Information in a Knowledge Graph?

Entities are attached with various kind of data in KG. So it's often very challenging to summarize the characteristics of entities. It means representing an entity with the most useful or meaningful information. In a KG entities are associated with huge information, which is difficult to process and present them in an organized way. Among all the information that can be enclosed to entities, type information is one of the vital pieces as it defines what the entities are; for example, in Freebase the entity "Tom Hanks" is defined to be a person, a director, a writer and several other things, including academy award winner, producer, and thing. Nevertheless, as we can see, entities are often associated with several types (Barack Obama has 119 types in DBpedia), and selecting the one type or showing the users about the most relevant types can be challenging.

1.2.4 Dependencies on Abelian Group

The current knowledge graph embedding models are some form of Abelian group [12]. It is interesting to know that, to accommodate all possible KG datasets, there are five requirements for the relation embedding: closure, identity, inverse, associativity, and non-commutativity. The first four coincide with the algebraic definition of groups in

mathematics, Embedding all relations into a group manifold (and designing mapping operations as group actions) would automatically satisfy all requirements; in addition, the last requirement, non-commutativity, further suggests implementing non-Abelian groups for the most general KGE tasks. It is not sure what kind of non-Abelian group would be helpful here. It is a open challenge for the community and there's a lot to contribute in near future.

1.3 Thesis Contributions

In this dissertation, we tackled several tasks connected to knowledge graph embeddings, entity type ranking, joint text and representation models for type-like triples scoring and action mining for actionable knowledge graph generation. We proposed effective methods to deal with them. In the following we give a succinct overview of the tasks we studied, and of the contributions we made. We also mention peer reviewed articles published along our research work.

1.3.1 Knowledge Graph Embedding

Knowledge graph (KG) embedding is to embed components of a KG including entities and relations into continuous vector spaces. As KG has quickly gained massive attention., currently, several artificial intelligence and machine learning applications are using KG.

We target the polysemy characteristics of entities in different relational context. It is intuitional that one entity can play many roles based on relations [13, 14]. We measure the semantic similarity of entities and relations based on entity types on relational context. With this type-based semantic similarity, we integrate type information into entity representations. We model each entity as multiple semantic vectors with type information to represent entities more accurately and meaningfully. Our model projects the multiple semantics of entities separately and utilizes the semantic similarity to distinguish entity semantics in different relations. We apply our idea with the popular KG embedding models and find very impressive performance in terms of accuracy and semantic expression.

1.3.2 Entity Type Ranking

We now give more focus on entity type and learn how ranking of relevant entity types associated with the entity can contribute search queries. Nowadays searches are totally entity centric. In a KG, entities are bind with huge information. So displaying that information in a organized way is a big challenge. Moreover, it's often the case that one entity is associated with too many types in a KG. Entity types typically summarize the entity and provide a brief introduction of that entity.

In our work [15], we show how entity type ranking problem can be tackled by means of knowledge graph embedding. We define a novel way to use knowledge graph embedding model for ranking entity types. To do this we propose a technique to construct knowledge graph to address type ranking problem.

1.3.3 Type-Like Triple Scoring

The Type-Like Triple Scoring task involves the calculation of the relevance scores between entities and correspond types. In our previous task we converted the crowdsourced dataset into a graph and rank the entity types based on relevance. In type-like triple scoring task we worked on the dataset provided by WSDM Cup 2017. Several teams participated in this task and the BOKCHOY team achieved the most better performance. We improvise the best model and achieve good accuracy than the award winning team. We use ensemble learning to combine multiple base scorers and observe that our model achieves better predictive performance than each single model. Here, we propose a model which exploits the entity descriptions available in Wikipedia and employs KG embeddings model to obtain the vectors (head/tail). In our model we used the continuous bag of words model [16] to convert the description into vectors, which is surely the most basic one. There is a good chance to enhance the model performance from it's present form. We briefly discuss them in the future plans.

1.3.4 Other Contributions

In addition to the core contributions described above, we also contributed to other scientific researches.

We participated in the Action Mining (AM) task organized by NTCIR-13 [17]. In

recent years, popular search engines are utilizing the power of Knowledge Graph(KG) to provide specific answers to queries and questions in a direct way. It is expected that search engine result pages (SERPs) will provide facts about the quires satisfying semantic meaning, which encouraging researchers to constructing more powerful Knowledge Graph. One of the major challenges is disambiguating and recognizing entities and their actions stored in KG in a context. To achieve and advance the technologies related to actionable knowledge graph presentation, Action Mining (AM) is an essential step and relatively new research direction to nurture research on generating such KG that is optimized for facilitating entity's actions e.g. for entity "Donald J. Trump" most potential actions could be "won the US Presidential Election" or "targeting US journalists". We employ a probabilistic model to address the AM problem [18].

We prepared a dataset which includes type information of a subset of freebase triples. Our data is open for use.

1.4 Thesis Outline

We conclude this first chapter by outlining the structure of this dissertation. This thesis is structured in seven chapters. The outline of the following chapters:

- **Chapter 2**

  In the following chapter we introduce basic knowledge to help readers understanding the rest of the dissertation. We first explain knowledge graph with an intuitive example and we also discuss the components of knowledge graph. We also summarize current automatic methods for building knowledge graphs and existing datasets.

- **Chapter 3**

  Here we summarize the related works of this dissertation. We start by introducing different categories of knowledge graph embedding models. Then we provide an overview of previously proposed knowledge graph embedding approaches. Our work seeks to extend new approaches to support KG embeddings (please refer to Chapter 4), and to capture complex relation properties. Then we describe entity

type ranking system: TRank. Lastly, we summarize the related works of triple scoring task. We also discuss the contributions of the papers presented in the WSDM Cup 2017.

- **Chapter 4**

  This chapter proposes, an approach for Knowledge graph embedding, which combines structured information and type information. Proposed approach makes full use of type information and accurately captures semantic features of entities. Extensive experiments show that our model achieves significant improvements against the baselines

- **Chapter 5**

  In this chapter we describe several approaches for entity Type Ranking problem. We show how we managed to contribute to the entity type ranking problem by introducing the KG embedding models. We compare the results with the state-of-the-art entity type ranking models.

- **Chapter 6**

  In this chapter we describe our model for the Triple Scoring Task. We demonstrate the significant potential of using joint graph and text embedding techniques to learn the relevance scores from type-like relations.

- **Chapter 7**

  This chapter summarizes the contributions of the thesis and outlines directions for future work.

2

Background Knowledge

This chapter provides readers background information needed to understand the rest of this dissertation. After this chapter, readers will have an idea of what a knowledge graph is, how different knowledge graphs have been formed, knowledge graph embedding techniques, entity type ranking problems and how those models help us in real life applications. In addition, we will briefly cover aspects related to the evaluation of the methods we propose in subsequent chapters.

In what follows, we focus on simplicity and concision: we try to provide all and only the information needed to follow the content of the thesis, sometimes by skipping technical aspects which are not relevant to this thesis, and that would just burden the reader. We try, however, to provide pointers to further material that interested readers can consult to get more information on the topics we mention.

2.1 Knowledge Graph Concept

Recently, construction of knowledge graph has received great attention from both academia and industry. Early work usually involved much human effort like ontologies in different languages. Expert-edited knowledge graph is of high quality, but the construction and maintenance are very expensive and labor-intensive, making it not scalable. The automatic construction of knowledge graph helps solve those problems and thus has become increasingly popular. We will discuss about automatic KG construction in later sections. Here, we discuss the general intuition behind storing data in graph format and common terminologies.

2.1.1 Structure of Knowledge Graph

Knowledge graph can be divided into two types: monolingual and multilingual. Multilingual concept become popular over mono lingual. In current knowledge graphs, such as Wikipedia, Dbpedia, and Concept- Net (Speer and Havasi, 2013), vast amounts of multilingual knowledge are being created across the multiple language-specific versions of the knowledge base. Such multilingual knowledge, including inter-lingual links, and triple-wise alignment, is very useful in aligning and synchronizing different language-specific versions of a knowledge base that evolve independently, as needed to further improve applications built on multilingual knowledge. However, such cross-lingual knowledge is far from complete, while extending it is challenging due to the fact that it is almost not possible for existing corpus to directly provide such knowledge of expertise. Existing approaches involve either extensive human involvement or require training comprehensive models on information that is external to knowledge graphs.

Figure 2.1 shows a small example knowledge graph. Knowledge graphs are mainly composed of three types of object: entities, relation properties, and entity types.

Entities

Entities are the concepts which are described by the knowledge graph. In Figure 2.1 entities are represented by images: the TV show titled "Big Bang Theory", the charecters Leonard and Sheldon, and the television series name Big Bang Theory are

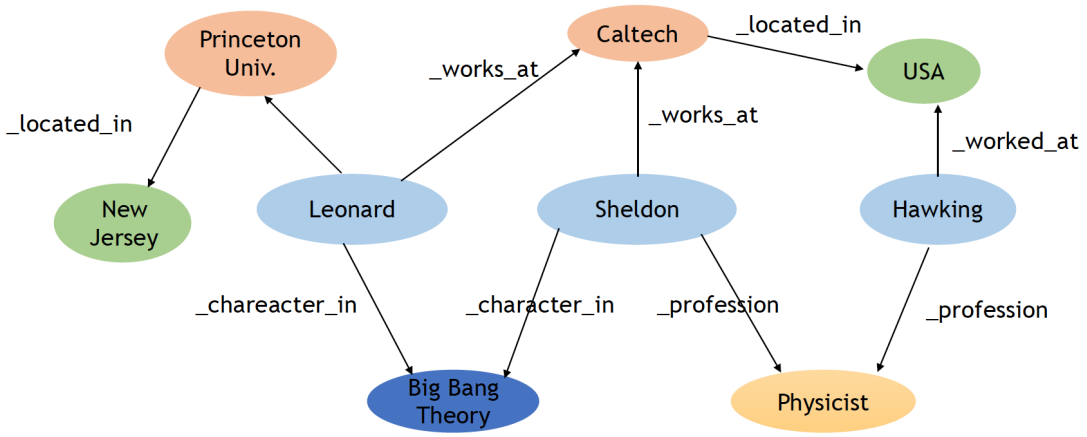

Figure 2.1: Knowledge Graph Example

the entities composing our small knowledge graph. In the rest of this dissertation we use the words "entity", "concept", and "resource" interchangeably even if in some contexts they denote different concepts.

Properties

Properties are used in two ways: either they are used to give information about entities by using literals such as the string "Big Bang Theory", or they represent relations between pairs of entities. Note that we use different fonts to distinguish string literals from entities. In Figure 2.1 properties are represented by labeled arrows. For example, the property "_profession"gives information about the entity Hawkingby linking it to an entity Physicist. The direction of such arrows is essential since it defines which entity does what: TV shows star actors but not vice versa. In the rest of this dissertation, if not explicitly stated, we use the words "property" and "predicate" interchangeably.

Entity Types

Entity Types are used to specify what an entity is and which properties it usually has. Entities can be instances of one or more types, for example, in Figure 2.1 "Howking" is both a "Person" and an "Physicist". In many knowledge graphs types are ordered by specificity, for instance, in our example the property "subtypeOf" is used to state

that "Physicist" is a subtype of "Person", thus every instance of "Physicist" is also an instance of "Person", but not vice versa. In this case we say that "Physicist" is more specific than "Person". In the rest of this dissertation, if not explicitly stated, we use the terms "entity type", "type", and "class" interchangeably.

2.1.2 Knowledge Graph

A Knowledge Graph is a model of a knowledge which represents a collection of interlinked descriptions of entities with the help of intelligent machine learning algorithms. Some authors use the term "knowledge graph" to denote a semantic network, some consider knowledge graphs as ontologies, and some others use the term to denote knowledge bases. Finally, several people associate knowledge graphs with Big Data and state, basically, that a knowledge graph is a large ontology (or knowledge base), despite the notion of "large" being quite subjective.

The very formal definition of knowledge graph we used in this study is based on the RDF data model. We can easily describe KG by Eq. (2.1), where E denotes the set of all entities (individuals), T the set of all entity types (classes), P the set of all the properties used in the knowledge graph, and L the set of all literals (data values).

$$KG = \{(s, p, o) \mid s \in E \cup T \cup P, p \in P, o \in E \cup L \cup T \cup P\} \tag{2.1}$$

The equation basically states that a knowledge graph is a semantic network defined as a set of triples (s ,p ,o) (in later chapters while we denote triples as facts, there we use subject as the head entity, h; predicate as relation, r and object as the tail entity, t) specifying that a node s (either an entity or a type) is connected to another node o by the property p . We call the components of each triple as in RDF: subject, predicate, and object. Notice that, since knowledge is modeled by using a graph structure, only binary properties connecting pairs of entities are allowed, however, n -ary relations can still be encoded by using some of the techniques proposed by the W3C consortium. The simple definition of a knowledge graph we gave is accurate enough to express all the concepts we present in this dissertation; more complete and detailed definitions can be given by using a variant of the ontology model presented in [19] but doing so does not provide any added value to the reader of this thesis.

Unless otherwise stated, we assume that a knowledge graph contains all information

users need to tackle their tasks. In this sense, a knowledge graph can include (or make use of) several, possibly interlinked ontologies. With this statement we want to highlight the fact that KGs can reuse vocabulary defined by existing ontologies (e.g., by using both GeoNames properties and DBpedia properties), as well as entities which are part of other ontologies (e.g., by using links to both GeoNames and DBpedia entities). Moreover, we say that the schema of a knowledge graph is composed by its types, the subsumption relations among them, and by the formal declaration of the properties used in the knowledge graph, including their subsumption relations and the declarations of their domains and ranges. Knowledge graph schemata are usually defined by using RDFS and OWL. We will cover all these aspects in the following sections. Knowledge Graphs are often mentioned together with "Linked Data"; the two concepts are related but denote very different things since a knowledge graph is a knowledge base while Linked Data is a method of publishing structured data. A knowledge graph can be Linked Data if it is published according to the three "extremely simple" rules described by Tim Berners-Lee. Nevertheless, we consider the Linked Open Data Cloud, which we describe in Section 2.2, to be a Knowledge Graph.

2.2 Creating Knowledge Graphs

The goal of this section is to introduce readers to the knowledge graphs that we use throughout this thesis and to give them an overview of the core ideas currently used to create knowledge graphs. In the following we first briefly describe DBpedia, YAGO, Freebase and Wikidata, and then introduce Linked Open Data, and other methods for creating knowledge graphs worth mentioning. Interested readers can have more information on methods for creating and refining knowledge graphs by reading the survey made by [20]. Finally, readers who are willing to know more about the differences between the knowledge graphs we present can consult the article by Färber et al., devoted to this topic.

2.2.1 Wikidata

Wikidata[1], operated by the Wikimedia Foundation is a community-created knowledge base to manage factual information of Wikipedia and its sister projects operated by the Wikimedia Foundation [21]. In other words, Wikidata's goal is to be the central data management platform of Wikipedia. Currently, Wikidata contains more than 86 million items and has more than 26,957 active users[2]. An RDF export of Wikidata was introduced in 2014 and recently a few SPARQL endpoints were made available as external contributions [22]. Wikidata is a collection of entity pages. There are two types of entity pages: items and properties. Every item page contains labels, short description, aliases, statements and site links. Each statement consists of a claim and one or more optional references. Each claim consists of a property-value pair, and optional qualifiers. Values are also divided into three types: no value, unknown value and custom value. The no value marker means that there is certainly no value for the property, the unknown value marker means that the property has some value, but it is unknown to us and the "custom value " which provides a known value for the property.

2.2.2 DBpedia

In recent years Wikipedia[3] has became a important source of information to build knowledge graphs. The DBpedia[4] project extracts the structured information that is included in Wikipedia articles by using wiki markup. The DBpedia project was officially launched in 2007 and on those days the Linked Open Data Cloud was only a tiny collection of bubbles coalescing around DBpedia. Wikipedia's "infoboxes", are very rich and the most valuable source of information for DBpedia, which are tables summarizing the most important properties of the entity described by a Wikipedia page [23]. This extraction process generates one of the most popular knowledge graphs publicly available that, at the time of writing, contains more than 17M entities. DBpedia also features manually curated ontologies defining, in particular, a hierarchy of types and domain and range constraints for many properties. Being derived from Wikipedia, DBpedia features encyclopedic knowledge mostly focused on people,

[1] https://wikidata.org
[2] https://www.wikidata.org/wiki/Wikidata:Statistics
[3] http://wikipedia.org
[4] http://dbpedia.org

locations, organizations, and creative works such as books, pieces of art, movies, and music.

2.2.3 YAGO

Surprisingly YAGO also launched in the year of 2007 which shares with DBpedia [24] some core ideas but aims at integrating Wikipedia and WordNet[5], which is a very popular lexicon for the English language [25].

YAGO2 enhances the knowledge graph by adding spatial and temporal information to its data. Spatial information is attached to entities of type Event, Group (or Organization), and Artifact by means of special properties. Such properties connect the entity to geographical entities extracted either fromWikipedia or GeoNames, a large knowledge graph about locations containing more than 7M entries. Temporal information is mostly extracted from Wikipedia infoboxes and is attached to people, groups, artifacts, and events. In all cases, temporal information denotes the time of existence of the entity; for example, people exist from the point in time in which they were born to the one in which they died. YAGO2 puts particular emphasis also on the time dimensions of facts by using special algorithms to detect when facts, encoded by using properties, are valid. The latest version of YAGO, YAGO3, uses Wikidata (described later in this section) in order to merge information coming from different versions of Wikipedia redacted in different languages. Interestingly enough, in YAGO3 the spatial and time dimensions of entities and facts introduced in YAGO2 were not taken into consideration.

YAGO3 is a huge semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Currently, YAGO3 has knowledge of more than 10 million entities (like persons, organizations, cities, etc.) and contains more than 120 million facts about these entities.

According to Hoffart et al., who also built YAGO2, one of the main differences between YAGO and DBpedia is the fact that the two knowledge graphs have different type hierarchies: DBpedia features about 300 types while YAGO features more than 300,000 entity types.

This is due to the fact that DBpedia's type hierarchy was created manually, while

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

YAGO's was automatically derived starting from Wikipedia categories and WordNet. The main consequence of this fact is that YAGO has very specific entity types which can have very few instances and, therefore, are not always interesting for the users of the knowledge graph. We extensively discuss this issue in Chapter 4. Conversely, YAGO relies on carefully hand made patterns to extract information fromWikipedia infoboxes and tries to merge values coming from similar attributes of infoboxes (such as "birthdate" and "dateofbirth"), DBpedia does not resulting in a remarkable difference in the number of properties used by the two knowledge graphs: while YAGO features about 100 manually curated properties, DBpedia features more than 1,000 properties, which are sometimes duplicated or too specific (e.g., dbo:aircraftHelicopterAttack). Although DBpedia and YAGO can be considered to be competitors, they actually complement each other: there are numerous owl:sameAs links between the two datasets and DBpedia entities also feature YAGO types.

2.2.4 Freebase

Freebase was a large collaborative knowledge base launched in 2007 as well. Google took it over in 2010 [26]. It was used as the open core of the Google Knowledge Graph project, and has been attracted by many use cases outside the Google. Due to the success of Wikidata, Google had decided to close Freebase in 2014 and help with the migration of the content to Wikidata [21]. Freebase is built on the notions of objects, facts, types, and properties. Each Freebase object has a stable identifier called a "mid" (for Machine ID), one or more types, and uses properties from these types in order to provide facts. For example, the Freebase object for Barack Obama has the mid /m/02mjmr and the type /government/us_president (among others)) that allows the entity to have a fact with the property /government/us_president/presidency_number and the literal integer "44" as the value. Freebase uses Compound Value Types to represent n-ary relations with $n > 2$, e.g., values like geographic coordinates, political positions held with a start and an end date, or actors playing a character in a movie. Compound Value Types values are just objects, i.e., they have a mid and can have types [27]. Most non- Compound Value Types objects are called topics in order to discern them from Compound Value Types .

Google has stopped all the Freebase services from 2016 and its data was "donated"

to Wikipedia, though only 9.5% of its entities have actually been included in Wikidata, partly because of the notability criteria mentioned previously. The last dump of Freebase is still available for download[6].

2.2.5 Google's Knowledge Graph and Knowledge Vault

Nowadays in Google Search you can find info boxes with information about people, places and things. Infoboxes are designed to help end users quickly understand more about a particular subject by surfacing relevant facts and to make it easier to explore a topic in more depth. Information within info boxes comes from a Knowledge Graph, which is like a giant virtual encyclopedia of facts. In 2012 the Google's Knowledge Graph was invented and brought to the public, which was also when the term knowledge graph as such was coined. Google itself is rather secretive about how their Knowledge Graph is constructed; there are only a few external sources that discuss some of the mechanisms of information flow into the Knowledge Graph based on experience . From those, it can be assumed that major semi-structured web sources, such as Wikipedia, contribute to the knowledge graph, as well as structured markup (like schema.org Microdata on web pages and contents from Google's online social network Google+. According to the Google's Knowledge Graph contains 18 billion statements about 570 million entities, with a schema of 1,500 entity types and 35,000 relation types [28].

The Knowledge Vault is another project by Google. It extracts knowledge from different sources, such as text documents, HTML tables, and structured annotations on the Web with Microdata or MicroFormats. Extracted facts are combined using both the extractor's confidence values, as well as prior probabilities for the statements, which are computed using the Freebase knowledge graph (see above). From those components, a confidence value for each fact is computed, and only the confident facts are taken into Knowledge Vault. According to , the Knowledge Vault contains roughly 45 million entities and 271 million fact statements, using 1,100 entity types and 4,500 relation types [29, 28].

