Entity Alignment and Attribute Enhancement between Knowledge Graphs

by

Rumana Ferdous Munne

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

March 2023

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

Advisory Committee

1.	Prof. Seiji Yamada	National Institute of Informatics SOKENDAI
2.	Prof. Ken Satoh	National Institute of Informatics SOKENDAI
3.	Prof. Hideaki Takeda	National Institute of Informatics SOKENDAI
4.	Prof. Akiko Aizawa	National Institute of Informatics SOKENDAI
5.	Prof. Ryutaro Ichise	Tokyo Institute of Technology National Institute of Informatics

Acknowledgments

The completion of this study could not have been possible without tremendous support from many. All of them contributed in their own way and proved to be essential to my academic and professional growth, my personal development, and my mental health. Here, I want to thank them all.

My sincere and deep gratitude goes first and foremost to my supervisor, Professor Ryutaro Ichise, for his kind support, invaluable patience, and feedback. The path to obtaining a Ph.D. is long and difficult, and his insightful guidance has helped to get through this journey.

I am grateful to my advisory committee, Prof. Seiji Yamada, Prof. Ken Satoh, Prof. Hideaki Takeda, Prof. Akiko Aizawa and Prof. Atsuhiro Takasu. I highly appreciate their insightful and helpful comments.

I would like to thank all the members and internship students in our laboratory. I am thankful that I had the chance to work with them, and I am proud of the work we have done together.

Finally, I owe eternal gratitude to my son Ayan for being a patient companion throughout this process, my parents, Md Shariful Islam and Rowshan Ara, my brother Mohammad Saiful Islam for supporting me unconditionally, and my husband, Rabbi, without his constant encouragement this journey would not have been possible. They inspired and helped me become who I am.

Abstract

A Knowledge Graph (KG) is a knowledge model containing facts about real world entities represented as a graph. It is a collection of interlinked descriptions of entities, relationships, concepts, and events. We have witnessed rapid growth in knowledge graph creation and application in last few years. Several efforts have been made to develop knowledge graphs in general and specific domains such as DBpedia, YAGO, LinkedGeoData, and Wikidata and they have been served several fields of real world applications from semantic parsing and named entity disambiguation to information extraction and question answering. These knowledge graphs contain millions of facts about entities. However, these knowledge graphs are far from complete and mandate continuous enrichment and enhancement. One possible approach to enhance KG is integrating knowledge from various knowledge graphs based on their aligned information. In this thesis, we develop new effective methods to find aligned entities from different KGs first and later enrich the KGs by enhancing their attributes.

We start this thesis by presenting techniques for entity alignment. The task of entity alignment is to find entities in two heterogeneous knowledge graphs that represent the same real-world entity. Many knowledge graphs have been created separately for certain purposes with overlapping entity coverage. These knowledge graphs are complementary to each other in terms of completeness. Unfortunately, only a fraction of the entities stored in different KGs are aligned. We present an embedding-based entity alignment method that finds entity alignment by estimating the similarities between entity embeddings. Existing methods mainly focus on relational structures and attribute information for the alignment process. Such methods fail when the entities have a limited number of attributes or when the relational structure couldn't capture the meaningful representation of the entities. To overcome this problem,

we propose EASAE, an Entity Alignment method using Summary and Attribute Embeddings. We utilize the entity summary information available in KGs for entities' summary embedding by employing BERT. Our model learns the representations of entities by using relational triples, attribute triples, and summary as well. Extensive experiments show that entity summary exhibit useful semantic information in entity alignment task. Our proposed approach outperforms the concurrent state-of-the-art alignment models.

To enrich KG by enhancing their attributes, we propose an Attribute Enhancement Framework (AEF) that integrates multiple KGs based on their aligned information. Typically, similar entities from different KG contain a different set of attributes. AEF exploits a representation learning based ranking model to discover the significant attributes from reference KG. Later it employs a similarity mapping technique to integrate new attributes into the target KG. AEF also determines the attribute value inconsistency between two KGs. With this study, we aim to include all the essential attributes of the existing KGs towards a more robust and complete knowledge graph. The results of the attribute enhancement work indicate important directions for future work.

Contents

List of Figures

List of Tables

1 Introduction

Knowledge is the core power in the age of data and information and incorporating the knowledge in a graphical representation is known as Knowledge Graph (KG). It facilitates the complex process of searching and exploration as a lot of information is in the form of data and context about an entity or object. We intend to enrich and enhance KG, which in turn helps to represent human knowledge into structured models that plays a vital role in the machine learning domain. We discuss the motivation, problem statement, and contributions of the thesis, in this chapter.

1.1 Motivation

The knowledge graph is a structured representation of real-world knowledge consisting of entities, relationships, attributes, and textual descriptions. An entity or instance is an object in the real world; a relationship describes the interaction and relation between two entities; an attribute describes the property of an entity; and a textual description includes the entity abstract, short summary, string information, etc.

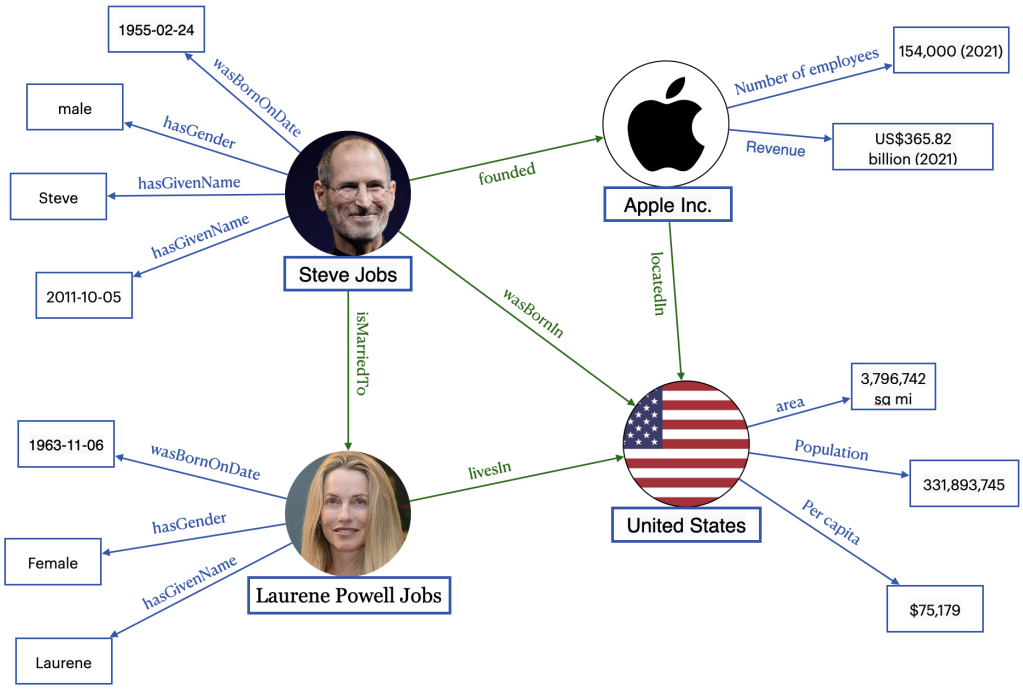


Figure 1.1: Knowledge Graph

Knowledge graph is modeled by a graph structure and facts are mainly represented in triple format. A triple consists of a subject, a predicate/relation, and an object where the predicate/relation indicates the relationship between an entity as the subject and the other entity (or literal) as the object. A relational triple can be denoted as (h, r, t) where h and t are the head entity and the tail entity, respectively, and r is the relation between the h and t. If the object is literal, then we call it attribute triple. Figure 1.1 shows the example of relation and attribute in a typical KG scenario.

There has been a surge of research and development on KGs for several decades due to their effective role in storing and representing knowledge and facts. Several well-known KGs can be found on the web, including open-source ones such as DBpedia [6], Freebase [7], YAGO [8], as well as commercial ones such as those developed by Google [9] and Microsoft [10]. In the last few years, there has been an explosive growth of interest in KGs in both the research community and the industry due to their indispensable role in AI applications such as natural language processing (including

dialogue systems/chatbots, question answering, sentence generation, etc.), search engines [11], recommendation systems, and information extraction.[3, 9]

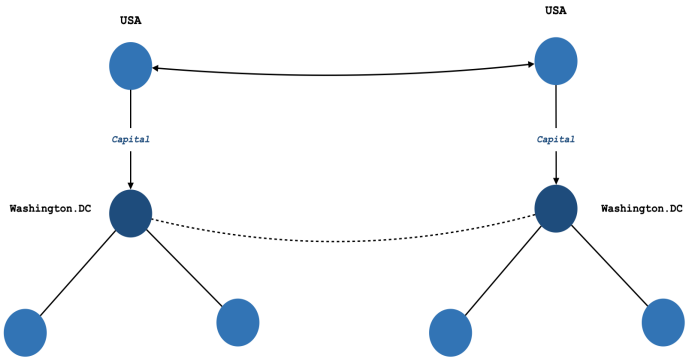
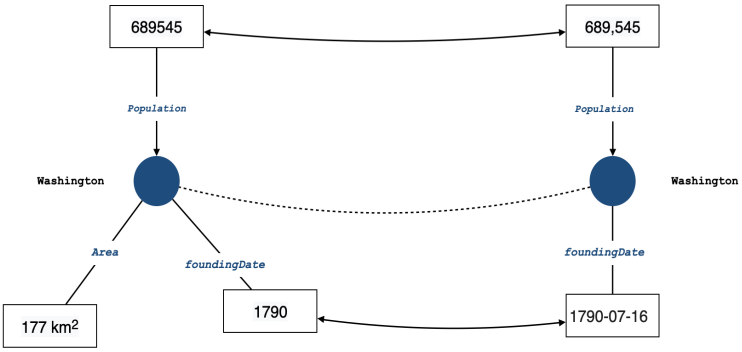
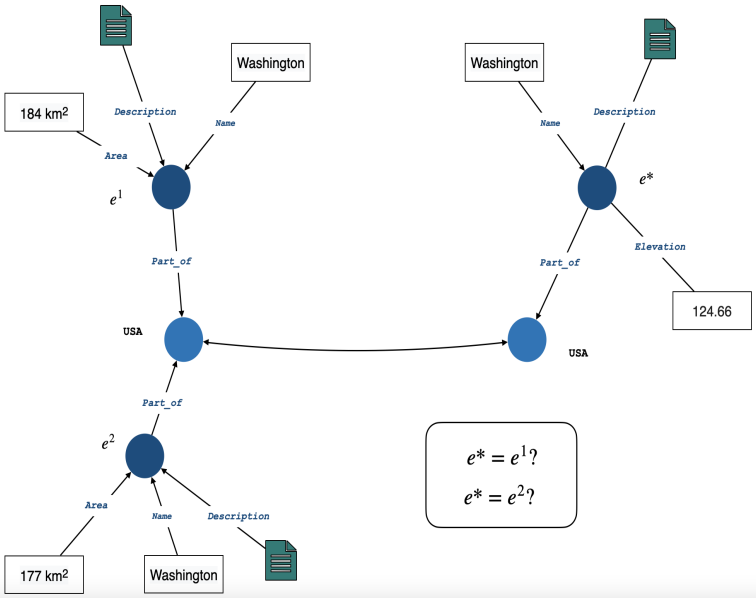
KGs are mostly constructed based on crowd-sourced content and automatic extraction methods; therefore, they are not always complete and error-free. Also, different KGs are initially constructed independently to serve various purposes, and they are heterogeneous by their structure. Integrating multiple KGs can complement each other toward a more complete and robust knowledge representation which requires KG alignment process. As a result, KG alignment and knowledge enhancement task become very prominent research area.