[6] https://developers.google.com/freebase/

2.2.6 Yahoo!'s Knowledge Graph

Yahoo! has their internal knowledge graph and launched in 2013 to support entity-oriented services [30, 31]. Which has proven impact on their search results [32]. It is an organized collection of normalized information about entities modeled as a graph. Information is extracted from the Web and various public and commercial databases and integrated with a central knowledge base where it is normalized, reconciled and refined automatically. The knowledge graph builds on both public data (e.g., Wikipedia and Freebase), as well as closed commercial sources for various domains. It uses wrappers for different sources and monitors evolving sources, such as Wikipedia, for constant updates. Yahoo's knowledge graph contains roughly 3.5 million entities and 1.4 billion relations. Its schema, which is aligned with schema.org, comprises 250 types of entities and 800 types of relations [29].

2.3 Resource Description Framework (RDF)

The Resource Description Framework (RDF)[7] is an infrastructure that enables the encoding, exchange and reuse of structured metadata. The Resource Description Framework (RDF), developed and introduced by World Wide Web Consortium (W3C). To represent a knowledge graph we usually adopt the RDF data model. RDF is an application of XML that enable us to browse and maintain semantic web data in with some set of well-defined methods. RDF additionally provides a means for publishing both human and machine readable expressions intended to encourage the reuse and extension of metadata semantics among semantic web research fields. These methods of RDF include standard mechanisms for representing semantics that are grounded in a simple, yet powerful, data model.

2.3.1 SPARQL and Querying RDF Data

It is intuitive that we need to run query in RDF data. We know that RDF triple has three parts: subject, predicate and object. SPARQL, the query language for RDF data recommended by the W3C, allows us to do so by using triple patterns. The SPARQL[8]

[7] https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/
[8] https://www.w3.org/TR/sparql11-query/

is a declarative query language similar to SQL for performing data manipulation and data definition operations on data represented as a collection of RDF statements. Every SPARQL Query has a head (or solution modifier) and a query body. The head provides the basis for categorizing different types of SPARQL Query solutions. The query body comprises a collection of of RDF statement patterns that represent the entity relationships to which a query is scoped.For example, if we consider the graph depicted in Figure 2.1, the following SPARQL query selects all the actors who played in "Big Bang theory".

```
SELECT ?actor WHERE {
?actor <rdf:type> <dbo:Actor> .
<dbr:Big_Bang_Theory> <dbo:starring> ?actor .
}
```

Here actor is a variable which is used in two triple patterns in a conjunctive way, that is, the same entity has both to be an actor and to play in the TV show. We use SPARQL query to extract the corresponding entity types for our entity type ranking task. This wonderful tool helped us to make the great use of KGs in our work.

3 Related Works

3.1 Knowledge Graph Embedding

In its present state, KG technology is far from fully matured, although link prediction is an effective approach to completing a KG. Various models have been proposed to address the link-prediction issue. The models proposed to date differ in terms of their scoring function.

First, we describe the notation used in this paper. A knowledge graph $G = \{(h, r, t)\} \subseteq E \times R \times E$ can be formalized as a set of triples, where E is the set of all entities and R is the set of all relations. A triple is denoted by (h, r, t), where h is the head entity, r is the relation, and t is the tail entity. The bold letters $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t}$ denote embeddings of h, r, and t, respectively, in an embedding space $\mathbb{R}^n$. $f_r(\boldsymbol{h}, \boldsymbol{t})$ is the scoring function of the model under consideration.

3.1.1 Translation-Distanced based Models

Unstructured Model (UM)

UM[33] is the preliminary image of TransE, considering only entities as embeddings. Because UM ignores relations, its scoring function is a simplification of that used in TransE. The scoring function is given as:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h} - \boldsymbol{t} ||_{l_{1/2}}, \tag{3.1}$$

where $\boldsymbol{h}$ and $\boldsymbol{t}$ are the embeddings of head and tail, respectively.

Structure Embedding (SE)

Bordes proposed the SE model[34], which introduces two different matrices to project separately the head and tail entities for each relation. Its scoring function is defined as follows:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{M}_{rh}\boldsymbol{h} - \boldsymbol{M}_{rt}\boldsymbol{t} ||_{l_{1/2}}, \tag{3.2}$$

where $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n$ and $\boldsymbol{M}_{rh}, \boldsymbol{M}_{rt} \in \mathbb{R}^{n \times n}$ are the role-specific projection matrices for the head entity and tail entity, respectively.

TransE, TransH, and TransR/CTransR

TransE [35] learns embedding as $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ where (h, r, t) holds. Therefore, $\boldsymbol{(h + r)}$ is very close to $\boldsymbol{t}$, when (h, r, t) holds. Here, the intuition is learning distributed word representations to capture linguistic regularities such as $Tokyo + CapitalOf \approx Japan$. TransE is the most popular translation-distance-based embedding model and is both very simple and fast.

Many researchers [36, 37] have claimed that TransE has problems in representing one-to-many, many-to-one, and many-to-many relations, with a number of models being proposed to address these issues.

The first such effort was TransH [37], which represents relations by hyperplanes. This model projects entities on the hyperplane corresponding to a relation. A single entity can have different representations on different hyperplanes. TransH models the relation r as $\boldsymbol{r}$ on a hyperplane with the normal vector $\boldsymbol{w_r}$. Given a triple (h, r, t),

the entity representations $\boldsymbol{h}$ and $\boldsymbol{t}$ are projected on the hyperplane of $\boldsymbol{w_r}$ with the restriction that $||\ \boldsymbol{w_r}\ ||= 1$. The calculation is expressed as:

$$\begin{aligned} \boldsymbol{h}_\perp &= \boldsymbol{h} - \boldsymbol{w}_r^\top \boldsymbol{h} \boldsymbol{w}_r, \\ \boldsymbol{t}_\perp &= \boldsymbol{t} - \boldsymbol{w}_r^\top \boldsymbol{t} \boldsymbol{w}_r. \end{aligned} \tag{3.3}$$

The scoring function is very similar to TransE:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = ||\ \boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp\ ||_{l_{1/2}}. \tag{3.4}$$

TransR [36] also addressed the flaws of TransE, but in a slightly different way than did TransH. TransR considers separate spaces for entities and relations, but the main principle is that entities and relations are completely different types of objects, implying that they should not occupy the same vector space. Given a triple (h, r, t), TransR projects the entity representations $\boldsymbol{h}$ and $\boldsymbol{t}$ into the space specific to a relation r. That is:

$$\boldsymbol{h}_r = \boldsymbol{M}_r \boldsymbol{h}, \quad \boldsymbol{t}_r = \boldsymbol{M}_r \boldsymbol{t}, \tag{3.5}$$

where $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n$, $\boldsymbol{r} \in \mathbb{R}^m$, and $\boldsymbol{M}_r \in \mathbb{R}^{n \times m}$ represents the projection matrix from the entity space to the relation space for relation r. The scoring function is:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = ||\ \boldsymbol{h}_r + \boldsymbol{r} - \boldsymbol{t}_r\ ||_{l_{1/2}}. \tag{3.6}$$

CTransR is an extension of TransR proposed by the same authors. In this model, entity pairs for a relation are clustered into different groups, and the pairs in the same group share the same unique relation vector.

TransD

TransD [3] can be considered as a special case of TransR. It replaces transfer matrix by the product of two projection vectors of an entity and relation pair. Specifically, for each triple (h, r, t), TransD introduces additional mapping vectors $\boldsymbol{h_d}, \boldsymbol{t_d} \in \mathbb{R}^n$ and $\boldsymbol{r_d} \in \mathbb{R}^m$, along with the entity or relation representations $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n$ and $\boldsymbol{r} \in \mathbb{R}^m$. Projection matrices for head/tail are accordingly defined as:

$$\begin{aligned} \boldsymbol{M}_{rh} &= \boldsymbol{r}_d \boldsymbol{h}_d{}^\top + \boldsymbol{I}, \\ \boldsymbol{M}_{rt} &= \boldsymbol{r}_d \boldsymbol{t}_d{}^\top + \boldsymbol{I}. \end{aligned} \tag{3.7}$$

These two projection matrices are then applied on the head entity $\boldsymbol{h}$ and the tail entity $\boldsymbol{t}$ respectively to get their projections, i.e.,

$$\hat{\boldsymbol{h}} = \boldsymbol{M}_{rh}\boldsymbol{h}, \quad \hat{\boldsymbol{t}} = \boldsymbol{M}_{rt}\boldsymbol{t}, \tag{3.8}$$

TransD obtains state-of-the-art performance on triplet classification and link prediction tasks.

TorusE

TorusE [38] addressed regularization problem of TransE. Regularization conflicts with the principle and makes the accuracy of the link prediction task lower. It introduced a torus, which is a compact Lie group that can be easily realized and achieved state-of-the-art performance.

RotatE

RotatE [39], which is able to model and infer various relation patterns including: symmetry/antisymmetry, inversion, and composition. Specifically, the RotatE model defines each relation as a rotation from the source entity to the target entity in the complex vector space. It showed that such a simple operation can effectively model all the three relation patterns: symmetric/antisymmetric, inversion, and composition. It achieved very impressive results in link prediction task.

HyperKG

HyperKG [40] exploits the hyperbolic space in order to better reflect the topological properties of knowledge bases. it focuses on the family of TransE that attempt to model the statistical regularities as vector translations between entities' vector representations, and whose performance has been lagging. HyperKG extended the translational models by learning embeddings of KB entities and relations in the Poincare-ball model of hyperbolic geometry.

3.1.2 Bilinear Models

RESCAL

For the link-prediction and triple-classification tasks, bilinear and neural-network-based models are also popular. RESCAL[41, 42] is a bilinear model, with each relation being represented by an *n-by-n* matrix in an embedding space $\mathbb{R}^n$ and the scores for the triples being calculated by a bilinear mapping.

DistMult

DistMult [43] simplifies RESCAL by restricting the matrices to diagonal matrices but it has problem with the score of (h, r, t) and (t, r, h) are the same.

ComplEx

ComplEx [4] addressed this issue of DistMult. It uses complex numbers instead of real numbers and takes the conjugate of the embedding of the tail entity before calculating the bilinear mapping.

TuckER

Recently proposed model TuckER [44], a relatively straightforward but powerful linear model based on Tucker decomposition of the binary tensor representation of knowledge graph triples. This model gained popularity for it's fully expressive feature. In TuckER for any given triple, it assumes that there exists an assignment of values to the entity and relation embeddings that accurately separates the true triples from false ones. TuckER is a generalization of many state-of-the-art linear models, e.g., RESCAL [41], DistMult [43], ComplEx [4] and SimplE [45], are special cases of TuckER.

3.1.3 Neural Network based Models

NTN

The SLM model[46], proposed by Socher, concatenates head and tail entities as an input layer to the nonlinear hidden neural layer and has the scoring function:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = \boldsymbol{u}_r^\top f(\boldsymbol{T}_{r1}\boldsymbol{h} + \boldsymbol{T}_{r2}\boldsymbol{t} + \boldsymbol{b}_r), \tag{3.9}$$

where $\boldsymbol{u}_r$ is a relation-specific linear layer, $\boldsymbol{T}_{r1}$ and $\boldsymbol{T}_{r2}$ are weighting matrices, $\boldsymbol{b}_r$ is a relation-specific bias vector and $f(\cdot)$ is the $tanh$ operation.

The NTN model[46] is an extension of the SLM model. It considers second-order correlations as inputs to nonlinear hidden neural networks. Its scoring function is:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = \boldsymbol{u}_r^\top f(\boldsymbol{h}^\top \boldsymbol{T}_r \boldsymbol{t} + \boldsymbol{T}_{rh}\boldsymbol{h} + \boldsymbol{T}_{rt}\boldsymbol{t} + \boldsymbol{b}_r), \tag{3.10}$$

where $\boldsymbol{T}_r$ represents a three-way tensor, the bilinear tensor product $\boldsymbol{h}^\top \boldsymbol{T}_r \boldsymbol{t}$ results in a vector, $\boldsymbol{T}_{rh}$ and $\boldsymbol{T}_{rt}$ denote weighting matrices, $\boldsymbol{b}_r$ is the bias, and $f(\cdot)$ is the $tanh$ operation. To date, NTN has proved computationally expensive and has scalability issues.

NAM

Neural Association Model (NAM) [47] conducts semantic matching with a Deep Neural Network (DNN) architecture. Given a fact (h, r, t), it first concatenates the vector embeddings of the head entity and the relation in the input layer, which gives $z^{(0)} = [\boldsymbol{h}; \boldsymbol{r}] \in \mathbb{R}^{2d}$. The input $z^{(0)}$ is then fed into a DNN consisting of L rectified linear hidden layers such that

$$\begin{aligned} \boldsymbol{a}^{(l)} &= \boldsymbol{M}^{(l)}\boldsymbol{z}^{(l-1)} + b^{(l)},\ l = 1, \ldots, L, \\ \boldsymbol{z}^{(l)} &= ReLU(\boldsymbol{a}^{(l)}),\ l = 1, \ldots, L \end{aligned} \tag{3.11}$$

where $\boldsymbol{M}^{(l)}$ and $b^{(l)}$ represent the weight matrix and bias for the l-th layer respectively. After the feed-forward process, the score is given by matching the output of the last hidden layer and the embedding of the tail entity, $f_r(\boldsymbol{h}, \boldsymbol{t}) = \boldsymbol{t}^\top \boldsymbol{z}^{(L)}$.

ConvE and ConvR

In recent times, several convolutional models have been proposed for solving link prediction task. Dettmers et al. [48] proposed a multi-layer convolutional network model ConvE which is very efficient in terms of time and space complexity compare to other NN based models proposed earlier. Another convolutional model called ConvR [6] which enabled rich interactions between entities and relation representations and

achieved the state-of-the-art performance in link prediction task. In this paper, we proposed a model based on translation distance based model but extending our model with convolutional models can be an interesting future work.

InteractE

InteractE [49] targets the limitations of convE. This model analyzes how increasing the number of these interactions affects link prediction performance, and utilize the observations. InteractE is based on three key ideas – feature permutation, a novel feature reshaping, and circular convolution. InteractE authors claimed that increasing the number of such interactions is beneficial to link prediction performance, and show that the number of interactions that ConvE can capture is limited. InteractE is a novel CNN based KG embedding approach which aims to further increase the interaction between relation and entity embeddings.

3.1.4 Discussion on Bilinear and Neural-Network(NN) based Models

However, bilinear models add more redundancy than do translation-distance-based models. For this reason, they can have an overfitting problem. TPRC considers only entities' type constraints in relational context, aiming to retain simple model estimation. It exploits linear mapping and involves less parameter overhead than the latent-variable models. Although neural-network-based models also tend to encounter overfitting, the standard advantage of such models is that they can capture many kinds of relations. Other issues of NN based models are time and space complexity compare to translation distance based models and bilinear models though recently proposed NN based models e.g., ConvE [48] are highly parameters efficient.

3.1.5 Models Incorporating Entity Types

SSE [50, 51] is another model which incorporates the category information into KG embeddings, which requires entities of the same type to stay close. It employs two manifold learning algorithm for representation learning based on semantic smoothness assumption i.e., Laplacian Eigenmaps (LE) [52] and Locally Linear Embedding (LLE)

[53]. SSE can be extended with other translation distanced based models, Bilinear and Neural Network based models. SSE used LE and LLE regularization methods to measure the smoothness of the embedding space. The major limitation of SSE is: SSE introduces some hard constraints like each entity happens to belong to only one category, which is inconsistent with the KG. Another problem is each entity has same representation for all the relations where it appears.

TKRL proposed by Xie et al. [5], which considers hierarchical entity categories. The scoring function is accordingly defined as:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \ \boldsymbol{M}_{rh}\boldsymbol{h} - \boldsymbol{r} + \boldsymbol{M}_{rt}\boldsymbol{t} \ ||_{l_1}, \tag{3.12}$$

where $\boldsymbol{M}_{rh}$ and $\boldsymbol{M}_{rt}$ are projection matrices for h and t. $\boldsymbol{M}_{rh}/\boldsymbol{M}_{rt}$ can be presented as a weighted sum of all possible type matrices for each entity, i.e.,

$$\boldsymbol{M}_{rh} = \frac{\sum_{i=1}^{n_h} \alpha_i \boldsymbol{M}_{c_i}}{\sum_{i=1}^{n_h} \alpha_i}, \alpha_i = \begin{cases} 1, & c_i \in C_{rh} \\ 0, & c_i \notin C_{rh} \end{cases} \tag{3.13}$$

where n_h is the number of categories to which h belongs; c_i is the i-th category among the set of types of an entity; $\boldsymbol{M}_c$ is the projection matrix of c_i; α_i the corresponding weight; and C_{rh} the set of types of an entity in relation r. TKRL employed two types of encoders to compute the projection matrix $\boldsymbol{M}_c$: Recursive Hierarchy Encoder (RHE) and Weighted Hierarchy Encoder (WHE). They are defined in [5] as follows:

$$\begin{aligned} RHE \ &: \ \boldsymbol{M}_{c_i} = \boldsymbol{M}_{c_i{}^{(1)}} \boldsymbol{M}_{c_i{}^{(2)}} \ldots \boldsymbol{M}_{c_i{}^{(l)}} \\ WHE &: \ \boldsymbol{M}_{c_i} = \beta_1 \boldsymbol{M}_{c_i{}^{(1)}} + \cdots + \beta_l \boldsymbol{M}_{c_i{}^{(l)}} \end{aligned} \tag{3.14}$$

Here $c_i{}^{(1)}, \ldots, c_i{}^{(l)}$ are sub-types of c_i in the hierarchy; $\boldsymbol{M}_{c_i{}^{(1)}}, \ldots, \boldsymbol{M}_{c_i{}^{(l)}}$ their projection matrices; and $\beta_1, \ldots, \beta_l$ the corresponding weights. They also applied a sampling method called Soft Type Constraint (STC) with their encoders. TKRL is an extended version of original TransE. It has very high time and space complexity since each entity may have multiple sub-types. On an average TKRL considered 8 sub-types for each entity for FB15k dataset. That means for each entity there are 8 type-embodied matrices. Another model proposed by Krompaß[54] which is a latent-variable model that also consider relation and entity type constraints.

3.2 Entity Type Ranking

This section briefly describes the related works done so far in the entity type ranking field and connecting fields.

3.2.1 TRank: An Entity Type Ranking System

An entity may be associated to more than one single generic type. TRank[1] first introduced the entity type ranking task [55]. Authors evaluated several type-hierarchy and graph-based approaches that exploit both schema and instance relations. We briefly discuss the approaches they proposed in the following paragraphs:

FREQ: In this approach entity types have been ranked based on the frequency of the type in the background knowledge base system. The most frequent type of an entity will be ranked first.

WIKILINK: It focuses on relations attached to the entities. It uses the number of neighboring entities (i.e., outgoing and incoming links to that specific entity) that share the same entity type.

SAMEAS: TRank exploits the entity graph from the knowledge base by following $< owl : sameAs >$ connections and observes the types attached to specific URIs.

LABEL: Using a text similarity based approach it exploits TF-IDF similarity for ranking the types.

SAMETYPE: This approach takes into account the context in which the entity has appeared. It considers the frequency of a type co-occuring with the entity in the same context.

PATH: This is a variant of SAMETYPE approach which exploits both type hierarchy and the context where the entity appears.

DEPTH: It identifies the depth of an entity type in the knowledge graph type hierarchy in order to assess its relevance. This methods has higher computational complexity than the previous methods.

ANCESTORS: It takes into consideration how many ancestors of a specific type are also a type of a particular entity.

ANC_DEPTH: It is a variant of the *ANCESTORS* approach. It considers not just the

[1] https://github.com/eXascaleInfolab/TRank

number of ancestors of a type, but also their depth by exploiting the knowledge graph type hierarchy.

As TRank defined many approaches, authors also combined these approaches to retrieve the best types by using decision trees (DEC_TREE) and linear regression (LINREG).

These entity type ranking approaches are different than the approach we present In this chapter while we aim at solving the same task. For this reason, we evaluatad our methods and compared our results over the same datasets and using the same ground truth.

3.2.2 Knowledge Graph Embeddings and Type Ranking

Knowledge graphs are being used in many AI based tasks such as inferring new knowledge, question answering (Q&A), relations in social network applications, item recommendation, etc. Knowledge graph embeddings refer to vector representations of entities and relations that attempt to preserve the structure and the semantics of the knowledge graph. Several embedding models have been proposed for knowledge graph completion. Roughly, they can be divided into three categories: bilinear models, neural network based models, and translation distance based models.

RESCAL [41, 42] is a bilinear model, each relation is represented by an *n-by-n* matrix on an embeding space $\mathbb{R}^n$ and the score of the triples are calculated by a bilinear mapping. DistMult [43] simplifies RESCAL by restricting the matrices to diagonal matrices but it has problem with the scores of (h, r, t) and (t, r, h) are the same. ComplEx [4] addressed the issue of DistMult. It uses complex numbers instead of real numbers and takes the conjugate of the embedding of the tail entity before calculating the bilinear mapping. ComplEx has shown impressive preformace over multi-relational KGs where each entity is involved in many relations with other entites.

SLM [46] proposed by Socher is an expressive model that concatenates head and tail entities as an input layer to a non-linear neural hidden layer. The NTN model [46] is an extension of the SLM model that considers second-order correlations into non-linear hidden neural networks.

Bordes proposed the Distant and SE model [56, 57, 34] that defines triples using a distance in the vector space. TransE [35] is the most popular translation based

embedding model, which is very simple and fast. Many researchers claimed that TransE has flaws in representing 1 to many, many to 1, and many to many relations. Many models have later been proposed to solve such issues. The very first effort is TransH [37]. It represents relations by hyperplanes. This model then projects entities on the hyperplane corresponding to a relation. TransR [36] also addressed some of the flaws of TransE, in a slight different way than TransH. TransR considers separate spaces for entities and relations. Their main principle is: entities and relations are completely different types of objects, so they should not be in the same vector space.

Recently, several KG embedding models have been proposed in the literature considering the entity type constraints [54, 14]. The 2017 edition of the WSDM Cup consisted of a triple scoring task [58] that focused on two relations: "profession" and "nationality". One person/entity may have multiple professions or nationalities, and this task aimed to find out the most relevant professions or nationalities by scoring the triples from a very limited ground truth data.

Knowledge graph representation learning showed promising results to enhance the performance of recommender systems. He [59] proposed a model where the selection of an item satisfies the translational relation in the latent vector space and the relations are regarded as related to users in sequential recommendations. Another model [60] showed that item and relations can be modeled via memory-based attention networks. Zhang [61, 7] showed how a unified graph can be constructed by adding relations between users and items. Piao [62] summarized recommendation results using different entity embeddings and found that node2vec [63] improves recommendation result quality.

In our data (typed triples) we do not have any symmetric or transitive relations. The relational structure is very simple. The head is always an entity and the tail is the corresponding entity type. However, bilinear models add more redundancy than do translation-distance-based models. For this reason, they can have an overfitting problem. Since the triple structure in our dataset is straightforward, only translational models have been employed as structured embedding in our proposed RL-TRank models.

3.2.3 Recommender Systems Using Knowledge Graphs

Knowledge graph representation learning showed promising results to enhance the performance of recommender systems. He [59] proposed a model where the selection of an item satisfies the translational relation in the latent vector space and the relations are regarded as related to users in sequential recommendations. Another model [60] showed that item and relations can be modeled via memory-based attention networks. Zhang [61, 7] showed how a unified graph can be constructed by adding relations between users and items. Piao [62] summarized recommendation results using different entity embeddings and found that node2vec [63] improves recommendation result quality.

3.3 Triple Scoring

This section provides an overview of the triple scoring task ,including a description of the task and the dataset, an overview of the participating teams and their approaches. In a nutshell, the task was to compute relevance scores for knowledge-base triples from relations, where such scores make sense. Due to the way the ground truth was constructed, scores were required to be integers from the range 0 to 7. For example, reasonable scores for the triples Tim Burton profession Director and Tim Burton profession Actor would be 7 and 2, respectively, because Tim Burton is well-known as a director, but he acted only in a few lesser known movies.

We exploited the WSDM Cup 2017 [1] challenges' dataset is to evaluate our model on triple scoring task. These datasets have the relevance scores for triples from type-like relations, more specifically, for profession / nationality types. The relevance score is used to indicate the strong bonding and quantifies the relationship between entities and it's corresponding types in terms of the degree to which an entity belongs to a type. Participants were provided several text files1 , including: 1) persons - contains more than 380,000 different entity names used for this task, 2) profession.kb/nationality.kb - each contains all of the profession / nationality tuples for a set of people, respectively, 3) profession.train/nationality.train - contains a subset of tuples from the respective .kb file together with their relevance scores generated based on human judgment, 4) professions/nationalities - contains a list of the 200 / 100 different professions /

nationalities from the respective .kb file, and 5) wiki-sentences - contains more than 33 million sentences from Wikipedia with annotations of these people. The goal was to find out most relevant types (profession/nationality) associated with those specific entities.

3.3.1 Most Attractive Methods

Total 21 teams submitted a valid run on this competition. Here we describe the key approaches and a brief description of the key teams.

Gailan

The team Gailan [64] generates a relevance score based on the textual description of the triple's subject and Object. It measures how similar (related) the text description of the subject is to the text description of its values. The generated similarity score can then be used to rank the multiple values associated with this subject entity. Here, the authors employed the Paragraph Vector algorithm to represent the unstructured text into fixed length vectors. The fixed length representation is then utilized to calculate the similarity (relevance) score between the subject and its multiple values.

Bokchoy

BOKCHOY[2] proposed an ensemble model for addressing the triple scoring task. Their main idea is similar to [1] and they add another additional component called path ranking approach [65] . As base scorers, BOKCHOY utilized four components e.g., employ word classification [1], word counting [1], word MLE [1], and also path ranking [65]. The first three base scorers find witnesses on the basis of Wikipedia, while the last one further makes use of Freebase [26]. The final scorers are then merged into an ensemble by weighted averaging. After that, an elongation step is introduced to further refine outputs of the ensemble scorer. Specifically, we create, for each target type, a list of trigger words by using publicly available lexical resources like WordNet. Trigger words of the profession Athlete, for example, may include hyponyms of athlete, such as runner and jumper. Given a triple, we detect, from the first sentence of the entity's Wikipedia page, occurrences of such trigger words, and accordingly refine the output of the ensemble

Cabbage

This team has focused on extracting features related to task difficulty [66]. Crowdworkers judge the relevance of multiple professions/nationalities for each person; hence, the difficulty of the task depends on the number of professions/nationalities. In addition, the popularity of the person or the familiarity of the professions/nationalities should facilitate the judgment. So they extracted these features from the data provided for the challenge organizers. The correlation between these extracted features and the judgment discrepancy of crowdworkers were investigated. Cabbage incorporated these crowdsourcing-specific features into the prediction model of relevance scores, in addition to the relevance features similar to those introduced in the [1].

Lettuce

Lettuce [67] proposed an approach wherein the outputs of multiple neural network classifiers are combined using a supervised machine learning model. This achieved very satisfactory results one out of three measures (i.e., Kendall's τ), and performed competitively in the other two measures (i.e., accuracy and average score difference).

The approach is: given a KB entity e and its target type t, our method predicts a score that represents the relevance of e belonging to t. Here, we adopt a two-step approach: the first step is a classification step that aims to estimate the probability of e belonging to t ($P(t|e)$) using multiple neural network-based classifiers. In the second step authors introduced a scoring function that uses a supervised machine learning model to convert the outputs of these classifiers to the target relevance score.

Catsear

This team proposed a hybrid approach for triple scoring that combines results from three different sources and they are: Path, Graph Cross and Skip Gram modules to gather information from the sources [68]. They also introduced a super classifier module to learn the trustworthiness scores associated with them. This approach showed good accuracy in the competition.

Celosia

Celosia [69] proposed a model which could automatically predict the relevance scores for unseen triples. In this model authors have chosen some very interesting features and exploited Logistic Ordinal Regression based classification to predict the outcome. The proposed system achieves an overall accuracy score of 0.73 and Kendall's tau score of 0.36.

Chicory

Chicory [70] deployed a large collection of entity tagged web data to estimate the correctness of the relevance relation of the type-like triples, in combination with a baseline approach using Wikipedia abstracts same as the Bat et. al. [1] . They utilized ClueWeb12 annotated by Google's entity linker, available publicly as the FACC1 dataset for relevance estimations. It is an automatically system that specified declaratively how the input data are combined into a final ranking of triples.

4

Leveraging Entity Types in Knowledge Graph Embedding

Knowledge graph embedding aims to embed entities and relations of multi-relational data in low dimensional vector spaces. Knowledge graphs are useful for numerous artificial intelligence (AI) applications. However, they (KGs) are far from completeness and hence KG embedding models have quickly gained massive attention. Nevertheless, the state-of-the-art KG embedding models ignore the category specific projection of entities and the impact of entity types in relational aspect. For example, the entity *"Washington"* could belong to the person or location category depending on its appearance in a specific relation. In a KG, an entity usually holds many type properties. It leads us to a very interesting question: are all the type properties of an entity are meaningful for a specific relation? In this paper, we propose a KG embedding model TPRC that leverages entity-type properties in the relational context. To show the effectiveness of our model, we apply our idea to the TransE, TransR and TransD. Our approach outperforms other state-of-the-art approaches as TransE, TransD, DistMult

and ComplEx. Another, important observation is: introducing entity type properties in the relational context can improve the performances of the original translation distance based models.

4.1 Introduction

Knowledge graphs encode structured facts of real world entities and their rich relations. An increasingly large number of organizations such as Google, Microsoft, Facebook or IBM create and maintain large KGs, as they aid in the integration of heterogeneous data sources, complex query resolution, and structured knowledge exploration. Wikidata [21], DBpedia [71], YAGO [9], and Freebase [10] incorporate very large numbers of facts that facilitate many AI tasks e.g., entity recommendations [7], question answering [72], recommender systems [73], [74] etc. Although existing KGs contain billions of entities and relations, they still have gaps and may contain incorrect facts.

Traditionally, KGs represent the relations/facts between their various entities as triples. A triple can be represented as (h, r, t), where h and t are entities in the real world and r is a relation between h and t. For example, consider the triple (*Tokyo, CapitalOf, Japan*), where *Tokyo, Japan* are the head and tail entities, respectively, and *CapitalOf* is the relation. KG completion problem is similar to link prediction in social network. The purpose of link prediction is to detect unknown pairs of head and tail entities that are correlated via some relation. For example, if the KG contains facts like (*ShinzōAbe, PrimeMinisterOf, Japan*) and (*AkieAbe, SpouseOf, Shinzō Abe*) but the fact (*AkieAbe, HasNationality, Japan*) is not stored, then we would like the machines to complete the missing link between the entity *AkieAbe* and entity *Japan* automatically by link prediction.

The concept of "embedding" has also been widely used for representing words and texts[75, 16], with many embedding models having been proposed for KG completion. Most of these models fall into one of three categories: bilinear models, neural-network-based models, and translation-distance-based models.

The translation-distance-based models have gained popularity both for their simplicity and their effectiveness, where they have achieved state-of-the-art performance. Bordes et al.[35] proposed TransE, which is the simplest and smartest way of predicting the links in a KG. TransE was inspired by Mikolov's skip-gram

model [16, 76]. It learns vector embeddings for entities and relations, with relations being represented as translations in the embedding space. The basic principle is that $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$, where (h, r, t) holds. Here, $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}$ are each embeddings of h, r, t, respectively. To solve the one-to-many/many-to-one/many-to-many issues in TransE, TransH[37] has been proposed. It involves a principle stating that entity representations will differ based on various relations. Similarly, TransR[36] assumes that each relation has its own embedding space. However, TransR proposes using separate spaces for entities and relations. TransD [3] proposed dynamic mapping matrix to improve the performance of TransR, which showed very significant results compared to other translation-distanced-based models.

In these models, the entities' type has been completely ignored. In the real world, entities can be categorized in terms of several types, such as person, movie, or organization. We can often assume that entities of the same type should share strong similarities and that, in their relation, their type also plays an important role. As an example, the *HasNationality* relation requires a person-type head entity and a location/country-type tail entity. On the other hand, the *CapitalOf* relation requires location-type entities for both head and tail. We can imagine the existence of two different triples for these two relations: (*Washington, HasNationality, U.S.*) and (*Washington, CityOf, U.S.*). Here, the entity "Washington" plays two completely different roles in these two relations, based on their type. However, an entity is usually not associated to a single generic type but rather to a set of more specific types in the context of a specific relation. As an example, in Freebase entity *DonaldJhonTrump* has 32 types information including "Person", "Organization founder", "Businessman", "Celebrity", "Politician", "Actor", "Architectural structure owner". Consider two triples: (*DonaldJhonTrump, PresidentOf, USA*) and (*DonaldJhonTrump, StarredIn, HomeAlone2*). In the first triple, the most expressive type of the head entity should be "Politician" in the context of relation *PresidentOf* and appropriate type of the head entity *DonaldJhonTrump* for second triple would be "Actor". Fortunately, Freebase provides very rich *rdfs#domain* and *rdfs#range* information of entity types for each relation, considering the relational context.

In our model, we explicitly define the role of entity type in a relation. We propose a model that leverages entity type properties in the relational context (TPRC), where, for each relation r, entities are mapped based on both type and relationship.

In [14], we exploited the basic entity type information of entities e.g., "Person", "Location" etc. The basic type information for real-world entities had been collected from the schema.org[1] vocabulary. According to the definitions in schema.org, there are 10 basic types of real-world entities. Although one entity may involve in various subcategories/subtypes, in [14], we focused only on basic entity types. So the representation for one entity was same for every relation. Later, we propose a more fine-grained model, which introduces entities' type mapping matrix considering relational context [13]. TPRC defines the role of types in the head or tail entity more explicitly and clearly. For TPRC entity representation changes based on their role and relation. Our model can be easily combined with other state-of-the-art models to produce more accurate predictions. We evaluated our model using two tasks that involve link prediction and triple classification on the standard datasets FB15K, FB15k-237, YAGO3-10 and FB13.

4.2 Complexity Analysis

The parameters and the complexity of the related works (see Section 3.1) are listed in Table 4.1. Here, N_e and N_r are the number of entities and relations respectively; n and m the dimensionality of entity and relation embedding space respectively (here, we assume $n = m$);$\bar{t}$ is the number of relations specific entity types; $\bar{n}$ the number of entities which have type information (for SSE model); $\bar{k}$ the number of sub-types for each entity; and L is the total number of hidden layers in the network in NAM model. SSE [50, 51] extended several models (e.g., TransE[35], RESCAL[41], SME[57] etc) using LE/LLE regularization as mentioned earlier in Section . In Table 4.1 *-LE/*-LEE denotes a model with the LE/LLE regularizer included. For *-LE/*-LEE: $\boldsymbol{E}$ is a matrix consisting of the entity embeddings; $\boldsymbol{D}, \boldsymbol{I} \in \mathcal{R}^{(n \times n)}$ are the diagonal and Identity matrices respectively. During the analysis we assume that $n, m \ll N_e$and all the models are trained under the open world assumption.

We can draw the following conclusions. First, models which represent entities and relations as vectors (e.g., TransE, TransH, TransD, and ComplEx) are more efficient. They usually have space and time complexity that scales linearly with n. Second,

[1] http://schema.org

models which represent relations as matrices (e.g., TransR, $\text{TPRC}_{\text{TransD}}$, SE, and RESCAL) usually have higher complexity in both space and time, scaling quadratically or cubically with the dimensionality of embedding space. Third, models based on neural network (e.g., SME, NTN, and NAM) generally have higher complexity in time, if not in space, since matrix or even tensor computations are often required in these models (recent NN based models are very efficient in terms of time and space complexity e.g. ConvE [77]). Finally, TPRC incorporates relations specific types with the state-of-the-art translation based models which increase the complexity on updating embeddings. $\text{TPRC}_{\text{TransD}}$ achieves (see section) the better performance in our experiment and it's time complexity is $O(n^2)$, which is larger than TransD original model but considering the performance issue we think it's reasonable.

4.3 Our Method

Translation-distance-based embedding models mostly follow TransE. Both TransE and TransH assume embeddings of entities and relations within the same space $\mathbb{R}^n$. However, relations and entities are completely different objects and it may not be appropriate to represent them in a common semantic space. Although TransH extends modeling flexibility by employing relation hyperplanes, it does not fully address the restrictions of this assumption. In contrast, the entities in TransR are mapped to vectors in different relational spaces, according to their relations. TransD considers the diversity of relations and entities. However, none of these models consider the significance of entity type. Therefore, they cannot judge the exact role of each entity, based on its relation. In our model, we deliberately include the type information. The entity's type can be incorporated easily by introducing an entity type-mapping matrix. For the relations, the entity type information plays a significant role. For example, the "*CapitalOf*" relation would imply that both the head and the tail would be location type entities. If we think more precisely and look for more compact types, then we can understand that, the type of the head entity would be country and the type of the tail entity would be city/town, both of them are sub-types of location. As we mentioned earlier we can obtain such kind of relation-wise rich information from Freebase. Therefore, we propose a more fine-grained model to map the entities with their corresponding types considering relational context. Using type information in a

Table 4.1: Parameters and complexity of related works.

Models	Parameters	Space Complexity	Time Complexity
RESCAL [41]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{W}_r \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^2)$	$O(n^2)$
RESCAL-LE [51]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{R}_k, \boldsymbol{W}_r \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n^2)$	$O(n^2)$
RESCAL-LLE [51]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{W}_r \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 K n + N_r n^2)$	$O(n^2)$
BILINEAR [78]	$\boldsymbol{h}\boldsymbol{W}_r t, \boldsymbol{W}_r, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^2)$	$O(n^2)$
BILINEAR-LE [51]	$\boldsymbol{h}\boldsymbol{W}_r t, \boldsymbol{W}_r, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n^2)$	$O(n^2)$
BILINEAR-LLE [51]	$\boldsymbol{h}\boldsymbol{W}_r \boldsymbol{t}, \boldsymbol{E}, \boldsymbol{I} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n} K n + N_r n^2)$	$O(n^2)$
DistMult [43]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n + M_r n)$	$O(n)$
ComplEx [4]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^n$	$O(N_e n + M_r n)$	$O(n)$
UM [33]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n)$	$O(n)$
SE [34]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{M}_{rh}, \boldsymbol{M}_{rt} \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^2)$	$O(n^2)$
SE-LE [51]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n^2)$	$O(n^2)$
SE-LLE [51]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{I}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n} K d + N_r n^2)$	$O(n^2)$
SME (lin) [57]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n + N_r d)$	$O(n^2)$
SME(lin)-LE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n)$	$O(n^2)$
SME(lin)-LLE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{I}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n} K n + N_r n)$	$O(n^2)$
SME (bilin) [57]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e d + N_r n)$	$O(n^3)$
SME(bilin)-LE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n)$	$O(n^3)$
SME(bilin)-LLE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{I}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n} K n + N_r n)$	$O(n^3)$
TKRL [5]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{M}_{rh}, \boldsymbol{M}_{rt} \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^k)$	$O(n^k)$
TorusE [38]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n + M_r n)$	$O(n)$
SLM [46]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r} \in \mathbb{R}^n, \boldsymbol{M}_r^1, \boldsymbol{M}_r^2 \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^2)$	$O(n^2)$
NTN [46]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{u}_r, \boldsymbol{b}_r \in \mathbb{R}^n, \boldsymbol{T}_r \in \mathbb{R}^{n \times n \times n}, \boldsymbol{T}_{r,h}, \boldsymbol{T}_{r,t} \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^3)$	$O(n^3)$
NAM [47]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n + N_r n)$	$O(Ln^2)$
ConvE [77]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r} \in \mathbb{R}^n$	$O(N_e n + N_r n)$	$O(n)$
TransE [35]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n$	$O(N_e n + N_r n)$	$O(n)$
TransE-LE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{D} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n}^2 n + N_r n)$	$O(n)$
TransE-LLE [51]	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{E}, \boldsymbol{I} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{n} K n + N_r d)$	$O(n)$
TransH [37]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r}, \boldsymbol{W}_r \in \mathbb{R}^n$	$O(N_e n + N_r n)$	$O(n)$
TransR [36]	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r} \in \mathbb{R}^n, \boldsymbol{M}_r \in \mathbb{R}^{n \times n}$	$O(N_e n + N_r n^2)$	$O(n^2)$
TransD [3]	$\boldsymbol{h}_d, \boldsymbol{t}_d, \boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r}_d, \boldsymbol{r} \in \mathbb{R}^n$	$O(N_e n + N_r n)$	$O(n)$
$\text{TPRC}_{\text{TransE}}$ (this paper)	$\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^n, \hat{\boldsymbol{M}}_{rh}, \hat{\boldsymbol{M}}_{rt} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{t} n^2 + N_r n)$	$O(n^2)$
$\text{TPRC}_{\text{TransR}}$ (this paper)	$\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r} \in \mathbb{R}^n, \hat{\boldsymbol{M}}_{rh}, \hat{\boldsymbol{M}}_{rt} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{t} n^2 + N_r n^2)$	$O(n^2)$
$\text{TPRC}_{\text{TransD}}$ (this paper)	$\boldsymbol{h}_d, \boldsymbol{t}_d, \boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n, \boldsymbol{r}_d, \boldsymbol{r} \in \mathbb{R}^n, \hat{\boldsymbol{M}}_{rh}, \hat{\boldsymbol{M}}_{rt} \in \mathbb{R}^{n \times n}$	$O(N_e n + \bar{t} n^2 + N_r n)$	$O(n^2)$

translation model can improve the efficiency of any such model.

An entity may belong to multiple sub-types and it's also possible to obtain hierarchical types of an entity. If we consider the subcategories in the hierarchy when modeling, it will make the model complex, thereby the time and space complexity will increase significantly. To address this issue in our model, we consider only the domain and range of the types for a specific relation.

Considering the diversity of entity types in different relations we propose TPRC. For each relation r, we introduce type-embodied mapping matrices $\hat{\boldsymbol{M}}_{rh}, \hat{\boldsymbol{M}}_{rt} \in \mathbb{R}^{n \times n}$ for the head and tail entity based on relational context. It means the type mapping matrix for an entity can be different based on the relation it appears. With the mapping

matrices, we defined the projected vectors of entities as:

$$\boldsymbol{h}_p = \hat{\boldsymbol{M}}_{rh}\boldsymbol{h}, \quad \boldsymbol{t}_p = \hat{\boldsymbol{M}}_{rt}\boldsymbol{t}, \tag{4.1}$$

where $(\boldsymbol{h}, \boldsymbol{t}) \in \mathbb{R}^n$. In our model, we enforce the constraints $|| \boldsymbol{h} ||_2 \leq 1$, $|| \boldsymbol{t} ||_2 \leq 1$, $|| \hat{\boldsymbol{M}}_{rh}\boldsymbol{h} ||_2 \leq 1$, and $|| \hat{\boldsymbol{M}}_{rt}\boldsymbol{t} ||_2 \leq 1$. It is not mandatory to have the same dimensionality for entity embeddings and entities' type embeddings. However, in our experiments to learn vectors and matrices, we keep the same dimensionality. The scoring function for each specific relation r (where $\boldsymbol{r} \in \mathbb{R}^n$) is correspondingly defined as:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h}_p + \boldsymbol{r} - \boldsymbol{t}_p ||_{l_{1/2}}. \tag{4.2}$$

TPRC can be easily combined with other state-of-the-art translation-distance-based models due to its simplicity. In this paper we apply it to TransE, TransR and TransD.