Over the last decade, several knowledge graphs have been created though most of them have significant overlapping entity coverage. These knowledge graphs e.g. DBpedia, YAGO, Wikidata are complementary to each other in terms of completeness. This study aims to integrate facts that belong to different knowledge graphs to form more robust knowledge graphs. Before we can incorporate multiple KGs, it is required to align them first. Aligning multiple KGs means we have to align the overlapping entities. Entity alignment task refers to finding the same real-world entities in multiple KGs.

The existing entity alignment models mostly depend on relational and attribute triples. This is because the neighbors of two equivalent entities in KGs usually contain equivalent entities and two equivalent entities often share similar attributes and values in KGs. However, these neighboring assumptions become inadequate when we have to deal with 1-N or N-N relations. Also, entity pairs from two KG will not always share equivalent attributes. It is very common for different KGs entities to contain a different set of attributes. Figure 1.1 demonstrates the phenomenon more clearly. In the first scenario, the 'Capital' relation is a 1-1 relation, and it is synced with both KGs. Therefore relation assumption is enough for the alignment task. Similarly when both KG contains a similar set of attributes with consistent attribute values attribute assumption is also adequate. However, in the third scenario, we see 1-N relation 'Part_of' and a different set of attributes in both KGs. So neither relational nor attribute property is enough for alignment. To deal with such issues, we propose a method that includes the textual description of entities assuming two equivalent entities should share a similar description.

Recently embedding-based approaches are popular for entity alignment task. Such

Table 1.1: Entity Alignment Principle

Relation Assumption the neighbors of two equivalent entities in KGs usually contain equivalent entities	
Attribute Assumption Two equivalent entities often share similar attributes and values in KGs.	
Summary Assumption Two equivalent entities should share similar description.	

models are built on top of a graph embedding model, which learns entity embeddings that capture the similarity between entities in a knowledge graph based on the relationship triples in a KG. To adapt the KG embedding for entity alignment between two KGs, the embedding-based models require both predicate and entity embeddings of two KGs to fall in the same vector space. Exiting models mostly rely on large numbers of seed alignments for this. However, the seed alignments between two KGs are rarely available, and hence are difficult to obtain due to the expensive human efforts required. We have exploited predicate alignment to ease the situation. We propose a novel model for embedding-based entity alignment which utilizes entity summary extracted from the KGs along with the relational and the attribute triples. Summary, relation, and attribute embeddings are jointly optimized to improve the alignment performance.

The goal of this research work is to integrate multiple KG towards a more complete and robust KG. Technically, when the KGs are aligned, they can complement each other both in terms of relations and attributes. However, traditional KG completion techniques are quite popular for solving relation completeness problem. Several probabilistic, embedding, and deep learning models are generally used for the completion of relations. Completion of attributes, on the other hand, is a more complex task. It is often challenging to detect the existence of missing attributes. Even if we can discover the missing attributes, the classification method may be challenging to apply widely due to the different hierarchical structures of different entities. Exiting models on attribute completion focuses on extracting attributes from the text description or source associated with specific entities mostly. This process works well when missing or incomplete attributes occurs due to inadequate data extraction method. But the issue still remains if the data source is not complete or the overall structure of KG lacks many important properties. In such cases, incorporating multiple KGs for attribute completion can be very useful. So we have proposed an attribute-enhancing framework that attempts to enrich the attributes of entities by integrating multiple KGs based on their aligned information. The task is done by detecting significant attributes from the reference KG and exporting them to corresponding entities from the target KG if the target KG is missing that information.

In this dissertation, we aim to incorporate multiple heterogeneous knowledge graphs, which in terms, lead us toward more complete KGs. We address the problem of entity alignment which is a vital process for KG integration and propose some

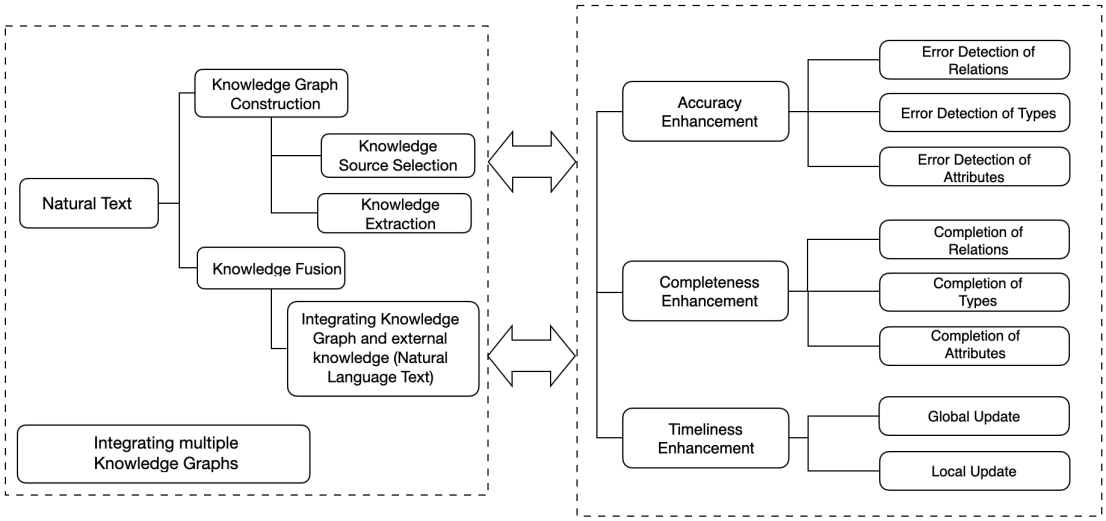


Figure 1.2: Knowledge Graph

techniques to tackle that task. We also discuss the knowledge integration processes and how we can solve them with a simple solution.

1.2 Problem Statement

Enhancing Knowledge graphs can be done either from natural text sources during or after construction, or they can be enhanced by integrating knowledge (relation, attribute, or type) from other Knowledge graphs (see figure 1.2). Our research mainly focuses on integrating knowledge from various KGs which can help overall knowledge graph completion and enhancement. In this paper, we try to solve the following two problems:

Knowledge graph alignment problem

In knowledge graph alignment, the main problem to integrate KGs is identifying the entities in different KGs that denote the same real-world entity. Entity alignment between KGs is a very challenging task in many aspects. Models should characterize the heterogeneous structures of different knowledge graphs and then capture the equivalent correspondence of entities them. Embedding-based alignment models rely

on large numbers of seed sets of aligned triples from two KGs to create unified vector space to compute the transition matrix. However, the seed alignments between two KGs are rarely available, and hence are difficult to obtain due to the expensive human efforts required.

The diversity between independently-created KGs makes the alignment task very challenging. Synonymous entity pairs in different KG usually contains a different set of relations and attributes. For entity alignment task this is one of the biggest challenges that we face.

Zero-shot case indicates the scenario where at least one of the entities in test triples is not present in the training sample. Embedding and GNN-based methods always suffer when they have to handle zero-shot scenarios. Popular entity alignment method based on these models usually ignores zero-shot scenario and highly depend on structure-based representations, so those models cannot deal with this situation because they have no representations for entities that are not present in training data. Another challenge is dealing with entities with no match alignment across them, where we have to rely on attributes or descriptions for entity alignment. For the above-mentioned reasons, proposing an ideal entity alignment method considering the challenges might be more complicated than the theoretical aspects.

Knowledge Enhancement Problem

knowledge integration or enhancement between KGs is crucial while handling the heterogeneous nature of KGs. Aligned entities often contain different sets of relations and attributes. However, traditional KG completion methods work quite well for relation enhancement tasks but there is no defined method for attributes. Attribute enhancement is a very complicated task. First, it is often difficult to detect whether an entity is missing any attributes or values. Suppose, for a person type entity we can assume that it should contain an attribute similar to *birthdate*. However, we cannot claim that it may have marriage date or death date information. A living person will not have the property *deathdate* or likewise for an unmarried person entity. But at the same time, it is possible that the entity is actually referring to a death person who was married as well and that information is missing from the KG. Second, after discovering the missing attributes, the classification method may be challenging to apply widely

Table 1.2: Research Problems and Proposed Models

Research Problems	Proposed Models
Knowledge Graph Alignment Problem	EASAE
Knowledge Enhancement Problem	AEF

due to the heterogeneous hierarchical structures of different entities. Finally, the completion of numerical attributes may require the adaptation of regression methods under some accuracy requirements. By using aligned entities from multiple KGs we can get an idea of the attribute which might be missing. Also, we can get a tentative idea for numerical values or inconsistency in attribute values if exists. We want to leverage this idea in this study.

1.3 Thesis Contributions

In this dissertation, we tackled several tasks connected to knowledge graph alignment and attribute enhancement. We proposed an effective model to address these research problems. The problem and its corresponding framework are listed in Table 1.2. In the following, we give a succinct overview of the models we proposed, and the contributions we made. We also mention peer-reviewed articles published along with our research work.

Entity Alignment - EASAE

Leveraging the KG embedding techniques for addressing the entity alignment problem has drawn increased attention recently. we propose a novel embedding-based model EASAE [12] for entity alignment. EASAE exploits entity summary extracted from the KGs along with the relational and attribute triples. We propose EASAE model that consists of three components: entity summary embedding, relational embedding, and attribute embedding. We also introduce a weighted average technique to combine them. We exploit BERT [2] to learn the summary embeddings in EASAE and the translation-based model for relational and attribute embeddings. All three components are jointly optimized to improve the alignment performance. EASAE is an ensemble

method that incorporated entity summary as part of a symbolic system and embeddings as a sub-symbolic system. We apply EASAE framework with benchmark datasets and find very impressive performance. EASAE also shows effectiveness where attribute triples are limited and zero-shot scenarios.

Attribute Enhancement - AEF

Attribute property in a knowledge graph plays an important role to describe the properties of an entity and attribute completion is a very crucial task for KG completion and quality control. However, only a handful of work has been done in this field because of its complexity. We propose an Attribute Enhacement Framework (AEF) for entity attribute property enrichment based on significant attribute ranking and missing value inclusion using multiple KGs. KG is created by extracting data from multi-sourced unstructured or semi-structured information. So accuracy in knowledge cannot be guaranteed all the time. This is the reason we always experience missing values or inconsistency in data. Combining the information in different KGs can enrich the quality of the knowledge graph though it is a very challenging task. In this paper, we propose a method to rank entity attributes based on their importance of a reference KG, and based on that we can include new or missing attributes to the targeted KG. Our ranking method sorts the attributes based on importance and then adds the top-ranked attributes to the specific entity if they are missing in the target KG.

We propose embedding and probabilistic approaches to tackle the ranking problem and later prepare an ensemble method to rank the important attributes. AEF address two downstream tasks: 1) New or missing attribute proposing from reference KG to targeted KG; 2) Inconsistency detection of attributes. The first task utilizes both the ranking method and similarity match function together to recommend new or missing attributes. While the second task only leverages the similarity match function to detect the value inconsistency between two KGs.

1.4 Thesis Outline

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

- **Chapter 2**

  This chapter presents the fundamentals of knowledge graph and its concept and knowledge representation. Then, the related work on KG embedding, KG alignment, and attribute enhancement are reviewed and discussed respectively. Later, we further discussed the natural language models that frequently uses in the KG alignment and enhancement task. Finally, we discussed and identified the remaining issue before concluding the chapter.

- **Chapter 3**

  This chapter proposes, a joint entity summary and attribute embedding model along with relational structural embedding technique for entity alignment in between KG. Our proposed model uses the BERT embedding model for entity summary embeddings, and it is very effective in zero-shot scenarios. . Extensive experiments demonstrated that our approach achieved superior results than the baseline embedding approaches.

- **Chapter 4**

  In this chapter, we describe the attribute enhancement model using multiple knowledge graphs toward attribute completion. We discuss several approaches for attribute ranking for reference KG. We show how to incorporate these ranked attributes to target KG. We also discuss similar attribute mapping methods to avoid importing already existing attributes. We discuss about the number of new and missing attributes for the target KG in the process of attribute enhancement.

- **Chapter 5**

  This chapter discusses the achievement we accomplished by the proposed entity alignment framework EASAE and attribute enhancement framework AEF. Then, the scope, the limitation and the discussion of each framework is presented respectively.

- **Chapter 6**

  This chapter summarizes the thesis's contributions and outlines future work directions.

2

Fundamentals and Related Works

In this Chapter, we presented fundamentals of KG and KG embedding models in section 2.1 and 2.2 . Then we discuss some NLP methods that are widely used an for KG alignment and enhancement field in section 2.3. After that, the KG alignment and their related works are discussed in Section 2.4. Next, the survey on KG enhancement was reviewed and discussed in Section 2.5. In the following sub-section (Section 2.4.2). Finally, We discussed the summary of this chapter in Section 2.6.

2.1 Knowledge Graph

A knowledge graph is a network of real-world objects, events, situations, or concepts—and illustrates which are commonly referred as entities and the relationship between them.This information is usually stored in a graph database and visualized as a graph structure, prompting the term knowledge “graph.”

A knowledge graph is made up of three main components: nodes, edges, and labels. Any object, place, or person can be a node. An edge defines the relationship between

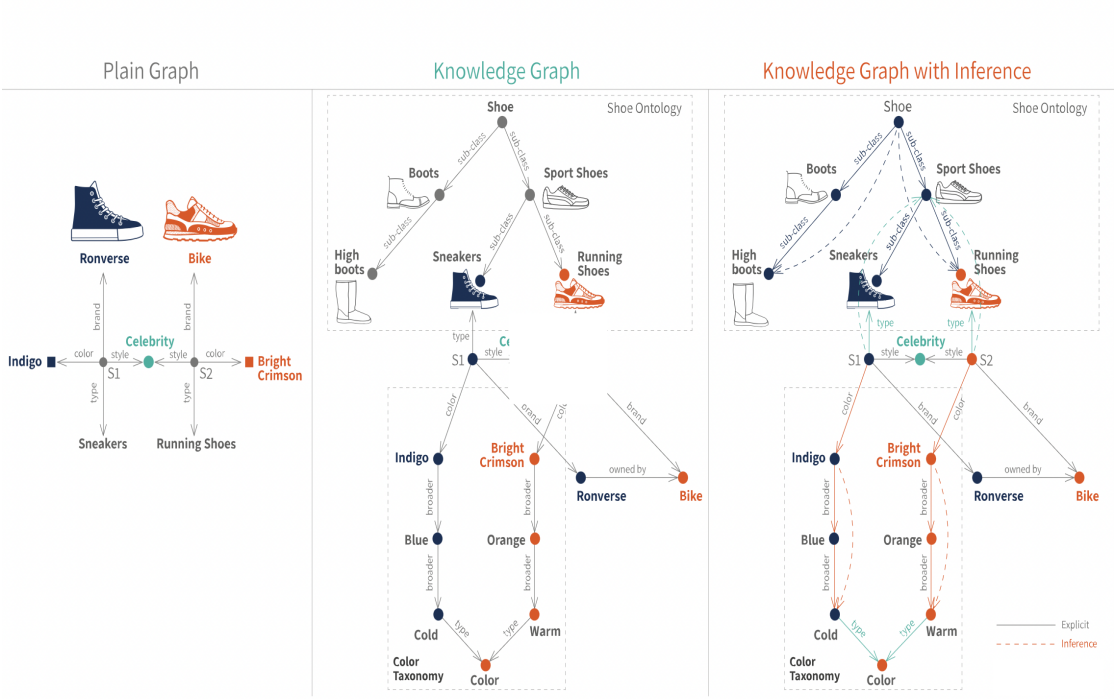


Figure 2.1: Knowledge Graph Concept

the nodes. Its definition can be formalized in the following definition.

Definition 1 *Knowledge Graph (KG): A knowledge graph KG = (E, R, T) , where E, R, T are the set of entities, relations and triples respectively.*

- ***Entity*** *is a vertex or a node in KG, which represents a unique real-world object. Note that, a description of the entity, referred as literal, can also be a node in KG.*
- ***Relation*** *is an edge with the label in KG, which expresses the relationship between entities or between an entity and its description.*
- ***Triples*** *is an RDF format consists of a subject, a predicate/relation and an object.*

2.1.1 Knowledge Graph Concept

Knowledgegraphs are usually built from various data sources, which normally vary in structure. These diverse data get structured using schemas (provide the framework

for the knowledge graph), identities (classify the underlying nodes appropriately), and context (define the setting for the knowledge). These components help separate words with multiple meanings, like distinguishing the difference between Apple, the brand, and apple, the fruit. knowledge graphs can also support the creation of new knowledge, establishing connections between data points that may not have been realized previously.

Figure 2.1 represent classic difference between graph and knowledge graph and the power of inference using knowledge graph. In recent years, knowledge graphs have been applied in various ways. For example, the most direct application is Question Answering [10–14]. This is a method of searching desired information in a knowledge graph by converting a question into a machine readable query. In the medical domain, knowledge graphs can help doctors to gather health data and diagnose disease [15–17]. In addition, knowledge graph is applied for content tagging, fact checking, and so on [3–5, 18–22].

2.1.2 Knowledge Graph Representation

Knowledge Graph is represented by the Linked Data concept [13]. In the Linked Data concept, the Resource Description Framework (RDF) is a standard model for publishing Linked Data [5]. It uses to describe the information and their relations and also make data become interchangeable [5]. The specification of RDF is introduced by W3C Recommendation (RDF 1.1 Concepts and Abstract Syntax) [27]. In the RDF specification, the key concept is a RDF graph, which is a collection of RDF triples. A RDF triple consists of a subject, a predicate and an object. A subject and an object are treat as a vertex, while a predicate is considered as a relation. The direction of the edge is used to identify which vertex is the subject and which vertex is the object. The edge of a RDF triple points out from the subject and points into the object. An RDF triple therefore can be viewed as a directed graph, which composes of two vertices and one directed edge, as shown fig2.2.

Furthermore, the RDF specification [27] defines three kinds of resource representations: Internationalized Resource Identifier, literal and blank node. Such representations are used to describe an element of a RDF triple. Note that, based upon our observation on many KGs, a blank node is usually ignored because the blank node does not provide

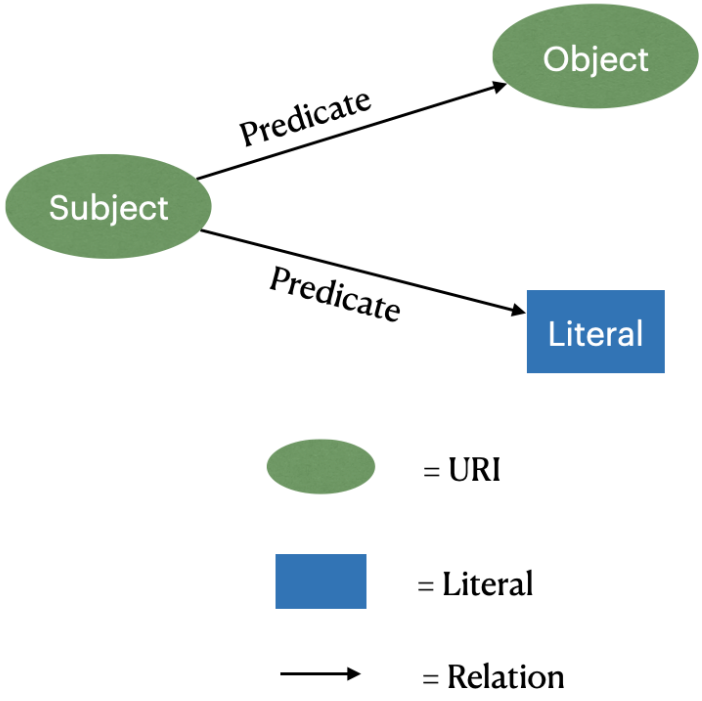


Figure 2.2: Knowledge Graph Enhancement

any meaning and makes representation of a KG become more complicated.

Unique Resource Identifier (URI) is a unicode string that comply the RFC standard and it uses to identify a resource as a unique identifier so that the resource could be referred [27]. wider range of the encoding characters [27].

Literal is a string used to identify a value. There are two types of literal: untyped and typed. Untyped literal is a string without identify type of the literal; it can be any string, e.g. “Tom”, “November 2022”, “3.23”. Typed literal is a string with the datatype can be identified. The datatype are string types such as language number, digit, date, etc. In order to identify the datatype of typed literal, the extension of the markup are used. As Example, "Tom Hanks"@en represents Tom Hanks as in English, "2022-11-01"^^xsd:date represents the date on 1st November 2022.

Due to the characteristic of a RDF triple as shown in Figure fig2.2 , a resource representation that can be used to describe each element of a RDF triple, is therefore

depended upon the position, which is a subject or a predicate or an object, of the element.

- **Subject** : The subject can be represented only by a URI. Since the subject is the entity. it need to be identified as a unique resource, which is only represented by URI.

- **Predicate** : The predicate is also expressed by a URI. As shown in Figure 2.1, a predicate is a edge with its label. Different relations therefore can be expressed by different types of labelled edges.

- **Object** : The object can be a URI or literal. In contrast with the subject, an object is a information that fulfills the relation with its subject. Therefore, object allows the representation as URI or Literal. If it is URI, it express the relation between entities, subject and object. In case of literal, it describes the detail for subject, known as its description

Based on the characteristic of object described by the RDF definition Knowledge graph triples can be categorized by two types.

- **Relational Triples** depict relationship between two entities

- **Attribute Triples** link entities with data values - string, number, or date

Throughout this thesis paper, we have shown the significance of these category of triples.

2.1.3 Popular Knowledge Graph

The knowledge graphs also refer as Linked Open Data (LOD) [13] because anyone can access those knowledge graphs online and freely. Numerous KGs can be found over the web on various domains. (Fig. 2.3). In the following sections, we will briefly discuss about some of the most popular KGs.