TransE

TransE is the most representative translational distance based model. The score function of TransE is defined as:

$$f(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} ||_{l_{1/2}}, \tag{4.3}$$

where $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^n$ and $\boldsymbol{r} \in \mathbb{R}^n$ are the vectors of a head entity, a tail entity, and a relation on a single embedding space.

If we incorporate TransE into the entities' type-mapping matrix model, denoted by $\text{TPRC}_{\text{TransE}}$ then the model's scoring function will be given by Eq. (12). TransE has the same vector representation for each entity. In $\text{TPRC}_{\text{TransE}}$, the head and tail entities are mapped according to their type based on the relation they appear.

TransR

In TransR, entities are mapped separately to a relation space. Equations (5) and (6) denote the embeddings and the scoring function of the TransR method. If we incorporate TransR into the entities' type-mapping matrix model ($\text{TPRC}_{\text{TransR}}$), then

the embeddings of head and tail entities become (we keep the same dimension for entity and relation vectors):

$$\boldsymbol{h}_p^r = (\hat{\boldsymbol{M}}_{rh}\boldsymbol{h})\boldsymbol{M}_r, \quad \boldsymbol{t}_p^r = (\hat{\boldsymbol{M}}_{rt}\boldsymbol{t})\boldsymbol{M}_r \tag{4.4}$$

and the scoring function is defined as:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h}_p^r + \boldsymbol{r} - \boldsymbol{t}_p^r ||_{l_{1/2}}. \tag{4.5}$$

The TransR mapping matrix $\boldsymbol{M}_r$ is the same for both head and tail entities. The key difference between our proposed model ($\text{TPRC}_{\text{TransR}}$) and the traditional TransR is that, before projecting to a relation space, entities are mapped via the entities' type-mapping matrix. In the relation space, entities therefore have vector representations based on their type and a specific relation property.

TransD

Different types of entities have different attributes and functions. It's the principle of TransD. The concept of TransD is very close to TPRC. This model defines the interaction between the vectors of different types of entities and relations (see Eq. (7)) implicitly. On the other hand, TPRC explicitly care about the entity types based on the relation. By applying, TPRC to TransD ($\text{TPRC}_{\text{TransD}}$), we define more fine-graned representations of entities and relations than TransD itself. The score function for TransD after projecting the entities into relational space is (Eqs. (7) and (8) define the projection procedure):

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \hat{\boldsymbol{h}} + \boldsymbol{r} - \hat{\boldsymbol{t}} ||_{l_{1/2}}. \tag{4.6}$$

For $\text{TPRC}_{\text{TransD}}$ the embeddings of the head h and tail t are:

$$\boldsymbol{h}_p^d = (\hat{\boldsymbol{M}}_{rh}\boldsymbol{h})\boldsymbol{M}_{rh}, \quad \boldsymbol{t}_p^d = (\hat{\boldsymbol{M}}_{rt}\boldsymbol{t})\boldsymbol{M}_{rt} \tag{4.7}$$

and the scoring function is defined as:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h}_p^d + \boldsymbol{r} - \boldsymbol{t}_p^d ||_{l_{1/2}}. \tag{4.8}$$

Table 4.2: Key differences with other type-constraints models

Models	Type	Addressed polysemy problem?	Incorporate with other models possible ?
TKRL	hierarchical	NO	NO
SSE	single	NO	**YES**
TPRC	multi-type	**YES**	**YES**

The design of the model TransR and TransD is identical except the mapping matrices. So the main difference between $\text{TPRC}_{\text{TransR}}$ and TransD is similar to the difference of $\text{TPRC}_{\text{TransD}}$ and TransD, before projecting to a relation space, entities are mapped via the entities' type-mapping matrices $\hat{\boldsymbol{M}}_{rh}\boldsymbol{h}, \hat{\boldsymbol{M}}_{rt}\boldsymbol{t}$ for head and tail accordingly.

Contribution and Key Differences with Other Models

Rich semantic information, such as types, descriptions, and other textual information, is an important supplement to structured information in KGs. Existing state-of-the-art models e.g., TransE, ComplEx focus on the structured information of triples and maximize the likelihood of them. However, they neglect semantic information contained in most knowledge graphs and the prior knowledge indicated by the semantic information. We introduce the type in the relational context to address the polysemy issue and make the entity representation for meaningful. Table 4.2 shows the key differences of TPRC model with most relevant models e.g., SSE and TKRL. SSE assumes that each entity belongs to one type only and on the other hand TKRL introduces hierarchical type information but neglects the polysemy issue. TKRL is also a very expensive model and unlike TPRC, it cannot be extended with other models.

In the above discussion, we have shown how to combine TPRC with the state-of-the-art translational distance based models. In the same way, we could combine TPRC with other existing translation-distance-based models.

4.4 Training

Earlier studies showed that negative sampling plays an important role in the model performance for knowledge graph embedding. Though, margin based loss function is very popular in translational model but they use the same ratio for positive and negative sampling. In this thesis, we use two type of loss functions: 1) Margin based loss function and 2) Log-sigmoid loss function.

4.4.1 Marginal Loss Function

We use a margin-based loss function for training:

$$L_M = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} max(0, f_r(\boldsymbol{h},\boldsymbol{t}) + \gamma - f_r(\boldsymbol{h}',\boldsymbol{t}')). \tag{4.9}$$

Here, γ is the margin, S is the set of correct triples, and $S^{'}$ is the set of incorrect triples. The ratio of the correct and incorrect samples is same, as we can guess it from the objective function. Existing KGs should only contain correct triples. An $S^{'}$ is constructed by replacing a head or a tail entity in an existing triple as follow [35]:

$$\begin{aligned} S^{'} = \{(h^{'}, r, t) | h^{'} \in E \wedge e^{'} \neq h \wedge (h, r, t) \in S\} \\ \cup \{(h, r, t^{'}) | t^{'} \in E \wedge t^{'} \neq t \wedge (h, r, t) \in S\} \end{aligned}$$

We also employ two strategies "unif" and "bern" reported in [37] to replace head or tail entity. We use a stochastic gradient descent (SGD) method to minimize L. TransR and TransD initialize the embeddings of entities and relations obtained from TransE. To avoid overfitting of $\text{TPRC}_{\text{TransR}}$ and $\text{TPRC}_{\text{TransD}}$, we also initialize entity and relation embeddings with the results of $\text{TPRC}_{\text{TransE}}$.

4.4.2 Log-sigmoid Loss Function

In the bilinear models we have observed the effectiveness positive and negative sampling ratio. Here we use a loss function similar to the negative sampling loss as word2vec for effectively optimizing distance-based models. It is also inspired by

[79, 39]. The loss function is as follows:

$$L_s = -log\ \sigma(\gamma - f_r(\boldsymbol{h}, \boldsymbol{t})) - \sum_{i=1}^{N} \frac{1}{N} log\ \sigma(f_r(\boldsymbol{h_i'}, \boldsymbol{t_i'}) - \gamma) \tag{4.10}$$

where γ is a fixed margin, σ is the sigmoid function, and $(\boldsymbol{h_i'}, r, \boldsymbol{t_i'})$ is the i-th negative triplet, N is the number of the negative samples per positive ones during training.

Self-adversarial Negative Sampling (SNS) has employed in this case. This method samples negative triples from the following distribution:

$$p(h_j', r, t_j')|\{(h_i, r, t_i)\} = \frac{exp\ \alpha f_r(\boldsymbol{h_j'}, \boldsymbol{t_j'})}{\sum_i exp\ \alpha f_r(\boldsymbol{h_i'}, \boldsymbol{t_i'})} \tag{4.11}$$

where α is the temperature of sampling. Therefore, the final negative sampling loss with self-adversarial training takes the following form:

$$L_S = -log\ \sigma(\gamma - f_r(\boldsymbol{h}, \boldsymbol{t})) - \sum_{i=1}^{N} \frac{1}{N} p(h_j', r, t_j')\ log\ \sigma(f_r(\boldsymbol{h_i'}, \boldsymbol{t_i'}) - \gamma) \tag{4.12}$$

$\text{TPRC}_{\text{TransD}}$ shows promising results so we applied loss function technique to it and show fair comparison based on the models which use positive and negative sampling and experimental protocols e.g., raw settings differently (please refer to the next section).

4.5 Experiment

Our proposed model was evaluated via two tasks: link prediction[35] and triple classification [46]. This section discusses the experimental procedures for our model.

4.5.1 Datasets

In this paper, the experiments are conducted on four benchmark datasets: FB15k [35], FB15k-237 [80], YAGO3-10 [9] and FB13 [46] are respectively extracted from real

Table 4.3: Datasets used in the experiments.

Dataset	#E	#R	#T	#Train	#Valid	#Test
FB15k	14,951	1,345	571	483,142	50,000	59,071
FB15k-237	14,541	237	181	272,115	17,535	20,466
FB13	75,043	13	10	316,232	5,908	23,733
YAGO3-10	123,182	37	21	1,079,040	5,000	5,000

knowledge graphs Freebase[2] and YAGO[3]. The link prediction task has been conducted with FB15k, FB15k-237 and YAGO3-10 datasets. In FB15k dataset has redundancy entries as it also includes inverse relations. It contains 14,951 entities, 1,345 relations and 592,213 triples with 541 relation specific types of head/tail entities. On the other hand, FB15k-237 was prepared from FB15k by removing relations that were considered as inverse relations of other relations. FB15k-237 dataset includes 14,541 entities, 237 relations and total 272,115 triples. It has 141 relation specific entity types. Recently, YAGO3-10 becomes a popular benchmark dataset with very large number of triples (1,079,040 triples) comparing to other datasets. Though it contains vast number of triples but it has only 37 relations. We collected the relation specific types of head/tail entities for all the mentioned datasets. TPRC didn't exploit hierarchical types and we already mentioned earlier how hierarchical types can effect the KG embeddings.

In this paper, the "FB13" dataset has been employed to evaluate the triple classification task. These datasets contain negative triples, which is helpful for this particular task. Moreover, it has only 13 relations and we also collected the relation specific type information for this dataset. Table 4.3 shows the statistics for the datasets used in this paper, where #E/#R/#T/#Train/#Valid/#Test denotes the number of entities/relations/ entity types/training triples/validation triples and test triples.

4.5.2 Link Prediction

We follow [35] and formalize link prediction task as a point-wise learning to rank problem, where the objective is learning a scoring function $f_r : E \times R \times E \rightarrow \mathbb{R}$. For a

[2]https://developers.google.com/freebase/

[3]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

triple (h, r, t), the link-prediction task predicts the missing h or t, given the relation and the other entity. The results were evaluated by ranking the predicted head or tail entity, as calculated by the scoring function $f_r(h, t)$ for test triples.

Experimental Protocol: For our experiments, we adopted the same protocol proposed by Bordes et al.[35]. For each testing triple (h, r, t), we corrupted it by replacing the tail t or head h with every entity e in the KG or the current dictionary and calculated a probabilistic score for the corrupted triple (h, r, e) or (e, r, t), respectively, in terms of the scoring function $f_r(h, e)$. It's the "Raw" setting protocol. Because we have corrupted the triples randomly, this same triple may already exist in the actual KG and would be considered correct. During the ranking, it is logically possible that such triples may appear before the original triple. To eliminate this issue, we intentionally remove those corrupt triples that are created by replacing h or t randomly but that already exist in the KG before computing the rank of each testing triple. They may exist in any of the training, valid, or testing sets. This revised setting protocol is called the"Filter" setting. In addition, we employ the same two sampling methods, "bern" and "unif," that were used in the previous studies.

Three evaluation metrics were used: (1) Mean Rank (MR, the mean of all predicted ranks); (2) Mean Reciprocal Rank (MRR, the mean of all the reciprocals of predicted ranks); and (3) Hits@10 (the proportion of testing triples whose rank did not exceed those of the top 10 predictions). For both settings, a lower MR and, higher MRR and Hits@10 imply a better performance.

We have conducted three experiments for link prediction to better understand the performance of our model and also included the most recent advancement in this field. In the first experiment, we exploited marginal loss function along with sampling methods unif and bern. In the second experiment, we tried to focus on the most recent models. Here, we employed log-sigmoid loss function using SNS sampling method. In the last experiment, we tried to focus on how type effects while head and tail entity have the same types or different types. We showed results based on marginal loss.

Baselines

Though the Goal of TPRC is to increase the effectiveness of translation distance based models but we address the results of the state-of-the-art- bilinear models. To demonstrate the effectiveness of our models, we compare results with the following baselines.

Translation-Distance Based Models: They are as follows:

SE [34]: Bordes proposed the SE model[34], which introduces two different matrices to project separately the head and tail entities for each relation.
UM [33]: UM ignores relations, its scoring function is a simplification of that used in TransE.
TransE [35]: It learns embedding as $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ where (h, r, t) holds.
TransR [36]:This model considers separate spaces for entities and relations, but the main principle is that entities and relations are completely different types of objects, implying that they should not occupy the same vector space.
TransD [3]: It can be considered as a special case of TransR. It replaces transfer matrix by the product of two projection vectors of an entity and relation pair. .
TorusE [38]: This model addressed regularization problem of TransE.
RotatE [39]: It is able to model and infer various relation patterns including: symmetry/antisymmetry, inversion, and composition.
HyperKG [40]: It exploits the hyperbolic space in order to better reflect the topological properties of knowledge bases.

Bilinear Models: Here, we considered the most popular bilinear models.

RESCAL [41]: It is a bilinear model, with each relation being represented by an *n-by-n* matrix in an embedding space $\mathbb{R}^n$ and the scores for the triples being calculated by a bilinear mapping.
DistMult [43]:DistMult simplifies RESCAL by restricting the matrices to diagonal matrices.

ComplEx [4]:It uses complex numbers instead of real numbers and takes the conjugate of the embedding of the tail entity before calculating the bilinear mapping.
TuckER [44]: In TuckER for any given triple, it assumes that there exists an assignment of values to the entity and relation embeddings that accurately separates the true triples from false ones.

Neural Network Based Models: Recent, Neural Network based models are listed here.

ConvE [77]: It proposed a multi-layer convolutional network model ConvE which is very efficient in terms of time and space complexity compare to other NN based models proposed earlier.
ConvR [6]: It enabled rich interactions between entities and relation representations and achieved the state-of-the-art performance in link prediction task.
InteractE [49]: InteractE targets the limitations of convE. This model analyzes how increasing the number of these interactions affects link prediction performance, and utilize the observations.

Type-constraints Based Models: Type-constraints are the main focus of this study.

TKRL [5]: It considers hierarchical entity categories.
SSE [50]: It incorporates the category information into KG embedding. It employs two manifold learning algorithm for representation learning based on semantic smoothness assumption.

Optimization and Implementation: We conducted a grid search to tune suitable parameters of TransE, TransR, TransD, $\text{TPRC}_{\text{TransE}}$, $\text{TPRC}_{\text{TransR}}$ and $\text{TPRC}_{\text{TransD}}$. For bilinear models (DistMult and ComplEx) we followed the instructions in their original paper. We have used the FB15k and FB15k-237 datasets to compare our models with the baseline models. To obtain the best settings for our models, we fine-tuned five parameters. We selected the margin γ from the set {0.5, 1, 2}, the dimensionality of entity and relation vectors from {50,100, 200, 500, 1000}, the learning rate α from {0.01,0.05, 001,0.005, 0.0001,0.0005}, the number of training triples in each mini-batch

Table 4.4: Link prediction results on FB15k and FB15k-237 (raw and filter settings using MR, MRR and Hits@10). The results of RESCAL, UM and SE were reported by [3]. MRR (filter setting) and Hits@10 (filter setting) of DistMult and ComplEx were copied from [4]. MR and Hits@10 for FB15k of TKRL [5] were copied from and for FB15k-237 (TKRL) the code has been taken from https://github.com/thunlp/TKRL. We implemented the extended models of TransE, TransR and TransD. The code of TransE, TransR, TransD, DistMult and ComplEx are taken from https://github.com/thunlp/TensorFlow-TransX and https://github.com/ttrouill/complex. The sign " - " means result is not available in the corresponding paper for that particular matric or dataset.

Dataset	FB15k						FB15k-237					
Metric	MR		MRR		Hits@10		MR		MRR		Hits@10	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
RESCAL	828	683	-	-	0.284	0.441	-	-	-	-	-	-
DistMult	164	95	0.242	0.654	0.401	0.818	391	251	0.133	0.244	0.262	0.419
ComplEx	212	98	0.242	0.692	0.534	0.840	490	339	0.139	0.247	0.248	0.428
UM	1,074	979	-	-	0.045	0.063	-	-	-	-	-	-
SE	273	162	-	-	0.288	0.398	-	-	-	-	-	-
TKRL(RHE)	184	68	0.242	0.410	0.492	0.694	512	232	0.139	0.240	0.283	0.416
TKRL(WHE)	186	68	0.244	0.417	0.492	0.696	523	228	0.135	0.236	0.287	0.419
TKRL(RHE+STC)	202	89	0.267	0.459	0.504	0.731	531	238	0.137	0.249	0.293	0.426
TKRL(WHE+STC)	202	87	0.280	0.464	0.503	0.734	519	236	0.141	0.251	0.291	0.423
TransE (unif)	297	99	0.181	0.353	0.402	0.518	614	401	0.994	0.176	0.244	0.344
TransE(bern)	256	91	0.231	0.386	0.424	0.621	587	375	0.122	0.207	0.267	0.378
TransR (unif)	226	82	0.205	0.388	0.438	0.657	54	387	0.091	0.211	0.270	0.419
TransR(bern)	198	78	0.242	0.408	0.487	0.688	510	401	0.132	0.247	0.273	0.420
TransD (unif)	240	77	0.251	0.462	0.491	0.744	509	360	0.142	0.235	0.275	0.406
TransD(bern)	210	90	0.456	0.658	0.546	0.781	545	396	0.147	0.252	0.289	0.443
$TPRC_{TransE}$(unif)	223	92	0.241	0.403	0.433	0.568	599	356	0.124	0.199	0.288	0.384
$TPRC_{TransE}$(bern)	198	87	0.241	0.398	0.464	0.642	568	370	0.129	0.215	0.299	0.378
$TPRC_{TransR}$(unif)	222	79	0.243	0.411	0.488	0.667	510	360	0.121	0.242	0.301	0.419
$TPRC_{TransR}$(bern)	166	83	0.272	0.448	0.507	0.690	499	321	0.141	0.251	0.302	0.428
$TPRC_{TransD}$(unif)	141	**61**	0.355	0.599	0.613	0.802	480	256	0.149	0.255	0.312	0.437
$TPRC_{TransD}$(bern)	**102**	81	**0.506**	**0.701**	**0.644**	**0.846**	**387**	**157**	**0.168**	**0.286**	**0.322**	**0.468**

Table 4.5: Link prediction results on YAGO3-10 (filter setting). We implemented the extended models of TransE, TransR and TransD. The code of TransE, TransR, TransD, DistMult and ComplEx are taken from https://github.com/thunlp/TensorFlow-TransX and https://github.com/ttrouill/complex

Models	MR	MRR	Hits@10
RESCAL	-	-	-
DistMult	6051	0.351	0.563
ComplEx	6134	0.380	**0.592**
UM	-	-	-
SE	-	-	-
TransE	6499	0.211	0.435
TransR	5926	0.224	0.471
TransD	4803	0.361	0.558
$\text{TPRC}_{\text{TransE}}$	6400	0.241	0.476
$\text{TPRC}_{\text{TransR}}$	5115	0.229	0.482
$\text{TPRC}_{\text{TransD}}$	**4386**	**0.382**	**0.592**

from {20,50, 200, 300,1440, 2000,4000,4800, 5000}, and the dissimilarity measure in the embedding scoring function from $\{L_1, L_2\}$. The parameters of $\text{TPRC}_{\text{TransE}}$, $\text{TPRC}_{\text{TransR}}$ and $\text{TPRC}_{\text{TransD}}$ are initialized as described in [35], [36] and [3] respectively.

The optimal configurations for $\text{TPRC}_{\text{TransE}}$ were $\gamma = 1$, $d = 50$, $\alpha = 0.0001$, $B = 50$, and using L_1 as the dissimilarity function for FB15k dataset; $\gamma = 1$, $d = 50$, $\alpha = 0.0001$, $B = 50$, and using L_1 as the dissimilarity function for FB15k-237 dataset; $\gamma = 1$, $d = 150$, $\alpha = 0.0005$, $B = 100$, and using L_1 as the dissimilarity function for YAGO3-10 dataset.

The optimal configurations for $\text{TPRC}_{\text{TransR}}$ were $\gamma = 1$, $d = 100$, $\alpha = 0.05$, $B = 4800$, and using L_1 as the dissimilarity function for FB15k dataset; $\gamma = 1$, $d = 100$, $\alpha = 0.01$, $B = 1440$, and using L_1 as the dissimilarity function for FB15k-237 dataset; $\gamma = 1$, $d = 200$, $\alpha = 0.05$, $B = 1440$, and using L_1 as the dissimilarity function for YAGO3-10 dataset.