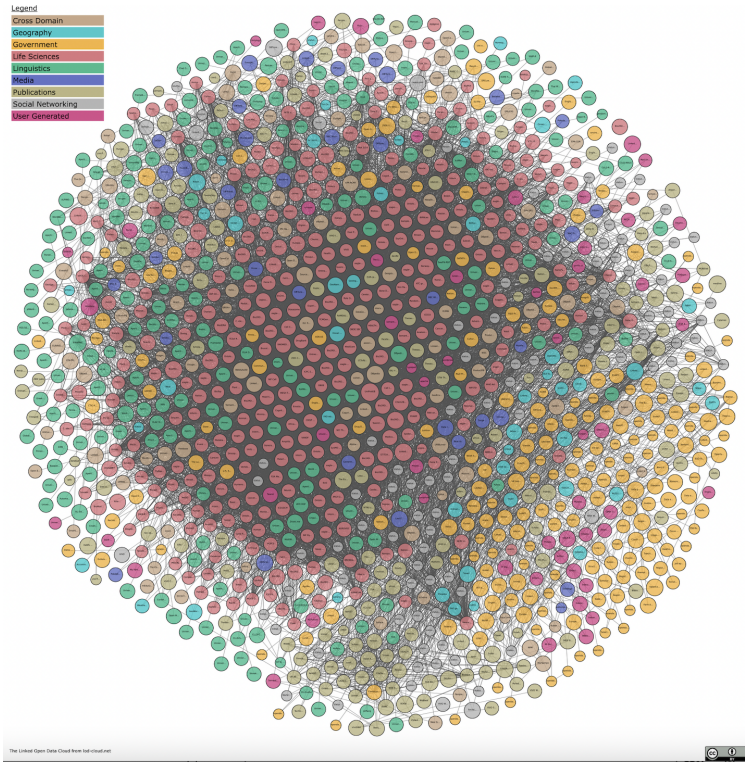


Figure 2.3: Knowledge Graphs in the Linked Open Data Cloud on November 3rd, 2022

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 [14]. Wikidata is a collection of entity pages. Entity pages are of two types: 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[1].

[1]https://wikidata.org

DBpedia

In recent years Wikipedia[2] has become an essential source of information to build knowledge graphs. The DBpedia[3] project extracts the structured information that is incorporated 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 [15]. This extraction process generates one of the most popular knowledge graphs publicly available that, at the time of writing, contains more than 17M entities and as of June 2021, it contains over 850 million triples. DBpedia [6] 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, locations, organizations, and creative works such as books, pieces of art, movies, and music.

YAGO

YAGO [8] also launched in the year of 2007 which shares with DBpedia [6] some core ideas but strives at integrating Wikipedia and WordNet[4], a very popular lexicon for the English language [16].

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 from Wikipedia or GeoNames, an extensive 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.

YAGO3 is a huge semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Currently, YAGO3 has knowledge of more than 10 million entities (like

[2] http://wikipedia.org
[3] http://dbpedia.org
[4] https://wordnet.princeton.edu/

persons, organizations, cities, etc.) and contains more than 120 million facts about these entities.

One of the key 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 [8]. This is due to the fact that DBpedia's type hierarchy was created manually, while 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.

Freebase

Freebase was a large collaborative knowledge base launched in 2007 as well. Google took it over in 2010 [7]. 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 [14]. 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. 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 [17]. 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[5].

Google's Knowledge Graph and Knowledge Vault

Nowadays in Google Search we can find info boxes with information about people, places, and things. Infoboxes are designed to help end users quickly understand more

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

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. Google 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 [18].

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 [19, 18].

2.2 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

Table 2.1: Knowledge Graph Embedding

Embedding Models	Methods
Translation-based	TransE, TransH, and TransR, CTransR, TransD, TorusE, RotatE RotatE, HyperKG
Bilinear	RESCAL, DistMult, ComplEx , TuckER
Neural Network-based	NTM, NAM, ConvE, ConvR, InteractE

scoring function of the model under consideration. We can divide KG embedding models typically into three groups as shown in Table2.1. In the following sections we will present some discussion on these methods.

2.2.1 Translation-Distanced based Models

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

$$\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t} \tag{2.1}$$

where **h**, **r**, and **t** are the embeddings of h, r, and t of a triple (h, r, t), respectively. The principle is to obtain distributed representations of words from a text and in which the differences between word representations often represent their relation.

TransE

TransE[20] is the first translation-based model, which embeds entities and relations in a real vector space. TransE employs the following score function:

$$|| \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} || \tag{2.2}$$

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

TransH

Many researchers [21, 22] 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 [22], which represents relations by hyperplanes. TransH projects entities on the hyperplane corresponding to a relation. A single entity can have different representations on different hyperplanes. It 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:

$$\begin{aligned} \boldsymbol{h}_\perp &= \boldsymbol{h} - \boldsymbol{w}_r^\top \boldsymbol{h} \boldsymbol{w}_r, \\ \boldsymbol{t}_\perp &= \boldsymbol{t} - \boldsymbol{w}_r^\top \boldsymbol{t} \boldsymbol{w}_r. \end{aligned} \tag{2.3}$$

The scoring function is very similar to TransE:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h}_\perp + \boldsymbol{r} - \boldsymbol{t}_\perp ||_{l_{1/2}}. \tag{2.4}$$

TransR/CTransR

TransR [21] also handled the shortcomings of TransE, but in a slightly different way in comparison to TransH. It 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:

$$\boldsymbol{h}_r = \boldsymbol{M}_r \boldsymbol{h}, \quad \boldsymbol{t}_r = \boldsymbol{M}_r \boldsymbol{t}, \tag{2.5}$$

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:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = || \boldsymbol{h}_r + \boldsymbol{r} - \boldsymbol{t}_r ||_{l_{1/2}}. \tag{2.6}$$

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 [23] 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:

$$\begin{aligned} \boldsymbol{M}_{rh} &= \boldsymbol{r}_d \boldsymbol{h}_d{}^\top + \boldsymbol{I}, \\ \boldsymbol{M}_{rt} &= \boldsymbol{r}_d \boldsymbol{t}_d{}^\top + \boldsymbol{I}. \end{aligned} \tag{2.7}$$

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.,

$$\hat{\boldsymbol{h}} = \boldsymbol{M}_{rh}\boldsymbol{h}, \quad \hat{\boldsymbol{t}} = \boldsymbol{M}_{rt}\boldsymbol{t}, \tag{2.8}$$

TorusE

TorusE [24] 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 [25], 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.

2.2.2 Bilinear Models

RESCAL

For the link-prediction and triple-classification tasks, bilinear and neural-network-based models are also popular. RESCAL[26, 27] 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 [28] 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 [29] 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

TuckER [30], is a relatively straightforward but powerful linear model based on Tucker decomposition of the binary tensor representation of knowledge graph triples. TuckER gained popularity for its fully expressive feature. In this model, 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 [26], DistMult [28], and ComplEx [29], are special cases of TuckER.

2.2.3 Neural Network based Models

NTN

Neural Tensor Network (NTN) [31] has a standard linear neural network structure and a bilinear tensor structure. It concatenates head and tail entities as an input layer to the

nonlinear hidden neural layer and has the scoring function: This can be considered as a generalization of RESCAL. The weight of the network is trained for each relation.

NAM

Neural Association Model (NAM) [32] conducts semantic matching with a Deep Neural Network (DNN) architecture different from other neural network-based models. A deep neural network is used to compose embeddings of the head entity and the relation for a triple (h, r, t). Then, the output vector is used to take the inner product with the embedding of the tail entity to score the triple.

ConvE and ConvR

In recent times, several convolutional models have been proposed for solving link prediction task. Dettmers et al. [33] 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 [34] 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 [35] 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.

Bilinear models add more redundancy than translation-distance-based models. therefore, they are prone to have an overfitting problem. 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. However, they add more time and space complexity.

2.3 Natural Language Processing

2.3.1 Word2vec

Word embedding actually refers to numerical representation of words. We commonly use similar for colors in form of RGB. Word2Vec basically means expressing each word in your text corpus in an N-dimensional space often refereed as embedding space. The simplest word embedding can be done by using one-hot vectors. If the word corpus contains 10,000 words as vocabulary, then one-hot encoding can represent each word as a 1x10,000 vector. The reason of choosing one-hot encoding is due to simplicity, robustness and observation that simple models trained on huge amounts of data outperform complex systems trained on less data [36]. The Word2vec model captures both syntactic and semantic similarities between the words[6]. One of the well known examples of the vector algebraic on the trained word2vec vectors is Vector(“France”)-Vector(“Pais”)= Vector(“Tokyo”)-Vector(“Japan").

Word2Vec is a predictive embedding model. Predictive models learn their vectors in order to improve their predictive ability of a loss such as the loss of predicting the vector for a target word from the vectors of the surrounding context words. There are two main Word2Vec architectures that are used to produce a distributed representation of words: 1) Continuous Bag of Words (CBOW) and Skip gram. We will discuss about them in detail in following sections.

Continous Bag of Words (CBOW)

In the continuous bag of word (CBOW) model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. When there is only one word per context, the model will predict one target word given one context word. Figure 2.4 shows the network model with only one word in the context. Here, the vocabulary size is V, and the hidden layer size is N. The units on adjacent

[6] https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30

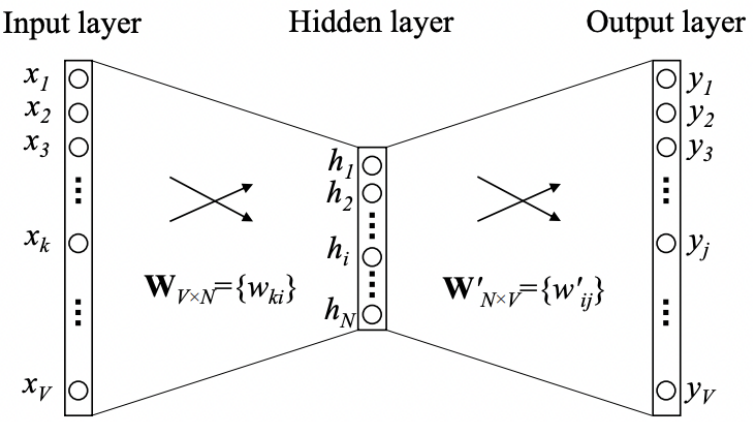


Figure 2.4: A simple CBOW model with only one word in the context [1]

layers are fully connected. The input is a one-hot encoded vector, which means for a given input context word, only one out of V units, x1, , x_V , will be 1, and all other units are 0. Figure 2.5 shows the CBOW model with a multi-word context setting. When computing the hidden layer output, instead of directly copying the input vector of the input context word, the CBOW model takes the average of the vectors of the input context words, and use the product of the input→hidden weight matrix and the average vector as the output [1]. Total weights involved in training CBOW model are N×D+D×log(2)V.

Skip-gram

Skip-gram model, the distributed representation of the input word is used to predict the context. Figure 2.5 shows the Skip-gram model. It is the opposite of the CBOW model. The target word is now at the input layer, and the context words are on the output layer. Skip-gram works well with a small amount of the training data and represents well even rare words or phrases. CBOW is several times faster to train than the skip-gram, with slightly better accuracy for the frequent words. The total complexity of the model is N×D+N×D×log2(V). Noticeably, N also gets multiplied to

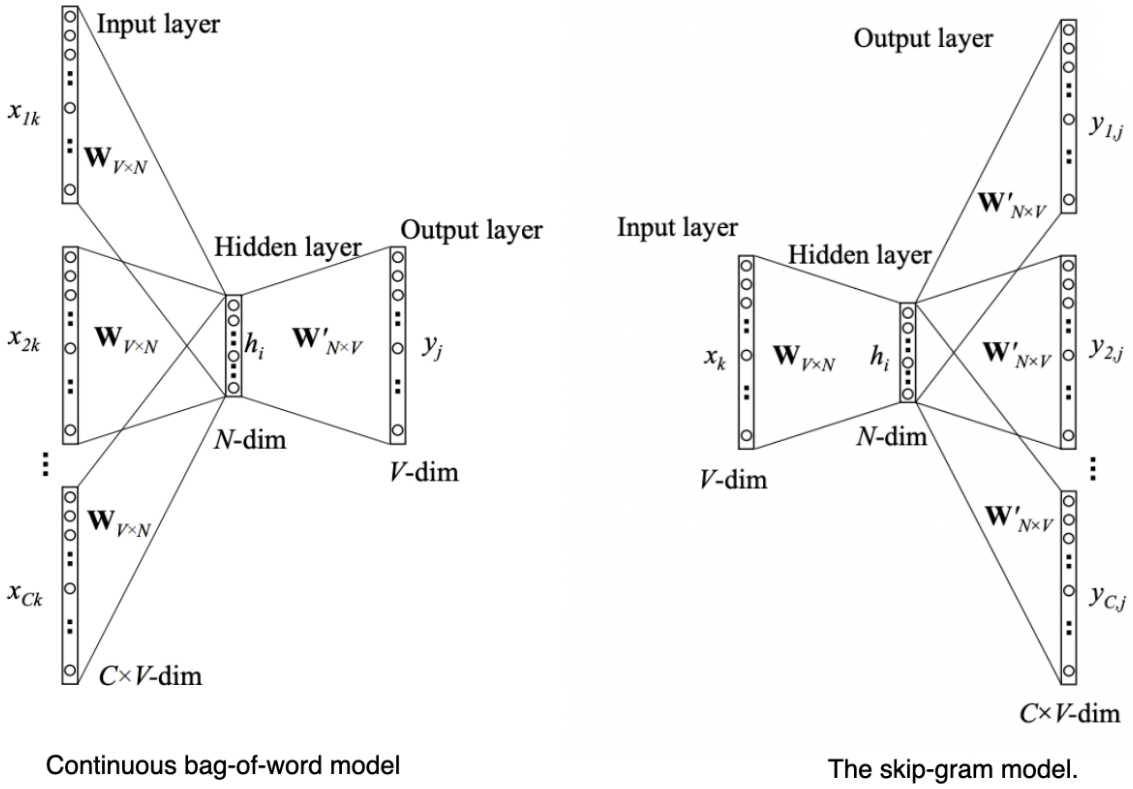


Figure 2.5: Continuous bag-of-word model and Skip-gram Model [1]

D×log2(V) term as its not a single class classification problem compared to CBOW, rather N class classification problem. Hence overall complexity of skip gram model is greater than the CBOW model[6].

2.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) has shown promising results in processing arbitrary sequences of input. For a given sequence of input $x_1, x_2, ..., x_n$, RNN model learns the current latent state with the input data at time t and the previous latent state at time $t-1$. Then the current latent state is used to predict the output. The RNN is derived as

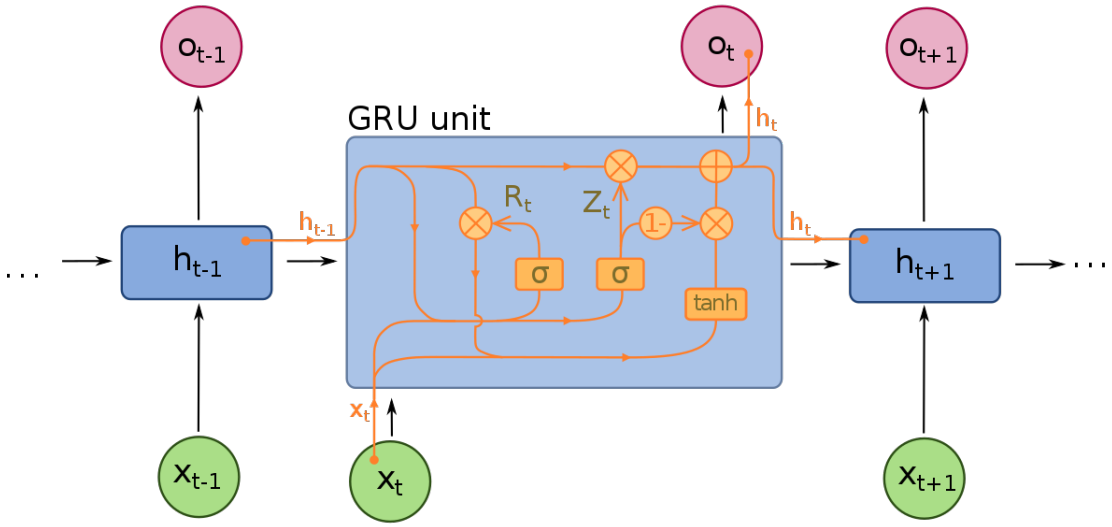


Figure 2.6: GRU (image collected from Wikipedia)

follows:

$$h_t = f(W_{i,h}x_t + W_{h,h}h_{t-1} + b_h)$$
$$y_t = g(W_{h,y}h_t + b_y)$$

where x_t is the input vector at time t, h_t is the vector of hidden layer at time t, y_t is the prediction vector at time t, $W_{i,h}$, $W_{h,h}$, $W_{h,y}$ are parameter matrices, b_h, b_y are the bias parameters for the network and f, g are the activation functions, e.g., sigmoids.

Although RNN is able to handle a variable-length sequence input, long-term dependencies are difficult to be captured due to the gradients that tend to either vanish or explode. The long short-term memory (LSTM) unit and gated recurrent unit (GRU) are able to handle long-term dependencies and perform better than using traditional *tanh* unit [37].

2.3.3 Gated Recurrent Unit

A gated recurrent unit (GRU) was proposed by Cho et al. to make each recurrent unit to adaptively capture dependencies of different time scales [37]. Similar to the LSTM unit, the GRU has gating units that modulate the flow of information inside the unit, however, without having a separate memory cells. The GRU is like a long short-term

memory (LSTM) [38] with a forget gate [39], GRU has fewer parameters than LSTM, as it doesn't have an output gate. GRU's performance on specific tasks of polyphonic music modeling, speech signal modeling and natural language processing is similar to LSTM models, but it has shown better performance on certain smaller and less frequent datasets.

There are several variations on the full gated unit, with gating done using the previous hidden state and the bias in various combinations, and a simplified form called minimal gated unit.

Fully gated unit is defined as follows:

$$
\begin{aligned}
z_t &= f(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= f(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= g(W_h x_t + U_h(r_t \circ s_{t-1}) + b_h) \\
s_t &= z_t \circ s_{t-1} + (1 - z_t) \circ h_t
\end{aligned}
$$

where x_t is the input vector at time t, z_t is the update gate vector at time t, r_t is the reset gate vector at time t, h_t is the hidden layer vector at time t, s_t is the output vector at time t, W and U are parameter matrices, b is bias parameter, f and g are activation functions, and $\circ$ is the Hadamard product operation. σ_g is the sigmoid and ϕ_h s a hyperbolic tangent activation function.

2.3.4 BERT

BERT (Bidirectional Encoder Representations from Transformers) [2] is a transformer-based machine learning technique for natural language processing pre-training developed by researchers at Google AI Language. When BERT was published, it achieved state-of-the-art performance on a wide variety of NLP tasks, including GLUE (General Language Understanding Evaluation) task, and Question Answering (SQuAD v1.1). BERT achieved remarkable performance in Natural Language Inference (MNLI), SWAG (Situations With Adversarial Generations), Sentiment Analysis, and others.

BERT's key innovation is applying the bidirectional training of the Transformer. For language modeling, Transformer is a popular attention model. But previously, text sequences were looked at either from left to right or combined with left-to-right and

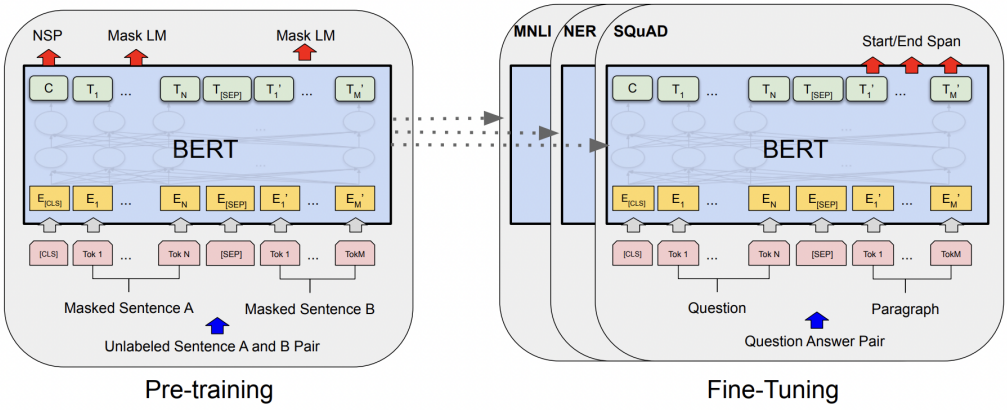


Figure 2.7: Overall pre-training and fine-tuning procedures for BERT [2]

right-to-left training. BERT shows that if the language model is bidirectionally trained, it can have a more profound sense of language context and flow, which is not possible in single-direction language models. BERT framework consists of two basic steps: pre-training and fine-tuning.

BERT Pre-training

During pre-training (see, Fig 2.7), the model is trained on unlabelled data over different pre-training tasks. BERT uses two unsupervised tasks for pre-training named as: Masked LM and Next Sentence Prediction (NSP).

Masked LM (MLM) For training, 15% of the words in each sequence is replaced with a [MASK] token before feeding into BERT. The model then tries to predict the original value of the masked words based on the context provided by the other words in the sequence.

Technically, the prediction of the output words requires [7] :

- Adding a classification layer on top of the encoder output.
- Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.