The optimal configurations for $\text{TPRC}_{\text{TransD}}$ were $\gamma = 1$, $d = 100$, $\alpha = 0.05$, $B = 1440$, and using L_2 as the dissimilarity function for FB15k dataset; $\gamma = 1$, $d = 100$, $\alpha = 0.01$, $B = 200$, and using L_2 as the dissimilarity function for FB15k-237 dataset; $\gamma = 1$, $d = 200$,

Table 4.6: Link prediction results on FB15k and FB15k-237 (filter settings only using MR, MRR and Hits@k where k=1,3 and 10), comparison with the recent state-of-the-art models. The results of ConvE and ConvR were reported by [6]. The results of the other models in this table are directly copied from the respected papers.

Model	FB15K					FB15k-237				
			Hits					Hits		
	MR	MRR	@1	@3	@10	MR	MRR	@1	@3	@10
ConvE	64	0.745	0.558	0.723	0.873	246	0.316	.237	.356	0.491
ConvR	-	0.782	0.720	0.826	0.887	-	0.350	0.261	0.385	0.528
HyperKG	-	-	-	-	-	-	0.28	-	-	0. 450
ComplEx		0.692	0.599	0.759	0.840		0.247	0.158	0.275	0.428
InteractE	-	-	-	-	-	172	0.354	0.263	-	0.535
TuckER	-	**0.795**	0.741	**0.833**	**0.892**	-	**0.358**	**0.266**	**0.394**	**0.544**
TorusE	-	0.733	0.674	0.771	0.832	-	0.307	0.219	0.337	0.485
RotatE	**40**	0.797	**0.746**	0.830	0.884	177	0.338	0.241	0.375	0.533
TransE(bern)	91	0.404	0.251	0.493	0.621	375	0.207	0.125	0.227	0.377
TransR (bern)	78	0.408	0.302	0.526	0.688	401	0.247	0.137	0.241	0.420
TransD (bern)	90	0.658	0.324	0.586	0.781	256	0.286	0.179	0.291	0.453
$\text{TPRC}_{\text{TransE}}$(bern)	87	0.398	0.291	0.520	0.642	370	0.215	0.146	0.251	0.378
$\text{TPRC}_{\text{TransR}}$(bern)	83	0.448	0.322	0.537	0.690	321	0.251	0.142	0.270	0.428
$\text{TPRC}_{\text{TransD}}$(bern)	61	0.701	0.401	0.668	0.846	**157**	0.286	0.215	0.347	0.468
$\text{TPRC}_{\text{TransD}}$(SNS)	**40**	0.735	0.499	0.783	0.858	161	0.311	0.236	0.362	0.502

$\alpha = 0.01$, $B = 300$, and using L_2 as the dissimilarity function for YAGO3-10 dataset. For the $\text{TPRC}_{\text{TransD}}$ using Log-sigmoid loss with self-adversarial negative sampling α is chosen from {0.5, 1.0}, and fixed margin γ is chosen from {1, 3, 6, 9, 12, 18, 24, 30}.

We also had to find optimal parameter settings for original TransE, TransR and TransD. We achieved better results than the original paper by tuning the parameters for these three models. For TransE, they were $\gamma = 1$, $d = 50$, $\alpha = 0.0001$, $B = 50$, and using L_1 as the dissimilarity function for FB15k and FB15k-237 and $\gamma = 1$, $d = 100$, $\alpha = 0.0001$, $B = 100$, and using L_1 as the dissimilarity function for YAGO3-10 dataset. For TransR, they were $\gamma = 1$, $d = 50$, $k = 50$ (dimensionality for relation vectors), $\alpha = 0.001$, $B = 1440$, and using L_1 as the dissimilarity function for FB15k dataset; $\gamma = 1$, $d = 100$, $\alpha = 0.0001$, $B = 1440$, and using L_1 as the dissimilarity function for FB15k-237

Table 4.7: Relation-wise (head and tail have the different types) analysis on MRR

Relations	ComplEx	TransD	$\text{TPRC}_{\text{TransD}}$(bern)
profession	0.801	0.799	**0.817**
written_by	**0.637**	0.636	0.636
film_in_this_genre	0.739	0.734	**0.751**
olympics_participated_in	0.699	0.700	**0.719**
directed_by	0.681	0.679	**0.684**
films_production_designed	**0.574**	0.571	0.572
place_of_birth	0.721	0.714	**0.730**

Table 4.8: Relation-wise (head and tail have the same type) analysis on MRR.

Relations	ComplEx	TransD	$\text{TPRC}_{\text{TransD}}$(bern)
includes_diseases	**1.000**	**1.000**	**1.000**
administrative_parent	**0.697**	0.600	0.614
sibling	**0.559**	0.557	**0.559**
spouse	**0.624**	5.436	5.433
is_part_of	**0.723**	**0.723**	0.702
award_nominee	0.758	0.742	**0.772**
children	**0.666**	0.615	0.647
parents	**0.453**	0.441	0.422

dataset; $\gamma = 1$, $d = 200$, $\alpha = 0.0005$, $B = 1440$, and using L_1 as the dissimilarity function for YAGO3-10 dataset. The optimal configurations for TransD for FB15k and FB15k-237 datasets were: $\gamma = 1$, $d = 100$, $\alpha = 0.0001$, $B = 300$, and using L_2 as the dissimilarity function; for YAGO3-10 dataset configurations were $\gamma = 1$, $d = 200$, $\alpha = 0.0005$, $B = 300$, and using L_1 as the dissimilarity function.

Result Analysis: As we mentioned earlier we have conducted three experiments. Here, we analyze the results of three experiments as follows:

Experiment 1: In this thesis, we have applied the idea of TPRC to the translation distanced based model only. In Translation distance based models it is very common practice to was raw and filter settings along MR, MRR and Hits@10 metric. As we

mentioned earlier the marginal loss function maintain same ratio for positive and negative triples. Accuracies are reported in the Table 4.4 and Table 4.5 using marginal loss function only. Results in bold font are the best obtained results. In Table 4.5 we reported the filter setting results of YAGO3-10 dataset of our models and other models as well. Here, we have employed "bern" for all the translation distanced based models. From Table 4.4 and 4.5 we have the following findings:

- Our models outperformed all other methods, using the experimental datasets FB15k and FB15k-237 with both Raw and Filter settings on all metrics. $\text{TPRC}_{\text{TransD}}$with "bern" achieves the best results on MR, MRR and Hits@10 in the both experimental datasets except the MR value (filter setting) of FB15k of $\text{TPRC}_{\text{TransD}}$ with "unif" achieves the best performance. The closest competitor is ComplEx model. DistMust also showed very promising results. Though ComplEx was slightly better than TransD but exploiting type information in relational context helps the performance of $\text{TPRC}_{\text{TransD}}$. In YAGO3-10 dataset ComplEX and $\text{TPRC}_{\text{TransD}}$ achieved the best performance for hits@10 and $\text{TPRC}_{\text{TransD}}$ outperformed all other methods on other metrics. Another interesting observation is no model achieved very significant performance compare to other state-of-the-art models in YAGO3-10 dataset. The test samples of this dataset is 5,000 only, which is very small compare to FB15k and FB15k-237 datasets. We also observe that prior information of head or tail entities' types contribute in the prediction face on our other models;

- The results appear to show that the "bern" (Table 4.4) sampling method performs slightly better than the "unif" method for the translation distance based models;

- All the translation distance based models showed good performance for the dataset FB15k-237. For this dataset $\text{TPRC}_{\text{TransD}}$(bern) significantly outperform DistMult, ComplEx and other models on MR, MRR and Hits@10. Here we used the metric Hits@10 only because we extended the TPRC with the translation distance based models. Nowadays Hits@1 and Hits@3 metrics are popular among the semantic matching and NN based models. As we mentioned earlier FB15k-237 has no inverse relations, so it shows that translation distance based models can work well with such datasets than bilinear models. Another interesting

observation is the results of raw setting of translation distance based models is higher than the bilinear models;

- From all the results, based on the good basic model TransD, the extended models of TransD can achieve the best performance compared with other state-of-the-art baselines TKRL, ComplEx, DistMult, TransE, TransR and TransD itself. Despite of very high time and space complexity, TKRL achieved very good results in FB15k and FB15k-237 datasets but it has failed to outperform ComplEx, DistMult, TransE, TransR and TransD. TKRL exploited very rich type information in their model. The advantage of TPRC over TKRL is: TPRC can learn KG embedding jointly with other state-of-the-art translation distanced based models with less parameters overhead. $\text{TPRC}_{\text{TransD}}$(bern) achieved 0.846, 0.468 and 0.592 of Hits@10 for the datasets FB15k, FB15k-237 and YAGO3-10 respectively, which are 8.3%, 5.6% and 6.1% higher than the original TransD (bern). It also obtains very impressive results over ComplEx on FB15k-237 dataset.

Experiment 2: Link prediction in the KG is one of most competitive fields of machine learning right at the moment. Everyday new models are being proposed. To keep with the pace and show the recent advancement, we present the most recent contributions in this field in the Table 4.6. Please note that $\text{TPRC}_{\text{TransD}}$ shows better result with TransD in Table 4.4 but it has issues dealing with the negative samplings. Here, we applied Log-sigmoid Loss Function with the SNS sampling method (on TPRC only) and show how $\text{TPRC}_{\text{TransD}}$ improves the performace in the Hits@k metric. $\text{TPRC}_{\text{TransD}}$ (SNS) really did well in the MR metric but in other cases most recent and powerful model TuckER performed better than any other model. Moreover, analyze the performance of these models elaborately here (Table 4.6):

- On FB15K dataset, the main relation patterns are symmetry/antisymmetry and inversion. We can see that bilinear models and convolutional models perform well while translational models (TransE, TransR and TransD) do not perform well specially in Hits@1 and Hits@3 metrics. But results of the translational models in the MR and MRR metrics are quite promising though RotatE showed good performance in all metrics. After applying the TPRC to TransE, TransR and TransD the performances improved tremendously specially TPRC (SNS)

achieved very competitive results. The reason is that for most of the relations in FB15K, the types of head entities and tail entities are different and TPRC works well when the types of head and tail entities are different. ComplEx can infer both symmetry/antisymmetry and inversion patterns while TransE cannot infer symmetry pattern. Surprisingly, DistMult achieves good performance on this dataset although it cannot model the antisymmetry and inversion patterns. It also same for ConvE and ConvR. But TuckER did achieve splendid performance.

- On FB15k-237, the main relation pattern is composition. We can see that translational models perform really well while other could not show their same performance as it was in FB15k. The reason is that, as discussed before, translational models are able to infer the composition pattern while bilinear models cannot infer the composition pattern. So TPRC was strong in this case as it depends on the performance of translational models. Though bilinear model TuckER is an exception.
- Other very interesting observations are: Bilinear and Neural Network based models work very well for Hits@1 but they are not good in MR metric. In the other hand, all translational models are very good in MR metric. That means bilinear and neural network based models focused on the training data quite seriously but not in generalization. From our point of view, for totally new data these models may not work well and it is obvious that bilinear and neural network based models are very vulnerable to over fitting problem compare to translational models.

Experiment 3: Table 4.7 and 4.8 show the relation-wise analysis for FB15k dataset for few relations. In our datasets some relations comprise with same type of head and tail entities and some have different types in the head and tail entity. Our model achieves a bit better results for the relations which have different types of head and tail entities than the other type relation. We observed that same-type entities tend to converge and form clusters, which can lead to errors in some cases. In the near future, we aim to develop a data-sampling algorithm to address this problem.

The calculation time of TPRC models are shown in Fig. 4.1. They are measured by using four cores in a 3.90 GHz processor. Theoretically, the time complexities of

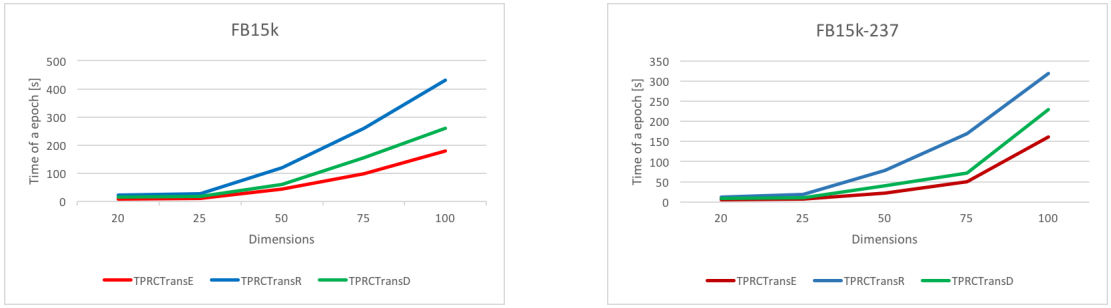

Figure 4.1: Calculation time of TPRC models on FB15k and FB15k-237 datasets.

$TPRC_{TransE}$, $TPRC_{TransR}$ and $TPRC_{TransD}$ are same. $TPRC_{TransE}$model took less amount of time than other two models. For FB15k dataset, $TPRC_{TransE}$, $TPRC_{TransR}$ and $TPRC_{TransD}$ took 179.7, 431.1 and 260.6 seconds respectively to complete one epoch when the dimension was 100. On the other hand, for FB15k-237 dataset, $TPRC_{TransE}$, $TPRC_{TransR}$ and $TPRC_{TransD}$ took 161.2, 321.8 and 230.2 seconds respectively to complete one epoch with same dimension. These models were trained for 1000 epochs for each dataset.

As noted, the entities' type plays a crucial role with respect to its relations. It is therefore logical that incorporating type information in relational context could be utilized to achieve better performance in the link-prediction task. We believe that the projection of entities based on the entities' type-mapping matrix would improve the performance of the proposed models. In these models, we are using entities' type information considering the relation it appers, in addition to the entities themselves, enabling the projected vectors to exhibit more semantic information than the vectors in TransE, TransR and TransD models.

4.5.3 Triple Classification

The triple classification task [46] checks whether a given triple (h, r, t) is correct or incorrect. When it was first introduced by Socher et al. in the NTN model, it acted as a

binary classification. Because the task requires negative samples, we employed the "FB13" dataset, which is a benchmark dataset from Freebase involving 13 relations.

Experimental Protocol: The experimental setup for triple classification task is very simple. To implement this task, we set a relation-specific threshold σ_r. For a triple (h, r, t), if the dissimilarity score (computed by the scoring function f_r) is below the σ_r threshold, then the predicted triple is positive. Otherwise, the prediction is negative. The value for σ_r is determined in accordance with the classification accuracy.

Table 4.9: Accuracy of the triple classification task on FB13 dataset. The results of SLM, NTN, SE, TransE, TransR and TransD were reported by Ji et al. [3].

Datasets	Accuracy (%)
SLM	85.3
NTN	87.1
SE	75.2
TransE(unif)	70.9
TransE(bern)	81.5
TransR(unif)	74.7
TransR(bern)	82.5
TransD(unif)	85.9
TransD(bern)	89.1
$\text{TPRC}_{\text{TransE}}$(unif)	75.3
$\text{TPRC}_{\text{TransE}}$(bern)	85.0
$\text{TPRC}_{\text{TransR}}$(unif)	81.6
$\text{TPRC}_{\text{TransR}}$(bern)	88.8
$\text{TPRC}_{\text{TransD}}$(unif)	87.6
$\text{TPRC}_{\text{TransD}}$(bern)	**89.9**

Optimization and Results: Table 4.9 shows the evaluation results for triple classification. The parameters and evaluation results for SLM, NTN, SE, TransE, TransR, TransD were obtained directly from the paper [3]. The parameter values for training $\text{TPRC}_{\text{TransE}}$, $\text{TPRC}_{\text{TransR}}$ and $\text{TPRC}_{\text{TransD}}$ are shown in Table 8.

We see from the Table 7 that $\text{TPRC}_{\text{TransD}}$ (bern) achieved the best performance. $\text{TPRC}_{\text{TransE}}$ is more accurate than TransE, $\text{TPRC}_{\text{TransR}}$is more accurate than TransR

Table 4.10: Parameter settings of FB13.

Models	α	B	γ	d	$D.S$
$TPRC_{TransE}$	0.001	50	1	100	L1
$TPRC_{TransR}$	0.001	300	1	100	L1
$TPRC_{TransD}$	0.0005	1440	2	100	L2

and $TPRC_{TransR}$ outperform TransD. These results imply that incorporating type information in the relation context can improve the model accuracy. Papers of the bilinear/semantic matching models (RESCAL, DistMult and ComplEx) usually do not include triple classification task. So, we report the results of translational models and neural network based models.

4.6 Conclusion

In this paper, we have proposed an embedding model that leverages entity-type properties in the relational context. The strengths of this model are: it can produce find-grained representation of entities considering the entity types in relational context and it can be combined easily with other translation-distance-based models to improve accuracy without making the models more complex. The TPRC model is conceptually simple and can demonstrate highly competitive results for link prediction and triple classification. The underlying idea of TPRC was applied to the most popular translation-distance-based models (TransE, TransR and TransD) with the experimental results showing better performances than for the basic TransE ,TransR, and TransD models and shows competitive performance with other state-of-the-art models as well. In the experiments, we have observed the effect of single and hierarchical types on link prediction task. Compare to those models the performance of injecting the relation specific types is better due to better representation of the entities. We can therefore conclude that the entities' type-based diversity in relations in a KG is an important factor and that the entities' type-mapping matrix in relational context is suitable for modeling KGs. In the future, we will utilize more sophisticated models to leverage entity types information. We also intend to use entity type ranking information in relational context and perform experiments on a wider variety of datasets.

5

Representation Learning for Entity Type Ranking

The type of an entity is a key piece of information to understand what an entity is and how it relates to other entities mentioned in a document. Search engine result pages (SERPs) often surface facts and entity type information from a background Knowledge Graph (KG) in response to queries that carry a semantic information need. In a KG, an entity usually holds multiple type properties. It is then important to, given an entity in a KG, rank entity types attached to the entity by relevance to a certain user and information need as not always the most popular type is the most informative within a textual context.

In this chapter we address the entity type ranking problem by means of KG embedding models. In our work, we show that entity type ranking can be seen as a special case of the KG completion problem. Embeddings can be learned from both the structural and probabilistic information of the entities. We propose a Representation Learning model for Type Ranking (RL-TRank) and the results of the

structure embedding and the probabilistic embedding are combined to get the entity type ranking. Experimental results show that the accuracy of RL-TRank approaches outperform the state-of-the-art type ranking models while, at the same time, being more efficient and scalable.

5.1 Introduction

Type information plays an important role in knowledge bases e.g, DBpedia [71], YAGO [9], Freebase [10], Wikidata [21]. It facilitates several applications such as entity linking, entity search, question answering, etc. As web search user expectations increase, the effective use of knowledge graphs to generate Search Engine Result Pages (SERPs) becomes essential. Given a user query, the generated SERP presents related information available in the knowledge graph. Entities' relevant types can be shown in the SERP to recommend more meaningful facts for entity-centric queries, which will increase user engagement [81]. Entity types carry information about the different roles of an entity [18]. As an example, in Freebase entity 'Donald Trump' has 32 types information including "Wealthy Person", "Organization founder", "Businessman", "Celebrity", "Person or entity appearing in film", "Actor", "Architectural structure owner". From those types we can understand the characteristic of the entity 'Donald Trump'. Out of all possible types attached to an entity in a knowledge graph, only few may be relevant to a user looking for information about an entity on the Web. Thus, the task of ranking entity types by their relevance to a user's information need becomes important. Entity type ranking approaches can be used to select which entity type to display given the limited space on entity cards presented to users in SERPs. We can also define the task of ranking entity types as a kind of entity summarization task. Tonon et. al. [55] proposed TRank, which is the first effort addressing the entity type ranking problem. TRank introduced different approaches to deal with the problem and used crowdsourced relevance judgment on real datasets to experimentally evaluate the proposed ranking methods.

In this work, we propose a representation learning model to tackle the entity type ranking problem by incorporating knowledge graph embedding models. In order to be directly comparable with previous work, in our experimental setup we use the same datasets used to evaluate previously proposed entity type ranking methods [55].

The advantages of our model over previous work are that it is easier to interpret as compared to learning to rank models and it is scalable at the same time.

A knowledge graph is a heterogeneous structure that stores facts about real world entities in a machine readable format, where nodes denote entities and links between entities denote relations among entities. This data is represented as a triplet, (h, r, t), where h and t are entities in real world and r is a relation between h and t. For example, consider a triple (Donald Trump, presidentOf, USA) where Donald Trump, USA are *head* and *tail* entities respectively, and *presidentOf* is the relation between these two entities.

Knowledge graphs have attracted research attention in many fields, e.g., recommendation systems [61, 59], dialogue systems [82, 83], information extraction [84, 85]. Several KG structure embedding models have been proposed over the last few years to address the knowledge graph completion task. Translation distance based models gained popularity for their simplicity and effectiveness in knowledge graph completion. Translation based models can successfully model relations between entities as translations in a vector space. The basic principle (TransE [35]) is $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$, where (h, r, t) holds ($\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}$ are embeddings of h, r, t respectively).

In this study, we introduce a "relevance" property and connect the entity and its respective types with this relevance property. So, the triples in this study are organized as (entity, relevance, entity_type). We denote the triple as $(e, relevance, tp)$, where e and tp are the entity and entity type respectively and $relevance$ connecting them is similar to the *rdfs:type* property in knowledge graphs.

To deal triples, we introduce two modules, namely Structure Embedding (SE) and Probabilistic Embedding (PE). Following the popular translation-based embedding models, we adopt three popular embedding models (TransE[35], TransH[37], and TransR[36]) for our SE module. We also evaluate those models individually. In our RL-TRank model, SE focuses on modeling relationship structures of the datasets and PE captures the correlations of entities with their corresponding types.