[7] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

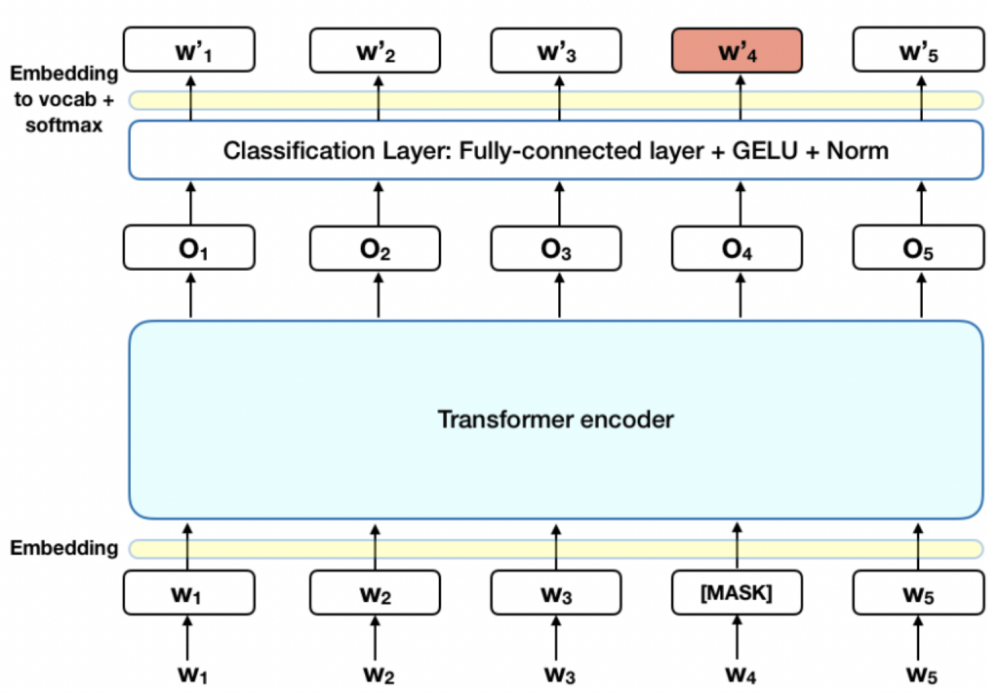


Figure 2.8: MaskLM [7]

- Calculating the probability of each word in the vocabulary with softmax.

Next Sentence Prediction (NSP)

Many important downstream NLP tasks are based on understanding the relationship between two sentences. This is not directly captured by previous language modeling. In understanding sentence relationships, BERT pre-trains for a binarized next sentence prediction (NSP) task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext) [2]. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2. A positional embedding is added to each token to indicate its position in the sequence[7].

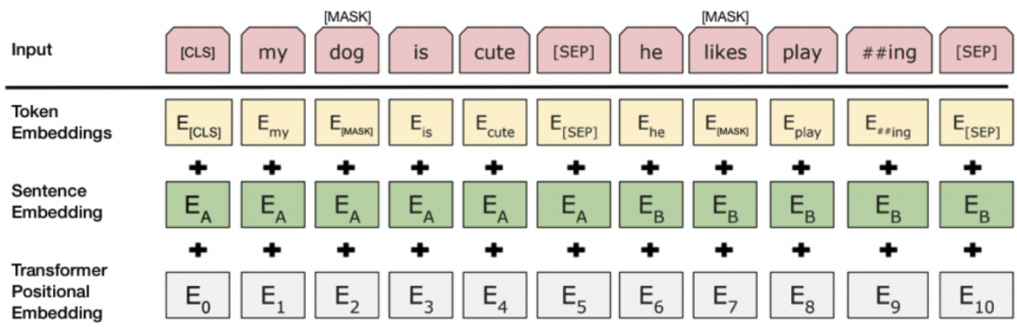


Figure 2.9: NSP [2]

BERT Fine-training

Fine-tuning is straightforward since the self attention mechanism in the Transformer allows BERT to model many downstream tasks. Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token. In Question Answering tasks, the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer[7]. Compared to pre-training, fine-tuning is relatively inexpensive.

2.4 Entity Alignment

Knowledge graphs are heterogeneous and complementary as they are often constructed for serving different purpose from multiple sources. So, merging heterogeneous knowledge from different data sources and languages into a unified and consistent knowledge graph is a feasible and necessary process. In order to marge different knowledge graphs that may complement each other more efficiently, Entity Alignment has become an important research field. The task of entity alignment refers to find equivalent entities in different knowledge graphs (KGs) that refer to the same real-world object. Given two knowledge graphs, i.e., a source KG and a target KG, the alignment is done by first identifying the similar entity in the source and target KGs and then expands to detect the synonymous pair between their neighbouring entities. Recently,

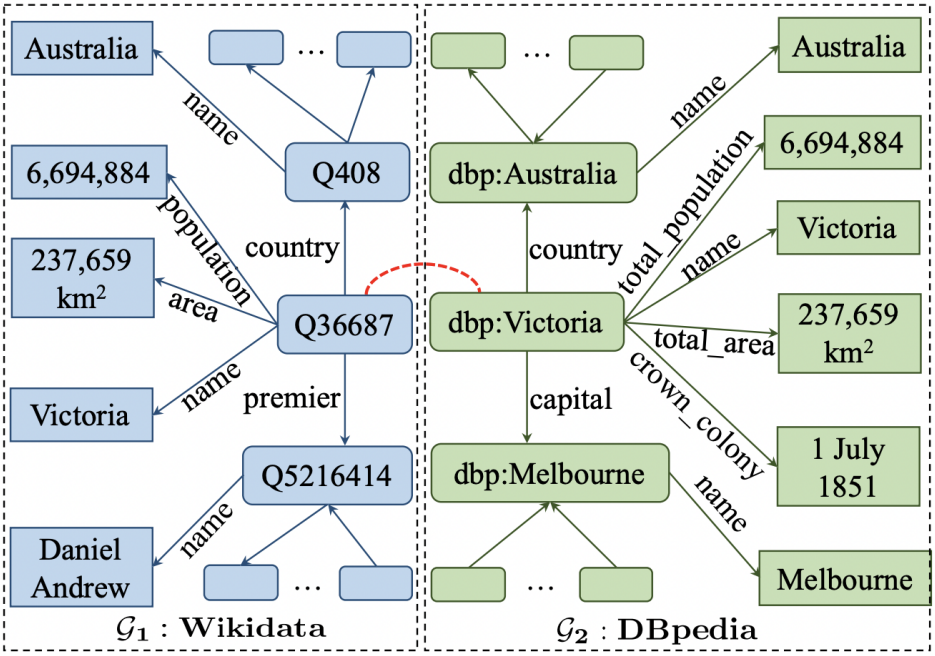


Figure 2.10: Entity Alignment Example [3]

various kind of model has been proposed for entity alignment. Although they are different in terms of procedure, typically they follow two major assumptions. One is relational assumption which means the neighbors of two equivalent entities in KGs usually contain equivalent entities. The second assumption is two aligned entities share similar attributes and values in KGs. These two assumptions are the basic principle for alignment task. However sometimes they are not enough to solve the problem. As shown in figure 1.1, it can be ambiguous if we have to deal with 1-N, N-N relations. Also similar entity pair can have different set of attributes which can be challenging for alignment task. To handle such problem in this thesis we propose a method which includes summary assumption. It includes textual description for entity alignment.

Recent approaches of entity alignment can be categorized in to three groups. The first is string-similarity-based approaches detailed in Section 2.3.1. The second is embedding-based approaches detailed in Section 2.3.2. and the third is GNN Based Alignment Models.

2.4.1 String Similarity based Entity Alignment.

Earlier entity alignment approaches basically use string similarity for alignment. Some techniques have focused on improving the effectiveness of the matching of entities via different entity similarity measures while others focus on the efficiency of entity matching. For example, models like RDF-AI [40], SILK [41], LD-Mapper [42], PARIS [43] fall into the first category while LIMES [44], HolisticEM [45] follows the second strategy. RDF-AI [40] implements an alignment framework that consists of pre-processing, matching, fusion, interlink, and post-processing modules. The pre-processing module inspects the ontology consistency and transforms properties into a standard form. For the matching module, RDF-AI exploits fuzzy string matching based on sequence alignment [46], word relation [16], and taxonomic similarity algorithms to compute a similarity score between entities in the source and target KBs. Based on this score, the interlinking module creates a temporary graph that contains entities and their properties from the source KG that correspond to entities in the target KG. The fusion module combines the target KG and the temporary graph from. Finally, the post-processor module verifies the resulting ontology consistency. SILK provides the Link Specification Language (LSL), which allows users to specify the similarity measures for comparing certain attributes [3]. In LSL, users can define the properties and the metrics to be used for entity similarity computation. SILK provides various similarity metrics, including string similarity, numeric similarity, date similarity, and URI equality. To reduce the number of comparisons between entities from two different KBs, SILK provides rough index pre-matching. All target resources are indexed based on the values of their properties. This index is used to look up potential matches for a given entity. LD-Mapper combines string similarity and entity nearest neighbors. PARIS includes schema matching (e.g., classes and sub-classes of entities) to compute the entity similarity.

On the other hand, LIMES utilizes triangle inequality to calculate similarities between entity pairs from the source and target KBs. First, it constructs clusters to group entities on the target KB. Then, the similarities between entities in the source KB and the generated clusters are calculated as an approximation. Using these approximations, LIMES avoids comparing every entity pair from the source and target KG, and thus speed-up the alignment process. Finally, the actual similarity between

entities from the source KB and entities in the corresponding cluster on the target KB is computed, and the entity pair with the highest actual string similarity is returned. HolisticEM [110] constructs a graph of potential entity pairs based on the overlapping attributes and the neighboring entities. From the constructed graph, the local and global properties are propagated using Personalized Page Rank to compute the actual similarity of entity pairs.

The string similarity approaches work well when the properties/relations to be compared between the entities are known. However, different knowledge graphs may use different property/relation names to store similar property values. It depends on the KG structure. Hence, these approaches rely on user defined rules to determine the comparable properties. The manually defined rules are not 100% correct because different entity types may contain a different set of properties. For example, properties such as birthdate and deathdate may exist in the set of properties of person type entities but not in the set of properties of location type entities.

2.4.2 Embedding-based Entity Alignment .

The embedding models for KG completion inspired several entity alignments based models. The KG embedding based entity alignment models represent different KGs as embeddings and find entity alignment by calculating the similarity between the embeddings. In this section we discuss the state-of-the-art KG embedding based entity alignment models.

MTransE

This model proposed by chen et al. [47] is the first multi-linguaal KG embedding based entity alignment model. It follows the principle of TransE. MTransE is a multilingual alignment model which consists of two components: the first component is the knowledge model that encodes the entities and relations from each language-specific graph structure, and the second component is the alignment model that learns the cross-lingual transitions from the existing seeds. This model aligned the entities of two different languages $\mathcal{L}_1, \mathcal{L}_2 \in \mathcal{L}$, where $\mathcal{L}$ is the set of all languages by considering two components: (1) knowledge model, which is actually TransE model and (2) alignment model consists of distance-based axis calibration model, translation vectors model,

and linear transformation model. In distance-based axis calibration they defined two different type of scoring functions. For $(h, r, t) \in \mathcal{L}1$ and $(h^{'}, r^{'}, t^{'}) \in \mathcal{L}2$ the scoring functions are as follows:

$$\begin{aligned} Z_1 &= || \mathbf{h} - \mathbf{h}^{'} || + || \mathbf{t} - \mathbf{t}^{'} || \\ Z_2 &= || \mathbf{h} - \mathbf{h}^{'} || + || \mathbf{r} - \mathbf{r}^{'} || + || \mathbf{t} - \mathbf{t}^{'} || \end{aligned} \tag{2.9}$$

where Z_1 denotes the correctly aligned multilingual expressions of the same entity tend to have close embedding vectors and Z_2 overlays the penalty of relation alignment to Z_1 to explicitly converge scores of the same relation.

Translation vectors model determines the cross-lingual transitions into vectors and the scoring function is defined as:

$$Z_3 = || \mathbf{h} + \mathbf{v}^e - \mathbf{h}^{'} || + || \mathbf{r} + \mathbf{v}^r - \mathbf{r}^{'} || + || \mathbf{t} + \mathbf{v}^e - \mathbf{t}^{'} || \tag{2.10}$$

where v^e and v^r are the entity and relation specific translation vectors between $\mathcal{L}_1$ and $\mathcal{L}_2$.

Linear transformations model introduced a $n \times n$ matrix $\mathbf{M}^1$ to transform the entity vectors from $\mathcal{L}_1$ to $\mathcal{L}_2$ (without considering the relation) as follows:

$$Z_4 = || \mathbf{M}^1\mathbf{h} - \mathbf{h}^{'} || + || \mathbf{M}^1\mathbf{t} - \mathbf{t}^{'} || \tag{2.11}$$

Linear transformation model can be extended to transform the relational vectors by introducing another $n \times n$ matrix $\mathbf{M}^2$ as follows:

$$Z_5 = || \mathbf{M}^1\mathbf{h} - \mathbf{h}^{'} || + \mathbf{M}^2\mathbf{r} - \mathbf{r}^{'} || + || \mathbf{M}^1\mathbf{t} - \mathbf{t}^{'} || \tag{2.12}$$

Later the MTransE model optimized the components described above and aligned the entities based on similarity.

IPTransE

This model [48] aligns the entities according to their semantic distance in the joint embedding space and iteratively label new entity alignment by utilizing the existing

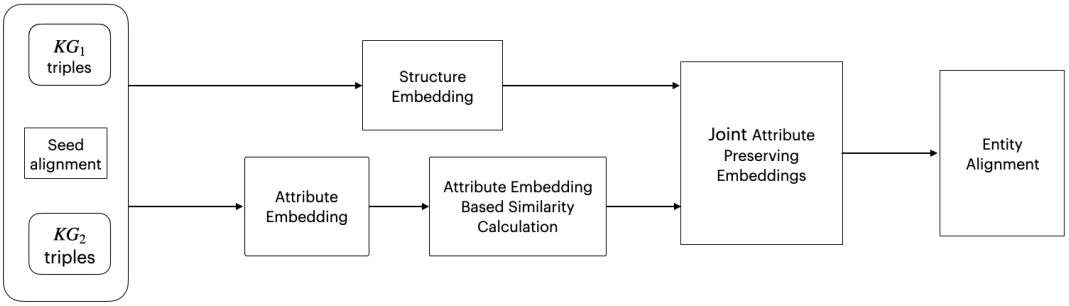


Figure 2.11: JAPE Framework

seeds. The main contributions are: they proposed parameter sharing model and iterative alignment model to learn joint embeddings and perform entity alignment simultaneously. The intuition behind the parameter sharing model was the aligned entities should have identical meanings in KGs, so aligned entities should share the same embeddings. Iterative alignment model exploited two strategies: hard and soft alignment for iterative learning of joint embeddings and entity alignment. In hard alignment model, newly aligned entities are directly stored in the aligned seeds set $\mathcal{S}_a$. Soft alignment model introduces a reliability score for newly aligned entity pair $(h1, h2)$ and based on that score the newly aligned entities are directly stored in the aligned seeds set $\mathcal{S}_a$.

JAPE

JAPE [49] proposed embeddings for entities and relations of different KGs in a common embedding space and it showed the effectiveness of attribute triples to learn the entity alignment in between different KGs. JAPE is the first model which models the attribute triples to address the entity alignment task between cross-lingual KGs. In between real world KGs there exist some aligned entities and properties but the number of aligned entities are very limited. On the other hand, the numbers of attribute triples are much larger than the relational triples. JAPE showed that attribute triples can serve as a bridge between the KGs to align entities.

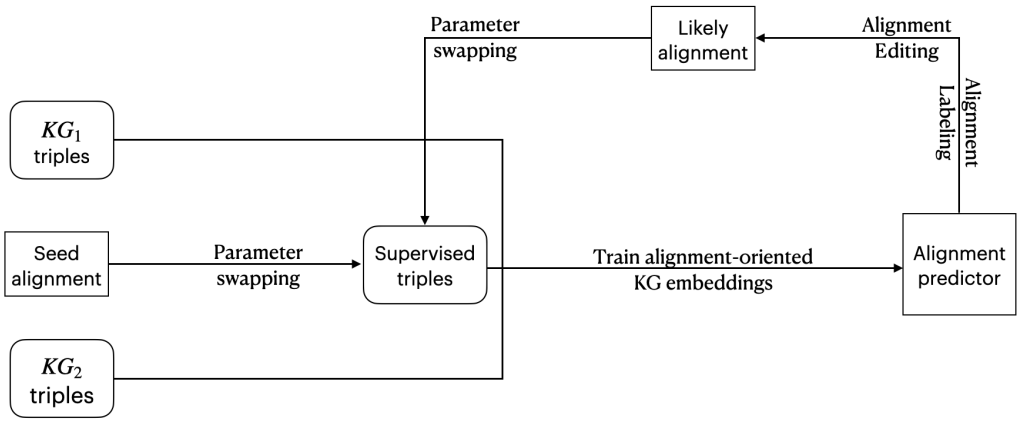


Figure 2.12: BootEA Framework

BootEA

Sun et al. [4] proposed a bootstrapping approach which also embeds the entities and relations of different KGs in a unified embedding space. The main advantage of BootEA is: it can iteratively train a classifier by using bootstrapping approach from both labeled and unlabeled data. The authors argued that traditional methods unitize negative samples by randomly replacing the head/tail entity which may mislead the results. So they exploited ϵ-truncated uniform negative sampling in their model. Like IPTransE, BootEA updates newly aligned entities to the exiting seeds set and enriches the training data iteratively.

AttributeE

AttributeE [50] exploited large numbers of attribute triples existing in the KGs. AttributeE utilized character-level literal embeddings and exploited the attribute triples like the JAPE model to learn the alignment. The main difference between JAPE and AttributeE is: AttributeE proposed attribute character embeddings method which contributed to their experimental results. JAPE and AttributeE showed satisfactory performance but quickly fails when the number of attributes triples are very limited.

MultiKE

MultiKE [51] learns the representations of the entities in the three views: name, attribute and structure. The combination strategies are proposed to integrate three view-specific embeddings to get the latest performance of the entity alignment. For each view, they employed an appropriate model to learn embeddings from it. For entity alignment, they designed two cross-KG identity inference methods at the entity level as well as the relation and attribute level, respectively, to preserve and enhance the alignment between different KGs.

COTSAE

COTSAE [52] alternatively trains structural and attribute embeddings and then combines the alignment results obtained from them[3].

Other Models

There are several other models addressed the entity alignment problem. JE [53] utilized structured embedding model (TransE) to embed different KGs into a unified space with the aim that each seed alignment has similar embeddings and showed how to use them in the cross-lingual scenario. KDCoE [54] proposed a multilingual KG embedding model and a multilingual literal description embedding model for cross-lingual entity alignment which is a semi-supervised learning approach. BERT-INT [55] proposed a method that incorporates entity description using BERT. However, in their experiment they have only used the description of multilingual dataset and zero-shot cases were ignored. Zero-shot case indicates the scenario where at least one of entities in test triples is not present in the training sample. Another embedding based model proposed by Gentile et al. [56] for aligning entities in the web tables.

2.4.3 GNN Based Alignment Models

GCN-Align

GCN-Align [57] is the first study on GNN-based EA. Like many GNN-based EA techniques, GCNAlign employs standard Graph Convolutional Networks (GCN) [58] to compute the entity embeddings from two different knowledge graphs separately. The

alignment is computed based on the distances between entities in the embedding space in the final layer of GCN using the seed alignments. They combine entity embeddings and attribute type embeddings in the convolutional computation of GCN to improve the performance of entity alignment.

GMNN

GMNN [59] formulates the EA problem as graph matching between two topic entity graphs. Every entity in a KG corresponds to a topic entity graph, which is formed by the one-hop neighbors of the entity and the corresponding relation predicates (i.e., edges). Such a graph represents the local context information of the entity. GMNN uses a graph matching model to model the similarity of two topic entity graphs, which indicates the probability of the two corresponding entities being aligned[3].

The graph matching model consists of four layers, an input representation layer, a node-level matching layer, a graph-level matching layer, and a prediction layer. The input representation layer uses a GCN to encode two topic entity graphs and obtain entity embeddings. The graph-level matching layer runs a GCN on each topic entity graph to further propagate the local information throughout the topic entity graph. Finally the prediction layer takes the graph matching representation as input and uses a softmax regression function to predict entity alignment.

MuGNN

MuGNN [60] addresses two challenges in the GNN based alignment models. The first is the heterogeneity of knowledge base structure problem, i.e., the neighbors of the same real-world entity in two knowledge bases are typically different, and may mislead the embedding learning and the alignment process. The second is the limited seed alignments problem. To tackle these problems, they use a two-step method that includes rule-based KB completion and multi-channel Graph Neural Networks entity alignment. The first step, the rule-based KB completion, aims to resolve the structural differences by completing the missing relations using a rule mining system AMIE [61]. The second step is the alignment step that uses a multi-channel Graph Neural Network, which is a combination of GCN and GAT (Graph Attention Network) [62]. In MuGNN, GAT is used to compute a connectivity matrix based on the self entity attention in each

KB and the cross-KB attention from the seed alignments, while GCN is used to capture the graph structure based on the connectivity matrix.

AliNet

AliNet [63] handles the heterogeneity of knowledge base structure problems. Specifically, they consider the multi-hop structure similarity. AliNet consists of two main components, including the Gated Multi-hop Neighborhood aggregation module and the noise reduction module. The aggregator consists of two GCN layers. The first layer is a standard GCN layer to capture the structure of a node' s immediate neighbors. The second layer is an attentive GCN layer that uses attention scores to compute the weight of the two-hop neighbors. The noise reduction module is a graph attention network to compute the aggregation of the one-hop and two-hop neighbors[3].

RDGCN

RDGCN [64] exploits dual relation graphs to improve GCN. The dual relation graph is a graph constructed by combining two KBs and creating additional edges if any relation that shares the same head or tail entities is found. To compute the interaction between the original knowledge base and the dual relation graph, RDGCN uses GAT. The attention scores computed on the dual relation graph are used as weights for the edges in the original knowledge bases to encode the graphs using GCN. Similar to the existing GNN based alignment models, the final layer of the GCN is used to compute the alignments based on the distances between entities in the seed alignments.

CEA

CEA [65] considers the dependency of alignment decisions among entities, e.g., an entity is less likely to be an alignment target if it has already been aligned to some entity [3]. The model proposes a collective EA framework. It uses structural, semantic, and string signals to capture different aspects of the similarity between entities in the source and the target KGs, which are represented by three separate similarity matrices. Specifically, the structural similarity matrix is computed based on the embedding matrices via GCNs with cosine similarity, the semantic similarity matrix is computed from the word embeddings, and the string similarity matrix is computed by

the Levenshtein distance between the entity names. The three matrices are further combined into a fused matrix. CEA then formulates EA as a classical stable matching problem on the fused matrix to capture interdependent EA decisions, which is solved by the deferred acceptance algorithm [66].