The main contributions of this chapter are:

1. A novel way to use representation learning model for ranking entity types.
2. The use of a "relevance" property within a novel knowledge graph construction technique to address type ranking problem.

3. An experimental evaluation showing that proposed RL-TRank models are more effective and outperform the state-of-the-art type ranking baselines.

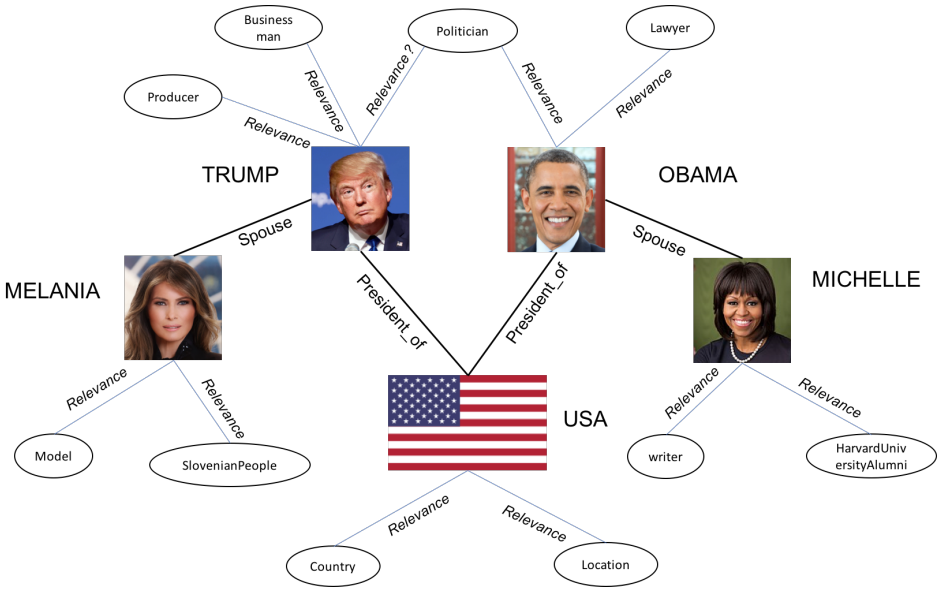

Figure 5.1: Entity type ranking based on a knowledge graph (image resources are obtained from Wikipedia).

5.2 Entity Type Ranking Approach

In this section, we provide the formal definitions and show how the entity type ranking problem can be interpreted as KG embedding models and how the ranking function for entity types can be derived from such models (Fig. 5.1). We mentioned the formal definition of KG in the chapter 2. In KG embedding we, work with the relational triples (in this case we also include type-like triple as well), so defining it based on triples only again in this chapter.

Definition 1 ***(Knowledge Graph)***

A knowledge graph is defined as a set of triples in the form (h, r, t) *where* $h, t \in E$ *and* $r \in R$*, where* E *is the set of entities and* R *is the set of relations among the entities. In our problem we assume entity,* $e \in E$ *and entity type,* $tp \in T \subset E$*,*

where T is the set of entity types. An entity type can be considered as distinct item/characteristic. Such an assumption helps us to define the type ranking problem using knowledge graph embeddings. Crowd-sourced judgments between an entity and an entity type is encoded by a special property "relevance".

Definition 2 ***(Entity Type Ranking)***

Given the pair composed by entity and entity type, we use the "relevance" property to generate a set of triples (e, relevance, tp). The task is to identify a ranking function $f(e, tp)$ that assigns a score to each entity-entity type pair and then sorts the types based on the scores computed for a specific entity.

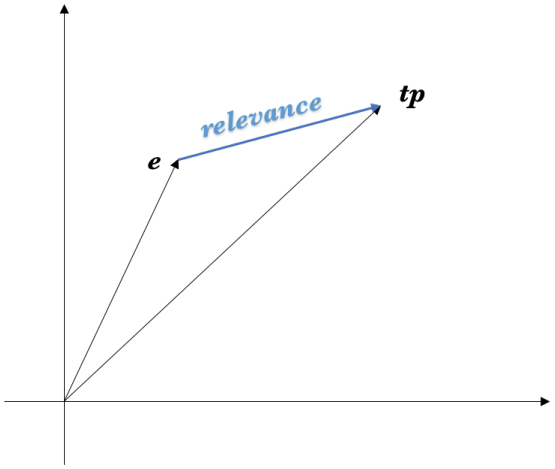

Figure 5.2: TransE: The ranking function is defined through the distance between $(e + relevance)$ and tp.

5.2.1 Structure Embedding (SE)

In KG, we interpret a relationship as the translation from the head entity to the tail entity, to characterize the structure information [35]. Knowledge graph embedding models map entities and relations in a KG to a vector space and predict unknown triples by scoring candidate triples. The translation distance based models have gathered attention because of their efficiency and simplicity among other KG embedding models. In SE embedding model, we employ state-of-the-art translation distance based models to rank the types as follows:

TransE [35] learns embedding as $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ where (h, r, t) holds. Hence, $(\boldsymbol{h} + \boldsymbol{r})$ is very close to $\boldsymbol{t}$ (see Fig. 5.2). The score function of TransE is:

$$f(\boldsymbol{h}, \boldsymbol{t}) = -|| \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} ||_{l_{1/2}} \tag{5.1}$$

which is high if (h, r, t) holds, and low otherwise.

For the entity type ranking problem, we define the score function as:

$$f_{relevance}(\boldsymbol{e}, \boldsymbol{tp}) = -|| \boldsymbol{e} + \boldsymbol{relevance} - \boldsymbol{tp} ||_{l_{1/2}} \tag{5.2}$$

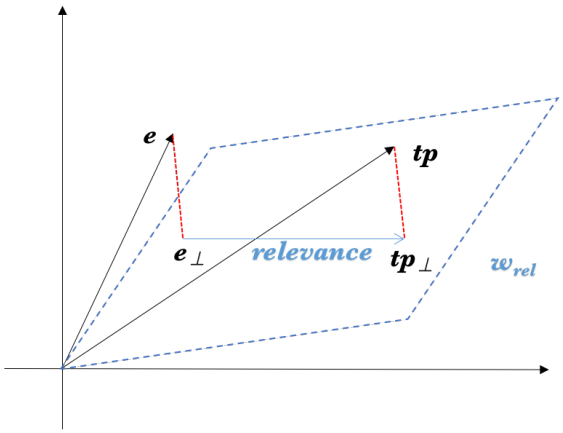

Figure 5.3: TransH: The ranking function is defined through the distance between $(e_\perp + relevance)$ and $tp_\perp$.

TransH represents the relations by hyperplanes in the knowledge graph embedding. This model projects entities on the hyperplane corresponding to a relation. One entity has different representations on different hyperplanes. TransH models the relation r as $\boldsymbol{r}$ on a hyperplane with the normal vector $\boldsymbol{w}_r$. Given a triple (h, r, t) the entity representations $\boldsymbol{h}$ and $\boldsymbol{t}$ are projected on the hyperplane of $\boldsymbol{w}_r$ by the restriction $|| \boldsymbol{w}_r ||= 1$ (see Fig. 5.3). Here, at the same time we see that the entity and the entity type representations $\boldsymbol{e}$ and $\boldsymbol{tp}$ are projected on the hyperplane of $\boldsymbol{w}_{rel}$ and that the calculation can be expressed as follows:

$$\begin{aligned} \boldsymbol{e}_\perp &= \boldsymbol{e} - \boldsymbol{w}_{rel}{}^\top \boldsymbol{e} \boldsymbol{w}_{rel}, \\ \boldsymbol{tp}_\perp &= \boldsymbol{tp} - \boldsymbol{w}_{rel}{}^\top \boldsymbol{tp} \boldsymbol{w}_{rel} \end{aligned} \tag{5.3}$$

The score function is very similar to that of TransE:

$$f_{relevance}(\boldsymbol{e}, \boldsymbol{tp}) = -|| \boldsymbol{e}_{\perp} + \boldsymbol{relevance} - \boldsymbol{tp}_{\perp} ||_{l_{1/2}} \tag{5.4}$$

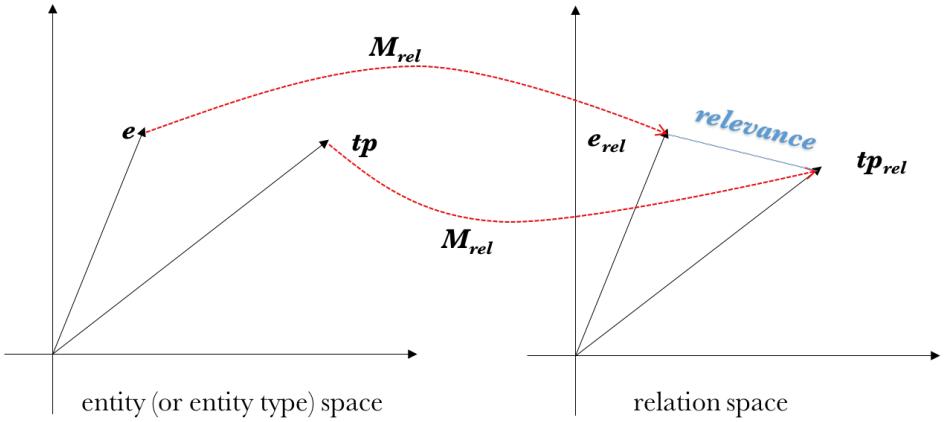

Figure 5.4: TransR: The ranking function is defined through the distance between $(e_{rel} + relevance)$ and tp_{rel}.

TransR [36] also addressed the flaws of TransE, in a slightly different way than TransH. TransR considers separate embedding spaces for entities and relations. The assumption is that entities and relations are completely different types of objects, so they should not be in the same vector space. Given a triple (h, r, t), TransR projects the entity representations $\boldsymbol{h}$ and $\boldsymbol{t}$ into the space specific to a relation r (see Fig. 5.4). Here we consider only the relation "relevance" as compared to other knowledge graph embedding models. The projection matrix, $\boldsymbol{M}_{rel}$ projects the entity and entity types to the relevance space, i.e.,

$$\boldsymbol{e}_{rel} = \boldsymbol{e}\boldsymbol{M}_{rel}, \quad \boldsymbol{tp}_{rel} = \boldsymbol{tp}\boldsymbol{M}_{rel} \tag{5.5}$$

where $\boldsymbol{e}, \boldsymbol{tp} \in \mathbb{R}^n$, $\boldsymbol{relevance} \in \mathbb{R}^n$, and $\boldsymbol{M}_{rel} \in \mathbb{R}^{n \times n}$ is the projection matrix from the entity space to the relevance space. The score function is as follows:

$$f_{relevance}(\boldsymbol{e}, \boldsymbol{tp}) = -|| \boldsymbol{e}_{rel} + \boldsymbol{relevance} - \boldsymbol{tp}_{rel} ||_{l_{1/2}} \tag{5.6}$$

In this study, we follow knowledge graph embeddings to model the entity type ranking task. According to the Open World Assumption (OWA) knowledge graphs contain only correct facts and non-observed facts can be either incorrect or missing from the knowledge graph. As for training purposes negative triples are also required,

we prepare negative triples for the model training as follows. We define a margin-based loss function as objective similar to knowledge graph embedding models for training [35]. We show how the loss function based on the *relevance* property is the same for all the relations in the set of R.

$$L_{SE} = \sum_{(e,relevance,tp)\in S} \sum_{(e',relevance,tp')\in S'} max\big(0, f_{relevance}(\boldsymbol{e},\boldsymbol{tp}) + \gamma - f_{relevance}(\boldsymbol{e'},\boldsymbol{tp'})\big) \quad (5.7)$$

$f_{relevance}(e, tp)$ is the energy function score of the positive triple and $f_{relevance}(e', tp')$ is that of the negative triple. γ is the margin, S is the set of positive triples and $S^{'}$ is the set of negative triples. Existing knowledge graphs only contain correct triples [35].

$$S' = \{(e', relevance, tp) | e' \in E \wedge e' \neq e \wedge (e, relevance, tp) \in S\}$$
$$\cup \{(e, relevance, tp') | tp' \in T \wedge tp' \neq tp \wedge (e, relevance, tp) \in S\}$$

As the equation suggests, we randomly replace the heads and tails of positive triples with other entities in E and entity types in T, respectively. Moreover, the new triples generated after such replacement will not be considered as negative triples if they already exist in S. In the training phase, the ratio of the positive and the negative triples is same.

5.2.2 Probabilistic Embedding (PE)

In PE model, we take the advantage of the probabilistic modeling [86, 85] of KGs but we keep the score function f simple. From the SE models we learn that the simplest way to measure the plausibility of a triple $(e, relevance, tp)$, by defining the score function, $f(\boldsymbol{e},\boldsymbol{tp}) = -||\ \boldsymbol{e} + \boldsymbol{relevance} - \boldsymbol{tp}\ ||_{l_{1/2}}$.

PE model defines the conditional probability on triples as follows:

$$Pr(tp|e, relevance) = \frac{exp\{f_{relevance}(\boldsymbol{e},\boldsymbol{tp})\}}{\sum_{\bar{tp}\in T} exp\{f_{relevance}(\boldsymbol{e},\bar{\boldsymbol{tp}})\}} \quad (5.8)$$

Eq. (5.8) states the conditional probability of an entity type tp where the relevance relation and an entity are given over a fact/triple. In the same way, we can define $Pr(e|relevance, tp)$ and $Pr(relevance|e, tp)$. So the objective function is given as:

$$L_{PE} = -\sum_{(e,relevance,tp)\in S} \big(\log Pr(e|relevance,tp) + \log Pr(relevance|e,tp) + \log Pr(tp|e,relevance)\big) \quad (5.9)$$

The summation is over S which is the set of all positive facts.

5.2.3 Ranking

Considering the above two component models together, the final loss is:

$$L = L_{SE} + L_{PE} \quad (5.10)$$

L_{SE} and L_{PE} are loss functions for the structure embedding and the probabilistic embedding, respectively. We adopt stochastic gradient descent (SGD) to optimize the above loss functions. Here, L_{SE} could be TransE, TransH or TransR model.

We defined the entity type ranking problem formally in Definition 2. The task is to rank a set of n entity types according to a specific entity. The core idea is to build a knowledge graph with existing relations in the knowledge graph and add entity and entity type pairs with a "relevenace" property to the knowledge graph together with other existing relations (see Fig. 5.1). Once this is done, the model learns the knowledge graph embedding, which includes all the triples as well as the "relevance" property triples. This model is evaluated over queries of the form $(e, relevance, ?)$ as a task of ranking tp based on the rank of gold entity type $tp*$. Here, we learn the vector representations of $(e, relevance, tp)$ triples along with other triples (h, r, t) and capture richer semantics for entities and their types.

5.3 Experiment

To evaluate the validity of the RL-TRank approaches, we use the datasets created by [55]. In this section, we discuss data sources and the experiments that we have conducted to assess the performance of the proposed entity type ranking models.

5.3.1 Data Collection and Annotation

First, we briefly discuss the data collection and crowd-sourced relevance judgments of our datasets. Data had been collected from the top news of each category from the New York Times (NYT) website during the period of February 21 to March 7, 2013. The retrieved entities were then linked to DBpedia and attached to corresponding types available in Linked Open Data (LOD). To collect the relevance judgments for entity and type pairs, anonymous crowd workers have been involved. Crowd workers were asked to select all types which are relevant to an entity in two ways: 1) by only looking at the entity, and 2) by judging type relevance given the textual context (e.g., a sentence, a paragraph, or collection of paragraphs) where the enitity appears. In each dataset one entity only appears once for each context. Crowd workers voted for each $(entity, type)$ pair. Type relevance has been measured based on how many workers voted for it. Pairs without any vote are considered irrelevant types. In this chapter we make use of the Entity Only (EO), Sentence Collection (SC), Paragraph (PG) and 3-Paragraphs (3PG) datasets from [55] to conduct our experimental evaluation. We construct the knowledge graph based on those datasets and evaluate RL-TRank models (see section 5.3.2).

5.3.2 Knowledge Graph Construction

We created knowledge graphs for the entity type ranking task based on the mentioned datasets. In the EO dataset we have 770 distinct entities with their associated types and relevance judgments. Entities of this dataset have been extracted from NYT articles and later linked with DBpedia. Each entity in the EO dataset has at least two types. We collected the related facts for those entities by leveraging DBpedia to create our knowledge graph. During the knowledge graph creation step we only considered those triplets that are directly connected to the entities in our datasets. One entity can have a subject or object role in the considered facts.

In order to select the most relevant properties for the knowledge graph construction, we select a subset of properties from the DBpedia ontology to create the knowledge graph by sorting them according to their frequency of occurrence for each entity. As an example, for the entity 'Tom Hanks', dbo:starring appears 51 times in DBpedia. We set a threshold of frequency to select the properties and included them in the

Table 5.1: Statistics of the datasets.

Dataset	#Entities	#Relations	#Triples
EO	21,655	336	360,366
SC	24,528	401	411,568
PG	28,772	162	378,227
3PG	14,258	157	270,348

knowledge graph. We add all the dbo:properties where frequency is more than 2. For example, for Tom Hanks we obtain: dbo:starring (51), dbo:producer (23), dbo: executiveProducer (6), dbo: director (3), dbo:writer (3), dbo: narrator (3), dbo:spouse (2), dbo: birthDate (2), dbo:portrayer (2). We collect facts for each mapped entity in this way and then preprocess the dataset by filtering out low frequency entities (i.e., frequency lower than 10) and relations (i.e., frequency lower than 10).

We finally include the "relevance" property, considering the crowd relevance judgment scores. In our dataset, $(entity, type)$ pairs have been judged over a four-level relevance scale: Score 3 means that pair counted as most relevant and score 0 means not relevant at all. We only include relevant types to the knowledge graph and filter out the (e, tp) pairs which have score 0 and score 1.

The Sentence Collection dataset (SC) is constructed as follows. In this collection sentences from NYT articles that had at least two entities were included. Then, crowd workers were asked to judge the relevance of the types of the entities appearing in a sentence by considering the sentence as textual context. In this dataset we have 1,108 unique entities.

In the PG dataset, entities are collected from paragraphs rather than from a sentence. Each paragraph contains at least two entities having more than two types. On the other hand, the 3PG dataset is a collection of 3 paragraphs for each entity. The purpose of construction SC, PG and 3PG were to give the crowd workers some context to decide on the best suited type for the entities. The knowledge graph construction and data preprocessing steps are the same for all the datasets, as described earlier in this section. We split the data into a training, validation, and test, containing, for each entity in each dataset, respectively 70%, 10% and 20% of the *relevance* properties.

In Table 5.1, we show the detailed statistics of the datasets after constructing the knowledge graph. In Table 5.1, we show the total number of entities (including the entity type, $tp \in T \subset E$), relations including the "relevance" property, and triples.

5.3.3 Evaluation Measures

We use Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) to assess the quality of the entity type ranking models we proposed. Average Precision is defined as

$$AP(T_e) = \frac{\sum_{tp \in T_e} relevant(tp_i) \cdot P@i}{|Relevant(tp_i)|}, \tag{5.11}$$

where T_e is the set of all types of an entity e, $Relevant(tp)$ is 1 if tp is a relevant type for the entity e and 0 otherwise, $Relevant(T_e)$ is the set of relevant types for e, and $P@i$ indicates precision at cutoff i. MAP is defined as the mean of AP over all entities in the collection.

The NDCG metric with cut-off level k, is calculated as: $NDCG@k = DCG@k/lDCG@k$, where $DCG@k = \sum_{i=0}^{k}(rel_i/log_2(i+1))$. $lDCG@k$ is the maximum attainable DCG, and rel_i is the graded relevance assigned to the result at position i. MAP considers binary relevance while NDCG can deal with non-binary relevance judgment.

5.3.4 Parameters of SE Models

In our implementation, TransE, TransH and TransR were optimized by SGD. For getting best model performance we tune five parameters. We select the margin γ from {0.5,1,2}, the embedding dimension d of vectors among {10, 20, 30, 40, 50, 75, 100}, the learning rate α from {0.0001, 0.0005 0.001, 0.005, 0.05, 0.01}, the batch size B from {20,50,100,200,300,1440,2000} and a dissimilarity measure ($D.S$) in embedding score function from {L1,L2}. The parameter values used in our experiments to obtain the best results are given at Table reftable:para51.

Table 5.2: Parameter settings for datasets.

Dataset	Model	α	B	γ	d	$D.S$
EO	TransE	0.0001	200	1	40	L1
	TransH	0.001	300	1	50	L1
	TransR	0.001	1440	1	50	L1
SC	TransE	0.0001	200	1	50	L1
	TransH	0.005	300	1	50	L1
	TransR	0.001	1440	1	50	L1
PG	TransE	0.0001	200	1	50	L1
	TransH	0.005	300	1	50	L1
	TransR	0.005	2000	2	50	L1
3PG	TransE	0.0001	200	1	40	L1
	TransH	0.005	300	1	50	L1
	TransR	0.005	2000	2	50	L1

5.3.5 Results

Since the datasets are the same as the ones used by [55], we directly copied the results as reported in literature [55]. We report the effectiveness of these models over four datasets (EO, SC, PG and 3PG). Table 5.3 indicates that RL-TRank models outperformed the baseline models along with the SE models separately. For RL-TRank model, it captures both the structures and the probabilistic information of KGs. RL-TRank with TransE and TransH obtain better results compared to RL-TRank with TransR. In the entity-only (EO) dataset RL-TRank achieves the good performance in MAP . In other datasets, RL-TRank (TransH) shows the best performance. Among previous approaches, DEC_TREE (decision trees) achieved best result in TRank. RL-TRank (TransH) achieves 10.32%, 12.42%, 12.32% and 6.17% improvement in performance (NDCG) over DEC_TREE on EO, SO, PG and 3PG datasets respectively.

RL-TRank consists of two components: SE model and PE model. We observed that SE models contribute largely to the over all performance of the RL-TRank than PE models. Some entities/types have very limited entry in training data, in such cases joint model show better result than individual models (SE/PE) in the testing phase.We have seen that SE models individually outperformed the TRank model. In ensemble

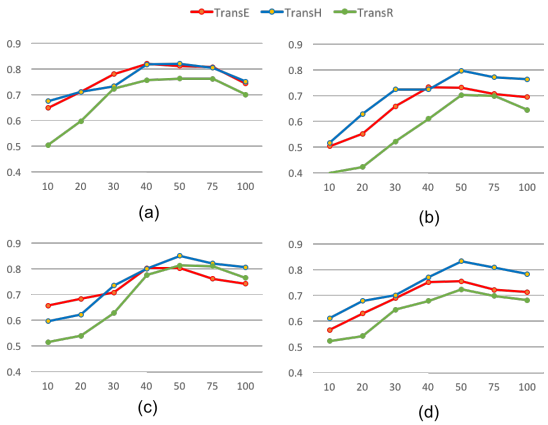

Figure 5.5: Dimension (X-axis) Vs NDCG (Y-axis) plotted in this figure for each datasets.Here, (a) Entity Only dataset (b) Sentence Collection (c) Paragraph dataset (d) 3-Paragraphs dataset

learning it is a common practice to add different model to improve the output results. By following that principle, we here add two components to compute the final results. It is possible to add more components with this type of ensemble models. In the next chapter (Chapter 6), we discuss about how to improve the ensemble learning and use other models as well to perform type-like triple scoring or entity type ranking task. For this reason we have analyzed the SE models deeply. One very interesting observation is the result of TransR among the SE models. TransR showed promising results in knowledge graph benchmark datasets compared to TransE and TransH. TransR is a more complex model than TransE and TranH. Over these type ranking datasets, TransR failed to project the entity and type relationships. In this study we focused on the "relevance" relation only. TransH obtains very good results for "relevance" relationship, but for other existing relations in our constructed KG datasets, the performance of TransR is satisfactory. One assumption that explains these results is that with more data of "relevance" relation property, TransR may show better type ranking performances.

One of the drawbacks of translation based models is the need to tune hyper parameters. In Table 5.2, we report results for the best parameter settings of the models based on the used datasets.

Table 5.3: Comparison of translation based models and baseline models on the four benchmark datasets. The code of TransE, TransH, TransR are taken from https://github.com/thunlp/TensorFlow-TransX.

Approach	EO		SC		PG		3PG	
	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP
FREQ	0.6284	0.4659	0.5409	0.3758	0.5315	0.3739	0.5250	0.3577
WIKILINK-OUT	0.6874	0.5406	0.6050	0.4521	0.6063	0.4550	0.6059	0.4444
WIKILINK-IN	0.6832	0.5342	0.5907	0.4213	0.5879	0.4254	0.5853	0.4143
SAMEAS	0.6848	0.5328	0.6049	0.4310	0.5990	0.4221	0.6172	0.4417
LABEL	0.6672	0.5067	0.6075	0.4265	0.5883	0.4104	0.5821	0.4034
SAMETYPE	-	-	0.6024	0.4452	0.5917	0.4327	0.5813	0.4256
PATH	-	-	0.6507	0.4956	0.6538	0.4974	0.6315	0.4742
DEPTH	0.7432	0.6128	0.6754	0.5385	0.6797	0.5475	0.6741	0.5354
ANCESTORS	0.7424	0.6154	0.6967	0.5637	0.6949	0.5662	0.6879	0.5562
ANC DEPTH	0.7469	0.6236	0.6832	0.5488	0.6885	0.5546	0.6796	0.5423
DEC-TREE	0.7614	0.6361	0.7373	0.6079	0.7979	0.7019	0.7943	0.6914
LIN-REG	0.7373	0.6079	0.6906	0.5579	0.6987	0.5702	0.6899	0.5529
TransE	0.8203	0.6675	0.7322	0.6122	0.8021	0.6339	0.7543	0.6343
TransH	0.8198	0.6631	0.7967	0.6544	0.8502	0.7394	0.8324	0.7141
TransR	0.7623	0.5509	0.7021	0.5687	0.8123	0.6913	0.7230	0.6032
RL-TRank (TransE)	0.8390	**0.6892**	0.7618	0.6434	0.8673	0.7078	0.7718	0.6577
RL-TRank (TransH)	**0.8401**	**0.6892**	**0.8289**	**0.6896**	**0.8962**	**0.7899**	**0.8435**	**0.7288**
RL-TRank (TransR)	0.7776	0.5700	0.7356	0.6057	0.8681	0.7535	0.7394	0.6216

Table 5.4: Performance in different embedding dimension d.

Approach	Dimension	EO		SC		PG		3PG	
		NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP
TransE		0.5888	0.0.4792	0.5029	0.3727	0.6559	0.5239	05650.	0.4623
TransH	10	0.5744	0.4782	0.5162	0.4050	0.5957	0.4753	0.6107	0.5273
TransR		0.4039	0.3201	0.3987	0.3263	0.5139	0.3716	0.5222	0.4043
TransE		0.6107	0.5056	0.5511	0.4133	0.6824	0.5346	0.6292	0.5097
TransH	20	0.6108	0.5006	0.6283	0.5071	0.6212	0.4700	0.6777	0.5566
TransR		0.5469	0.4449	0.4223	0.3450	0.5384	0.4069	0.5417	0.4116
TransE		0.7796	0.5622	0.6583	0.5041	0.7068	0.5832	0.6890	0.5549
TransH	30	0.7328	0.5227	0.7236	0.5767	0.7345	0.6115	0.7001	0.5701
TransR		0.7225	0.5218	0.5215	0.4215	0.6274	0.4974	0.6442	0.5142
TransE		**0.8203**	**0.6575**	0.7322	0.6123	0.8021	0.6339	0.7511	0.6411
TransH	40	0.8177	0.6386	0.7239	0.5766	0.8003	0.7834	0.7699	0.6511
TransR		0.7564	0.6119	0.6098	0.4773	0.7757	0.6259	0.6786	0.5433
TransE		0.8112	0.6331	0.7304	0.5815	0.8019	0.6529	0.7543	0.6326
TransH	50	0.8198	0.6631	**0.7967**	**0.6544**	**0.8502**	**0.7394**	**0.8324**	**0.7141**
TransR		0.7623	0.6155	0.7021	0.5687	0.8023	0.6913	0.7230	0.6042
TransE		0.8074	0.6102	0.7062	0.5362	0.7604	0.6243	0.7213	0.5891
TransH	75	0.8045	0.6102	0.7709	0.6432	0.8200	0.6688	0.8079	0.6578
TransR		0.7612	0.4932	0.6982	0.5282	0.8099	0.6786	0.6972	0.5627
TransE		0.7434	0.5407	0.6936	0.5522	0.7412	0.6143	0.7121	0.5808
TransH	100	0.7510	0.5407	0.7637	0.6025	0.8055	0.6657	0.7828	0.6513
TransR		0.7001	0.5201	0.6442	0.5017	0.7636	0.6441	0.6812	0.5303

We also observed that the embedding dimension of SE models affect the performances of the RL-TRank models. In Table 5.4, we show the NDCG and MAP values obtained by changing the embedding dimension d. TransE achieved the best results with $d = 40$ for the EO dataset while for other datasets, TransH performed better with $d = 50$. Though TransR showed poor result consistency, its effectiveness was low if d is set to 10 or 20. We observed that for $d > 50$ the performances decrease gradually for all models in our four datasets. In Fig. 5, we show the evolution of NDCG as a function of the embedding dimension, d for all the models in four datasets. We mentioned earlier that the triples with "relevance" relations are neither symmetric nor transitive. So, we employed simple SE models in RL-TRank rather than using complicated SE models e.g., ComplEx [4], DistMult [43]. However, it is comforting to see that the performance difference with the baseline models becomes wider when involving probabilistic embedding, because the probabilistic information provide additional information to embed entities and corresponding types, especially for sparse datasets.

5.4 Conclusion

In this chapter, we have presented a novel way to apply representation learning models to the entity type ranking task by introducing a "relevance" property between an entity and its corresponding entity types. One of the challenges of this task was building a KG to serve the ranking purpose. We described a way of creating a KG to tackle type ranking problem. We showed that knowledge graph embeddings can be effectively used in the entity type ranking problem. To do this we defined a ranking function based on the score of $(e, relevance, tp)$ triples. Later, we have experimentally shown how parameter tuning (especially the embedding dimension) affects the performances of the SE models. We compared our model against baseline methods using previously adopted benchmark datasets. We observed that RL-TRank models are significantly more effective on the entity type ranking task than the baseline models.

6

Joint Representation Learning with Text and Knowledge Graph Data for Triple Scoring

In this chapter, we extend the ideas of scoring the types relevant to entities. We devise an ensemble of five base scorers, so as to leverage the power of both text and knowledge graph embedding for that task. Our method inspired by two literatures [1, 2] which also handle the same task for scoring the triples.

6.1 Introduction

We mentioned earlier that the KB contains many entity types that are confusing for humans when querying a KB. For example, Barack Obama has four professions listed in Freebase, namely Politician, Lawyer, Law professor, and Author, but it is considered that people primarily want to retrieve Barack Obama as a Politician. Recently, Bast et

al. [1] addressed this problem by assigning a relevance score to each pair consisting of an entity and its type in KB. These scores enable us to enhance the ranking results of entity retrieval tasks by sorting the results based on these relevance scores.

We perform the Triple Scoring Task as the similar way as WSDM Cup 2017 [87, 1]. In this task type-like relations between persons and professions/nationalities have been addressed. The relevance score is the sum of seven crowdworkers' votes on whether they think that each profession/nationality is primary for the specific person. Hence, the score ranges from 0 to 7 by an integer. As an additional source of information Wikipedia sentences have been provided, which include annotations of persons, although according to the rules other data could be used. Small portions of relevance scores were also provided as training data.

As we mentioned earlier this task was introduced by WSDM Cup 2017, on that competition 21 teams made a submission and the team BOKCHOY won the first award based on accuracy. BOKCHOY extended the baseline paper and introduced path ranking to the model. This model solely learns from statistical method and relational path. In fact, in most KGs there are also concise descriptions for entities, with rich semantic information about these entities.

In our approach, we mainly rely on a set of structured facts, unstructured text and entity description retrieved from Wikipedia. This study presents a joint learning approach that learns entity vectors by leveraging resources of both raw and labeled text as well as graph knowledge. We first present a Joint Representation Learning with Text and Knowledge Graph Data (RL-TKG) for type-like triple scoring. Here we exploit the paragraph vector (PV) model [88] though a different training approach is adopted in our study. Proposed RL-TKG achieves the best performance in the accuracy metric compare to the other participants in the WSDM Cup 2017 triple scoring task.

6.2 Problem Statement

The triple scoring task is to predict, for each triple from type-like relations, a relevance score in the range of 0-7. Two types of such relations are considered, i.e., profession and nationality [87]. The prediction task is confined to 385,426 different persons, 200 different professions, and 100 distinct nationalities, all contained in Freebase. Let E, P, and N denote the sets of persons, professions, and nationalities, respectively. Training

data includes:

- wiki-sentences: 33,159,353 sentences from the English version of Wikipedia with annotations of the 385,426 persons in E;
- profession.kb: all professions for a subset of 343,329 persons, extracted from a 14-04-2014 dump of Freebase;
- nationality.kb: all nationalities for a subset of 301,590 persons, extracted from the same dump;
- profession.train: relevance scores for 515 triples (pertaining to 134 persons) from profession.kb;
- nationality.train: relevance scores for 162 triples (pertaining to 77 persons) from nationality.kb.

Results are evaluated on a test set consisting of triples from the two .kb files, but with relevance scores as ground truth. Three metrics are used for evaluation, i.e., accuracy (ACC), average score difference (ASD), and Kendall's tau (TAU).

6.3 RL-TKG

In Figure 6.1, we show the illustration of our approach.

6.3.1 Model Components

This model has five components: Word Classification Model , Word Counting Model, Generative Model (Word MLE), Path Ranking and Description Embeddings. Beside that we also briefly describe the text selection technique to train the models.

Text Selection for Model Training

We adopt the similar technique as described in [1, 2] for text data selection for training. We also exploit the English Wikipedia, utilizing the fact that persons in Freebase often have a link to their Wikipedia article. We need information about the entities which appear in the profession.kb and nationality.kb files. Moreover, we also collected the

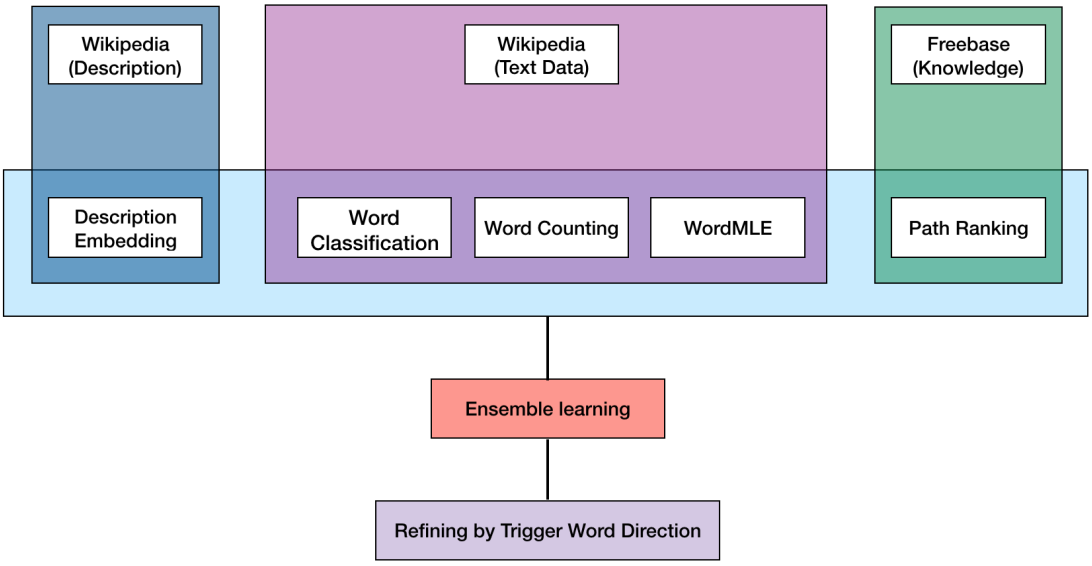

Figure 6.1: An overview of the RL-TKG. The blue part is introduced by this study (our), the pink and green parts are proposed by Bast et al. [1] and BOKCHOY [2] respectively.

Wikipedia description for each profession or nationality e.g., professon "Actor"[1] has a descriptive page on Wikipedia. For each profession P or each nationality N we select, from the profession.kb file/nationality.kb, people with only that profession/nationality as positive candidates, and people who do not have that profession/nationality at all as negative candidates. Then we extract the corresponding sentences of those entities. Afterward, we preprocess the raw text and prepare for the models.

Word Classification Model

In this model logistic regression has been introduced to train whether a profession/-nationality is primary or secondary for a given person according to the [1]. From the extracted text, word counts have considered as feature. Positive and negative examples have chosen randomly in training phase. The ratio of the positive and the negative examples were same. L2-regularization is used during training on the balanced examples (positive and negative). These models can then be used for persons

[1]https://en.wikipedia.org/wiki/Actor

with multiple professions/nationalities to make a binary decision if each of his or her profession/nationality is primary or not (secondary).

Word Counting Model

This model takes words as indicators of professions/nationalities, and based on the extracted Wikipedia sentences it predicts a person's most relevant profession/nationality by judging how much his/her associated text is indicative of that profession/nationality. Here, we employ TF-IDF as described in [1]. For each profession in P, we construct a training corpus consisting of text associated with only the positive candidates and calculate the TF-IDF value for each word in the training corpus, and weight that word by its tf-idf value. To speed up the learning process, we consider only the top 130,000 words with highest frequencies in the corpus. So scoring function can be defined as follows:

$$s = \sum_{w_i} n_{w_i} \cdot tf - idf_{w_i} \tag{6.1}$$

where $n_{w_i}$is the number of times the word w_i occurs in the person's associated text and $tf - idf_{w_i}$ is the weight of that word.

Generative Model (Word MLE)

Our third base scorer is a generative model and we named it as Word MLE model because it utilizes Maximum Likelihood Estimation (MLE). The idea of a generative model, Word MLE is to generate the texts associated with a person given his/her profession/nationality. For a person with k professions/nationalities and n word occurrences in the associated text: in the next step, we pick one of the k professions (here we show the equation with the profession example) with probability $P(p_i)$; generate word w_j with $P(w_j|p_i)$; repeat until all n words are generated. The joint probability for word w and profession p is then $P(w, p) = P(w|p)P(p)$. The log-likelihood of generating the n words from the k professions is:

$$\log L = \sum_{j=1}^{n} [tf_j \cdot \log \sum_{i=1}^{k} P(p_i)P(w_j|p_i)] \tag{6.2}$$

where tf_j is the frequency of word w_j , computed as its tf-idf value in the text associated with all persons in E. Only the top 20,000 words with highest frequencies are kept for efficiency reasons. To compute $P(w_j|p_i)$, we collect text associated with the positive candidates for profession p_i, and calculate the tf-idf value of word w_j. The training and the parameter settings of this model adopted from [1].

Path Ranking

Our fourth base scorer is knowledge graph path ranking algorithm. Path ranking approach usually used for reasoning on knowledge bases. The key idea is to build for each relation a binary classifier, with paths that connect two entities as features, to predict whether the two entities should be linked by that relation or not. To obtain labeled training data for path ranking, we use the same dataset as [2]. In the feature extraction step, we conduct Depth-First Search (DFS) to enumerate all paths with a bounded length of $l \leq 3$ between the two entities as described in [2]. After feature extraction, we simply compute the value of each feature as the number of times it appears in each labeled example. Then we adopt Random Forest Classifier to train a binary classifier. We randomly split the labeled examples into 70% training and 30% validation. Parameter settings are adopted from BOKCHOY [2] as we have extended the model to get better performance.

Learning from Description Embeddings

In RL-TKG we introduced a simple description embeddings model based on word2vec technique [16]. We collected the descriptions of entities and all the Wikipedia articles of professions (total 280 professions according to the dataset) and nationalities. Learning entity/profession/nationality vectors with raw text can be simply implemented by treating entities/professions/nationalities as words (or phrases) and learning them together with other words. To keep the model simple, we have employed skip-gram technique to calculate such vectors [16]. The simple approach to learning with labeled text is to average the vectors of words involved in the description of the current head or tail. It can be expresses as (here we show it for head entity for tail it works similarly):

$$v_e = \frac{1}{\mid D_e \mid} \sum_{w \in D_e} v_w \tag{6.3}$$

where v_e denotes the vector of entity e, and v_w denotes the vector of word w. D_e is the set of word tokens within the description of e, and $\mid D_e \mid$ represents the size of D_e. We minimize the following margin-based score function as objective for training:

$$L_D = \sum_{P_i \in P}^{K} max\{0, \gamma - f(e_h + r, e_t) + f(e_h^{'} + r, e_t^{'})\} \tag{6.4}$$

Here,γ is the margin, e_h, e_t denotes the vector of head entity (prepared as difined by 6.3) and r is the profession/nationality relation in this case.$f(\cdot)$ is the dissimilarity function between $e_h + r$ and e_t. Denote the related entity-type pairs defined by the set $P = P_i; P_i = (e_h, e_t)$. For each pair P_i, the negative sampling corrupts the pair by replacing either the head entity e_h or the tail entity, e_t with a randomly selected entity.

6.3.2 Adjusting the Weights of Trigger Words

It is a usual case that first sentence of the Wikipedia page contains the most valuable information about entity's profession and nationality. Picking those words may reveal the entities' true profession/nationality. Trigger words are those that have the same meaning with the entity type. One example would clear the concept. Trigger words of a profession include (i) the original and plural forms, e.g., actor and actors for Actor, (ii) synonyms, e.g., enterpriser for Entrepreneur, and (iii) hyponyms, e.g., runner andjumper for Athlete. Synonyms and hyponyms are obtained from WordNet 3.0. It also works same for the nationality. This trigger word concept introduced by [2]. For RL-TKG we adopt their heuristic system to get better performance.

6.3.3 Score Mapping

In this work, we employ five different scoring functions to judge the triples. In the ground truth dataset, the score ranges from 0 to 7 by an integer. So, it is required to map such results to integer triple scores in the range of 0-7. Three strategies have been employed and they are as follows:

Maplin

It is a very basic linear strategy introduced in [1]. Linearly scale computed values to the range 0 to 7. In practice, just divide all sums or probabilities by the highest one. Then multiply by 7 and round to the nearest integer. For mapping a value s to a triple score s^m as: where s_{max} is the highest value that a person get for all his/her professions/nationality.

$$s^m = \left\lfloor \frac{s}{s_{max}} \times 7 \right\rfloor \tag{6.5}$$

Maplog

To scale the values to the range 0-7 obtained from different scorer in a way that the next highest score corresponds to twice the sum or probability. In practice, divide all sums or probabilities by the highest one. Then multiply by 2^7 , take the logarithm to base 2, convert it to integer. It can be defined as:

$$s^m = \left\lfloor max\{0, \log_2(\frac{s}{s_{max}} \times 2^7)\} \right\rfloor \tag{6.6}$$

Note that as here logarithm to base 2 has been employed so for very small values of s we set s^m to 0.

Mapscale

Mapscale is proposed in [2]. It is a linear mapping applied only on probabilities:

$$s^m = \lfloor s \times 8 - \epsilon \rfloor \tag{6.7}$$

where $\epsilon = 10^{-4}$ so that we can get $s = 7$ with $s = 1$. Table 6.1 summarizes the mapping strategies used in the four base scorers for each of the two relations.

6.3.4 Model Training

As we mentioned earlier proposed RL-TKG is an ensemble learning model and we extend the idea of [1, 2]. We compute the scores of the triples by five different scorers and then we map them to the score ranges from 0 to 7 by an integer. We jointly learn

Table 6.1: Mapping strategies used in the five base scorers.

Models	Profession	Nationality
Word Classification	Mapscale	Maplog
Word Counting	Maplin	Maplin
Word MLE	Maplog	Maplog
Path Ranking	Mapscale	Maplog
Description Embeddings	Maplin	Maplin

the triple scores to achieve better predictive performance. Here we choose weighted averaging [89], which defines a triple t's relevance score $S(t)$ as:

$$S(t) = \left\lfloor \sum_{i=1}^{M_t} w_i s_i(t) \right\rfloor \tag{6.8}$$

Here, M_t is the number of base scorers that are used to score the triple; $s_i(t)$ an integer relevance score in the range of 0-7 predicted by the i-th base scorer; and $w_i = ACC_i / \sum_{j=1}^{M_t} ACC_j$ the weight of that base scorer. ACC_j is the ACC value that the j-th base scorer yield on the corresponding .train file.

6.4 Experiment

Here, we discuss in detail our experimental findings and their implications.

6.4.1 Evaluation Measures

We use Accuracy (AUC), Average Score Difference (ASD) and Kendall's Tau (TAU) to assess the quality of the model we proposed.

- Accuracy (AUC): the percentage of triples for which the score computed by the system differs from the score in the ground truth by at most 2.
- Average Score Difference (ASD): for each triple, take the absolute difference of the system score and the score from the ground truth; add up these differences and divide by the number of triples.

- Kendall's Tau (TAU): for each relation, for each subject, compute the ranking of all triples with that subject and relation according to the scores computed by the system and the score from the ground truth. Compute the difference of the two rankings using Kendall's Tau.

Table 6.2: Evaluation results on Profession Dataset

Models	ACC	ASD	TAU
Ensemble (BOKCHOY)	0.84	**1.44**	**0.25**
− Word Classification	0.83	1.47	0.25
− Word Counting	0.83	1.44	0.25
− Word MLE	0.81	1.52	0.28
− Path Ranking	0.82	1.44	0.25
Gailan	0.68	1.94	0.34
Radicchio	0.752	1.722	0.284
Celosia	0.69	1.74	0.35
Chicory	0.62	2.03	0.34
Cress	0.78	1.61	0.27
Word Classification	0.74	1.79	0.32
Word Counting	0.72	1.69	0.30
Word MLE	0.59	2.46	0.31
Path Ranking	0.67	2.06	0.40
Description Embeddins	0.70	2.07	0.42
Ensemble (RL-TKG)	**0.86**	1.53	0.32
− Word Classification	0.82	1.44	0.27
− Word Counting	0.83	1.40	0.26
− Word MLE	0.80	1.57	0.27
− Path Ranking	0.83	1.44	0.27
- Description Embeddins	0.83	1.47	0.25

6.4.2 Results

We evaluated the RL-TKG approach in the Triple Scoring Challenge at WSDM Cup 2017. The tasks consisted in scoring triples from datasets containing persons' nationalities and professions. The participants were free to use data from any source as well as any amount of computation. The training phase consisted of evaluating 515 triples

Table 6.3: Evaluation results on Nationality Dataset

Models	ACC	ASD	TAU
Ensemble (BOKCHOY)	**0.89**	**1.27**	**0.30**
− Word Classification	0.90	1.25	0.28
− Word Counting	0.89	1.27	0.29
− Word MLE	0.91	1.34	0.29
− Path Ranking	0.89	1.28	0.30
Gailan	0.80	1.40	0.39
Radicchio	0.779	1.689	0.428
Celosia	0.77	1.55	0.42
Chicory	0.66	1.82	0.38
Cress	0.77	1.62	0.40
Word Classification	0.72	1.82	0.45
Word Counting	0.76	1.68	0.45
Word MLE	0.64	2.09	0.41
Path Ranking	0.76	1.57	0.44
Description Embeddins	0.75	1.64	0.46
Ensemble (RL-TKG)	**0.89**	1.44	0.38
− Word Classification	0.89	1.29	0.30
− Word Counting	0.89	1.32	0.31
− Word MLE	0.89	1.37	0.28
− Path Ranking	0.88	1.29	0.33
- Description Embeddins	0.89	1.40	0.38

(pertaining to 134 persons) from professions and 62 triples (pertaining to 77 persons) from nationalities.

We first show the performance of using each of the five base scorers alone. The results are shown in Table 6.2 and Table 6.3 (second row from the bottom). We can see that (i) Word Classification, Word Counting and Description embedding perform quite well on both relations, but Word MLE substantially worse than them.

We further investigate different strategies to combine the base scorers into an ensemble. Please note that we exploit the technique refining by trigger word detection and find significantly better results, on both relations and with all the ensemble strategies. The results are given in the second part of 6.2 and Table 6.3 (bottom row), where "Ensemble" means combining all the five base scorers, and "Ensemble−Description Embeddings", for example, combining the other four base scorers except Description

Embeddings. We can see that combining multiple models generally performs better than using a single model alone, and Ensemble gets relatively good performance among these strategies; Ensemble−Word MLE performs even better than Ensemble-RL-TKG (in AUC), due to the low performance of Word MLE; Ensemble−Description Embeddings shows moderate performance among these strategies. It is worth to notice that Path Ranking strategy derived form freebase data and Description Embeddings from Wikipedia data. We observed that Ensemble (RL-TKG) worked better in AUC metric but for other metric it could not achieve the best performance.

6.5 Conclusion

In this chapter we described our joint representation learning model with text and knowledge graph data for triple scoring to task. Paragraph Vectors are trained to represent documents that describe each property value (in our case each profession or nationality). These vector representations are then used to measure the proximity between an instance (person) representation vector and the vector that represents the attribute to score (profession/nationality). We presented our experimental results on the test datasets. . We achieved good performance on AUC metric but there are a lot of scopes to improve in future.

7 Conclusion

In this thesis, we investigated, designed, and evaluated a number of methods and algorithms to exploit knowledge graphs and to increase the quality of their data. Our work contributed to advancing the state-of-the-art in several tasks related to knowledge graphs, namely, knowledge graph embedding for knowledge graph completion, entity type ranking, and type-like triple scoring. We also studied how other applications can benefit from knowledge graphs by designing and evaluating an entity-centric system considering entity types.

7.1 Summary

The lessons we have learned in this context are numerous and are related to the various aspects of research on effect of entity type on knowledge graphs we have explored so far.

In Chapter 4, we leveraged the entity types for knowledge graph embedding. Knowledge graph (KG) is the most popular method for presenting knowledge in search

engines and other natural-language processing (NLP) applications. To deal with the challenges of KGs, many state-of-the-art models have been proposed. An issue is that these proposed models ignore the category-specific projection of entities based on relation type and also the problem of polysemy relations. An entity may therefore involve multiple types or aspects. Considering all entities in just one semantic space is therefore not a logical approach to building an effective model. So, we proposed TPRC, which maps each entity based on its type considering the relation. We can then apply any other existing translation-distance-based embedding models such as TransE or TransR or TransD. We evaluated our model using two tasks that involve link prediction and triple classification. Our model achieved a significant and consistent improvement over other the state-of-the-art models. The strength of this model is that it can be combined easily with other translation-distance-based models to improve accuracy without making the models more complex.

In Chapter 5, we presented a representation learning based model for entity type ranking. Usually entities in knowledge graphs can be associated with several types, and we tackled the task of selecting the best type to show to users given an entity. Surprisingly, we realized that always returning the most specific type is often not the best choice. We defined a novel way of KG construction using the type information. We showed how we can use the knowledge graph embedding techniques to such graph and retrieve a ranked list of entity types.

In the later chapter, we devised an ensemble of five base scorers for triple scoring, so as to leverage the power of joint text and knowledge bases for that task. We compared our results with the participants of WSDM Cup 2017.

7.2 Future Work

In a broader context, we view our work as one of many contributions in NLP and Semantic Web that study the combination of structured and unstructured information (for different tasks). In the following we present some compelling ideas that could be pursued as an extension of this work and that can help advancing the current state of knowledge graph technologies and semantic web applications.

7.2.1 Utilizing Useful Information of Knowledge Graphs

Beside type information, knowledge graphs also include other vital information e.g., literals. So the target is to exploit available information present in the original KG, its latent representation as being only an approximation of the original KG, will perform equally well on tasks that depend on its semantic information content. Proper representation of entities and relations by including the datatyped literals will increase the model's semantic content and might thereby lead to quality improvement. High-order relational dependency of multiple relation facts [90, 91] can be captured to improve the characterization of a knowledge graph, and meanwhile can help logical rule induction from the latent representations of the relations.

7.2.2 Extending TPRC with Bilinear and Neural Network based Models

Recently, TuckER showed very impressive performance in the link prediction task. It's a fully expressive model and addressed the major issues of the KGE. We think it would be a interesting experiment to add the idea of TPRC with the TuckER model. Convolutional models are very parameter efficient. So it would be worth checking the performance of these models using type information in the relational context.

7.2.3 Actionable Knowledge Graphs (AKG) Generation

Knowledge graph has become an increasingly common and important component in search engine result pages (SERPs). Popular search engines have recently utilized the power of KGs to provide specific answers to queries in a direct way. SERPs are expected to provide facts in response to queries that satisfy semantic meaning. This encourages researchers to propose more influential knowledge graph generation techniques.

The purpose of actionable knowledge graph (AKG), as designed by NTCIR-13 is to select and rank attributes of entities in KGs that can best support "actionable" search intents. The objective of actionable knowledge graph is to foster research on generating knowledge graphs that are optimised for facilitating users' actions e.g., buying, booking, downloading, travelling, etc. NTCIR-13 first introduced this concept as a task. We have participated on that task and submitted our results. There are

several challenging issues connected to actionable knowledge graphs to be tackled. First of all, information on actions that can be performed on specific entities should be collected. Another possible task consists in understanding what actions users want to perform on certain entities, and map such transactional needs to their representations in the knowledge graph. If the interface between the knowledge graph and the user is a search box, the task is connected to Ad-hoc Object Retrieval. A future goal is generating an AKG in SERPs to support action-oriented search intentions, which is challenging but would be very useful to the end users.

7.2.4 Knowledge Graph Centric Research

The rapid increasing popularity of knowledge graphs for Web search will foster a renaissance of the Semantic Web vision: A web, in which terms have semantics attached and reasoning over different data sources is possible. The Web as a global Information System has revolutionized everyday life. As one of the most disruptive technologies of the last decades, the Web was responsible for drastic technological, economical, and social developments: it is well established as main source of information and entertainment, but is also the most influential infrastructure for commerce and business. Specially, the introduction of Linked Open Data (LOD) and the creation of knowledge graphs to collect, interlink, and access data about entities had far-reaching consequences: The Web has prospered, diversified and developed into the largest and omnipresent information source. Thus, a current research goal is bridging the gap between rich, yet rather unstructured LOD sources and modern graph databases featuring expressive query languages. Here, we discuss new directions to harness the power of graph-based query processing for accessing data from the LOD cloud or knowledge graphs in a data-driven way.

7.3 Outlook

Knowledge graphs are becoming an increasingly popular way of thinking about and organising data within major business firms. As with all data management and governance projects, we can define the use of KG and achieve the expected growth in managing the data. The way KG is growing it may become the new data management

system.

The modern businesses increasingly adopting machine learning approaches for decision making, it seems likely that knowledge graph technology will also evolve hand-in-hand. As well as being a useful format for feeding training data to algorithms, machine learning can quickly build and structure graph databases, drawing connections between data points that would otherwise go unnoticed. Machine learning is great for answering questions, and knowledge graphs are a step towards enabling machines to more deeply understand data in all formats such as text, video and audio that don't fit neatly into the rows and columns of a relational database.

It's a new era of entity centric data based research, in which KG would be a key factor definitely. The Internet could thus shift from being a document centric resource to being an entity-centric repository of knowledge and services, that is, a knowledge graph. If this happens only search engine might be enough for answering any types of queries.

Bibliography

[1] Hannah Bast, Björn Buchhold, and Elmar Haussmann. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 243–252, 2015.

[2] Boyang Ding, Quan Wang, and Bin Wang. Leveraging text and knowledge bases for triple scoring: an ensemble approach-the bokchoy triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08356*, 2017.

[3] 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 (ACL-IJCNLP)*, volume 1, pages 687–696, 2015.

[4] Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research (JMLR)*, 18(1):4735–4772, 2017.

[5] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2965–2971, 2016.

[6] Xiaotian Jiang, Quan Wang, and Bin Wang. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American*

Chapter of the Association for Computational Linguistics : Human Language Technologies (NAACL-HLT), pages 978–987, 2019.

[7] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data-mining (KDD)*, pages 353–362. ACM, 2016.

[8] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[9] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2015.

[10] 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 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1247–1250, 2008.

[11] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the 31st Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, 2017.

[12] Chen Cai. Group representation theory for knowledge graph embedding. *arXiv preprint arXiv:1909.05100*, 2019.

[13] Md Mostafizur Rahman and Atsuhiro Takasu. Leveraging entity-type properties in the relational context for knowledge graph embedding. *IEICE Transactions on Information and Systems*, 103(5):958–968, 2020.

[14] Md Mostafizur Rahman and Atsuhiro Takasu. Knowledge graph embedding via entities' type mapping matrix. In *Proceedings of the 25th International Conference on Neural Information Processing (ICONIP)*, pages 114–125. Springer, 2018.

[15] Md Mostafizur Rahman, Atsuhiro Takasu, and Gianluca Demartini. Representation learning for entity type ranking. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing (ACM/SIGAPP SAC)*, pages 2049–2056, 2020.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3111–3119, 2013.

[17] Md Mostafizur Rahman and Atsuhiro Takasu. Tlab at the ntcir-13 akg task. In *Proceedings of the NII Testbeds and Community for Information Access Research 2013 (NTCIR-13) Conference*, 2017.

[18] Md Mostafizur Rahman and Atsuhiro Takasu. Entity oriented action recommendations for actionable knowledge graph generation. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 686–693, 2017.

[19] John Domingue, Dieter Fensel, and James A Hendler. *Handbook of semantic web technologies*. Springer Science & Business Media, 2011.

[20] Virgil Pavlu, Shahzad Rajput, Peter B Golbus, and Javed A Aslam. Ir system evaluation using nugget-based test collections. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 393–402, 2012.

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

[22] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing wikidata to the linked data web. In *Proceedings of the 13th International Semantic Web Conference (ISWC)*, pages 50–65. Springer, 2014.

[23] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[24] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, pages 722–735. Springer, 2007.

[25] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[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 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1247–1250, 2008.

[27] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pages 1419–1428, 2016.

[28] 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)*, pages 601–610, 2014.

[29] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[30] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in web search. In *Proceedings of the twelfth International Semantic Web Conference (ISWC)*, pages 33–48. Springer, 2013.

[31] Changsung Kang, Dawei Yin, Ruiqiang Zhang, Nicolas Torzec, Jianzhang He, and Yi Chang. Learning to rank related entities in web search. *Neurocomputing*, 166:309–318, 2015.

[32] Kedar Bellare, Carlo Curino, Ashwin Machanavajihala, Peter Mika, Mandar Rahurkar, and Aamod Sane. Woo: A scalable and multi-tenant platform for

continuous knowledge base synthesis. *Very Large Data Bases (VLDB)*, 6(11):1114–1125, 2013.

[33] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of the 15th International Conference on ArtificialArtificial Intelligence and Statistics (AISTATS)*, pages 127–135, 2012.

[34] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th Association for the Advancement of Artificial Intelligence (AAAI) Conference on on Artificial Intelligence*, volume 6, page 6, 2011.

[35] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 2787–2795, 2013.

[36] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, volume 15, pages 2181–2187, 2015.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, volume 14, pages 1112–1119, 2014.

[38] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the 32nd Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, 2018.

[39] Zhiqing Sun, Zhi-Hong Deng1, Jian-Yun Nie3, and Tang Jian. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[40] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Hyperbolic knowledge graph embeddings for knowledge base completion. In *Proceedings of the 17th European Semantic Web Conference (ESWC)*, pages 199–214, 2020.

[41] 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 (ICML)*, pages 809–816, 2011.

[42] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 271–280, 2012.

[43] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[44] Ivana Balazevic, Carl Allen, and Timothy" Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China, 2019. Association for Computational Linguistics.

[45] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, pages 4284–4295, 2018.

[46] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 926–934, 2013.

[47] Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*, 2016.

[48] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32nd*

Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence, 2018.

[49] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha P Talukdar. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the 34th Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, pages 3009–3016, 2020.

[50] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 84–94, 2015.

[51] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. Sse: Semantically smooth embedding for knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 29(4):884–897, 2016.

[52] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 585–591, 2002.

[53] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[54] Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *Proceedings of the 14th International Semantic Web Conference (ISWC)*, pages 640–655. Springer, 2015.

[55] Alberto Tonon, Michele Catasta, Gianluca Demartini, Philippe Cudré-Mauroux, and Karl Aberer. Trank: Ranking entity types using the web of data. In *Proceedings of the 12th International Semantic Web Conference (ISWC)*, pages 640–656. Springer, 2013.

[56] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Irreflexive and hierarchical relations as translations. *arXiv preprint arXiv:1304.7158*, 2013.

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

[58] Hannah Bast, Björn Buchhold, and Elmar Haussmann. Overview of the triple scoring task at the wsdm cup 2017. *arXiv preprint arXiv:1712.08081*, 2017.

[59] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys)*, pages 161–169. ACM, 2017.

[60] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the 27th International Conference on World Wide Web (WWW)*, pages 729–739. International World Wide Web Conferences Steering Committee, 2018.

[61] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540*, 2018.

[62] Guangyuan Piao and John G Breslin. A study of the similarities of entity embeddings learned from different aspects of a knowledge base for item recommendations. In *Proceedings of the 15th European Semantic Web Conference (ESWC)*, pages 345–359. Springer, 2018.

[63] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data-mining (KDD)*, pages 855–864. ACM, 2016.

[64] Esraa Ali, Annalina Caputo, and Séamus Lawless. Triple scoring using paragraph vector-the gailan triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08360*, 2017.

[65] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 529–539. Association for Computational Linguistics, 2011.

[66] Yael Brumer, Bracha Shapira, Lior Rokach, and Oren Barkan. Predicting relevance scores for triples from type-like relations using neural embedding-the cabbage triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08359*, 2017.

[67] Ikuya Yamada, Motoki Sato, and Hiroyuki Shindo. Ensemble of neural classifiers for scoring knowledge base triples. *arXiv preprint arXiv:1703.04914*, 2017.

[68] Edgard Marx, Tommaso Soru, and André Valdestilhas. Triple scoring using a hybrid fact validation approach-the catsear triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08352*, 2017.

[69] Nausheen Fatma, Manoj K Chinnakotla, and Manish Shrivastava. Relevance scoring of triples using ordinal logistic classification-the celosia triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08673*, 2017.

[70] Frank Dorssers, Arjen P de Vries, Wouter Alink, and Roberto Cornacchia. Ranking triples using entity links in a large web crawl-the chicory triple scorer at wsdm cup 2017. *arXiv preprint arXiv:1712.08355*, 2017.

[71] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[72] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data-mining (KDD)*, pages 1156–1165. ACM, 2014.

[73] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys)*, pages 32–36. ACM, 2017.

[74] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding

of user preferences. In *Proceedings of the 28th International Conference on World Wide Web (WWW)*, pages 151–161, 2019.

[75] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3(Feb):1137–1155, 2003.

[76] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[77] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32nd Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, pages 1811–1818, 2018.

[78] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence (AAAI) conference on Artificial Intelligence*, 2016.

[79] Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies (NAACL-HLT)*, pages 1470–1480, 2018.

[80] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1509, 2015.

[81] Yashar Moshfeghi, Michael Matthews, Roi Blanco, and Joemon M Jose. Influence of timeline and named-entity components on user engagement. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 305–317. Springer, 2013.

[82] He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *arXiv preprint arXiv:1704.07130*, 2017.

[83] Yuyu Zhang, Hanjun Dai, Kamil Toraman, and Le Song. Kgˆ 2: Learning to reason science exam questions with contextual knowledge graph embeddings. *arXiv preprint arXiv:1805.12393*, 2018.

[84] Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. Neural collective entity linking. *arXiv preprint arXiv:1811.08603*, 2018.

[85] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, 2014.

[86] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 260–269, 2016.

[87] Stefan Heindorf, Martin Potthast, Hannah Bast, Björn Buchhold, and Elmar Haussmann. Wsdm cup 2017: Vandalism detection and triple scoring. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 827–828, 2017.

[88] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1188–1196, 2014.

[89] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data-mining (KDD)*, pages 226–235, 2003.

[90] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender

systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data-mining (KDD)*, pages 974–983, 2018.

[91] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3146–3154, 2019.