AVR-GCN

AVR-GCN [67] is based on the vectorized relational GCN (VR-GCN), which is the enhanced version of Relational GCN (R-GCN) [124]. Existing GCN-based models do not compute relation/predicate embeddings, VR-GCN explicitly computes predicate embeddings to be incorporated with the entity embeddings to learn the alignment. Before computing the embeddings using GCN, the entity representation is updated using translation based operations, i.e., if the entity is the head entity, the entity embedding is $t`r$, if the entity is the tail entity, the entity embedding is $h + r$. To enhance the alignment performance, they use the seed alignments in two ways. The first is they use the seed alignments to compute the objective function similar to the existing GCN-based methods. The second is they create a pseudo graph from the seed alignments then inject them into the original knowledge bases. The expanded graphs are expected to have more shared edges, and hence the embeddings of the same entities from two knowledge bases would be closer to each other.

AttrGNN

AttrGNN [68] learns embeddings from both relation triples and attribute triples in a unified network. It partitions each KG into four subgraphs containing attribute triples of entity names, attribute triples of literal values, attribute triples of digital values, and the remaining triples (i.e., relation triples), respectively. For each subgraph, entity embedding is computed based on attributes as well as the KG structure using GAT; then a similarity matrix between G1 and G2 is computed based on the entity embedding. Finally, the four similarity matrices are averaged to yield a final similarity matrix for the inference module. [3]

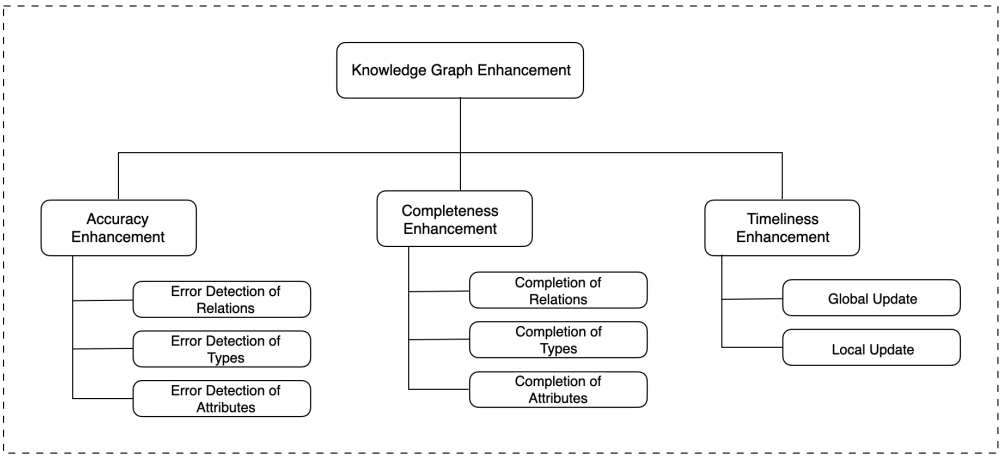


Figure 2.13: Knowledge Graph Enhancement

2.5 Knowledge Graph Enhancement

Although quality control methods are employed in each process of KG construction to ensure KG quality to an extent, there still exist some quality problems once the KG has been constructed, including the following: (1) Some errors inevitably occur during KG construction, such as missing entity or relation and wrong types of entities or relations. (2) The timeliness of some KGs gradually decreases over time, especially for KGs regarding quickly changing domains, such as user interest prediction and dynamic social networks. (3) New requirements for KG quality may arise in their application, leading to the need for KG expansion [69]. Therefore, it is necessary to correct, update, and complete KGs to enhance their quality. This section discusses KG quality enhancement methods from the above perspectives. As Figure 2.13 indicates, KG quality enhancement can be further divided into three major fields - 1) Accuracy 2)Completeness and 3) Timeliness. In the following sections, we will discuss about these sections.

2.5.1 Knowledge Graph Accuracy Enhancement

KG accuracy reflects the degree to which knowledge follows facts and it is our utmost goal to achieve ensure KG quality. However, KG accuracy is hampered by erroneous

relations, wrong entities, and inconsistent attributes in the KG. They happens due to several reasons like erroneous text source, incomplete automatic extraction method, and so on. To handle this, error detection should be performed for accuracy enhancement. Error detection is the task of finding and correcting knowledge in a KG that is not consistent with objective facts. KG error detection methods are divided into error detection of relations, entity type, and attributes (see Figure 2.13).

Error detection of Relation

Error detection of relation aims to find relation assertions in the KG that do not match facts, including detection with internal and external knowledge.

Internal knowledge-based detection methods utilize the knowledge from the KG itself and judge the relations by exploring the association features or distribution of the entities in the graph. Path information and the types of entities are used for error detection of relations in the methods proposed by Lao et al. [70] and Melo and Paulheim [71] respectively.

Statistical distribution-based methods are another representative method that exploits internal knowledge. Paulheim and Bizer [72] proposed an algorithm that calculates the distribution of the head and tail entities of a relation in a triplet to infer the probability that the triplet is correct.

Distant supervision is widely used in error relation detection based on external data. Dong et al. [9] used the path information of other KGs as prior knowledge to obtain entity-related information for error detection. Paulheim and Gangemi [73] mapped the relation and entity types of a triple instance to the corresponding types of an external ontology library.

Error detection of Types

The error detection of entity type could be considered as an extension of error detection of relations, aiming to detect erroneous entity types. Paulheim and Bizer [158] pointed that relations have more tendency to be wrong in KGs compared to entity type. Thus, detecting the wrong triple first using a method of detecting wrong relations may be a feasible method for error detection of entity type, and the corresponding part can be corrected according to different assumptions after the target triple is detected.

Therefore, most methods used in error detection of relations can be used in the task of finding erroneous type assertions.

Error detection of Attributes

The error detection of attributes is used to find erroneous attributes values of an entity. Outlier detection is mainly used in this field to identify the instances in the numerical data that are very different from the majority. Wienand and Paulheim studied the application of unsupervised numerical outlier detection methods in Knowledge Graphs and used the interquartile range (IQR) and semantic grouping to reduce the detection rate of natural outliers [74]. Fleischhacker et al. proposed an extension by merging the outcomes of multiple independent outlier detection [75]. They proposed a multi-attribute consistency method which detects erroneous attributes by combining the information of entities and discovering inconsistencies. Rahoman et al. proposed a nearest-neighbor based error detection technique for error detection over type annotated linked data [76]. Golab et al. used sequential dependencies to express relationships between ordered attributes and discover the erroneous attributes [77]. Koudas et al. studied metric functional dependencies, which is used for comparison of different attribute formats [78]. Besides sequential and metric functional dependencies, arithmetic operations between numerical attributes can be used for the error detection of attributes. Fan et al. proposed such a numerical function dependency relationship [79]. It allows users to specify arithmetic relationships between numerical attributes as data quality rules and detects incorrect attributes in a unified logic framework. Zhao et al. proposed a method that incorporate multiple KGs for link prediction and attribute error detection [80]. Li et al.proposed a probabilistic framework for detecting errors in the numerical attributes of an entity [81].

2.5.2 Knowledge Graph Completeness Enhancement

Similar to accuracy enhancement, the task of completeness enhancement has three aspects- (1) Completeness of Relation (2)Completeness of type and (3) Completeness of Attribute. In the following subsections, we will discuss each aspect respectively.

Completeness of Relation

The goal of the relation completion task is to predict missing relations between entities. Missing relation prediction methods are primarily classified into probabilistic, deep learning, and tensor factorization methods [82].

(1) Probabilistic models are generally used for completion of relations in the early search. For example, the probabilistic graphical model MLN[83], ProbKB[84] assigns the proper probability for each knowledge triple to reckon the rationality of knowledge. Moreover, the probabilistic graphical model predicts missing relations using the correct probabilities of knowledge triples in a KG.

(2) Deep learning approaches learn the latent factor representation of entities and relations for relation prediction. These approaches target the proximity of the embedding of the tail entity to that of the head entity, which is transformed by relation to decide whether the triplet is true. TransE [20] is a typical deep learning model used for relation prediction. Some methods adjust the framework and structure to incorporate prior knowledge of the KG into the model and have achieved even more promising results for this task. TransH [22], TransR [21], TransD [23], TranSparse [**?**] are some examples of such models.

(3) The tensor factorization approach depicts the semantic features of KGs through tensors, providing better interpretability than deep learning approaches. The main idea of tensor factorization is to represent a KG as a 3-D tensor and describe the relation prediction as a 3-D tensor completion problem [69]. Some notable studies include RESCAL[26], ComplEx [29], TuckER [30]. However, the expressiveness of the model is a bottleneck that must be improved. Moreover, they tend to increase the complexity of the model. Proper balance of expressiveness and complexity among these models is yet to be found.

Although significant work has been conducted in the relation completion domain, there exist some challenges. For instance, most studies are highly dependent datasets, and they may not be applicable to all KGs, especially those in industrial domains. The results obtained in this method cannot guarantee KG accuracy, and the predicted results cannot be used directly. Additionally, most studies have simply predicted the relationship between existing entities in a KG. These studies failed to predict entities and their corresponding relations that do not appear in the KG (zero-shot). The

open-world KGC task [85], aims to address this issue. This task relaxes the assumption of the closed-world hypothesis, enabling the KG to adapt to an environment in which entities are rapidly developing [69]. However, this new direction is relatively less explored and requires further research.

Completeness of Types

The task of completion of entity types is generally called entity typing, which aims to infer the missing type of an entity. (1) Entity typing through a heuristic probability model is a traditional statistical method. For example, Paulheim and Bizer [180] proposed SDtype, which uses a type inference mechanism based on heuristic links. The model infers the type of an entity by counting the types of possible entities that have a higher connection frequency with all relationships connected to the entity. Because the probabilistic method used by SDtype ignores the specificity of knowledge expression, SDtype encounters difficulties in fine-grained classes. There are bottlenecks in SDtype' s effectiveness, although it is good at predicting classes with a higher frequency and can make large-scale corrections.

(2) Learning-based methods for entity typing train a classifier to predict the type for entities. Earlier prediction methods made predictions by manually mining the entity representation patterns of a KG. In recent years, the use of DNNs for representation learning has gradually become a common choice because of the increasing requirements for model performance capabilities [197]. These entity typing methods map entities and relationships to a low-dimensional feature space through a DNN [198].

Completeness of Attributes

Attribute completion research is limited because it is much more complicated task than entity type predication or link prediction task. There is no defined standard to detect whether entities have missing attributes or not. Moreover, due to the different hierarchical structures of different entities, we cannot simply add the discovered missing attributes. Another very big challenge is to measure the accuracy of the overall Attribute completion method. Razniewski et al. proposed that the lack of attributes of an entity can be estimated by referring to similar entities and according to the amount of access of the entity and growth patterns of an entity' s attributes, Pattern

matching methods can be useful the task [86]. Probase+[87] proposed by Liang et al., uses collaborative filtering framework which assumes that similar entities have similar upper and lower levels . Hypernyms and hyponyms relationship in a KG helps this method to find missing attributes for entities.

2.5.3 Knowledge Graph Timeliness Enhancement

The timeliness enhancement of KGs is mainly achieved through global and local update methods. Global updates maximize the timeliness of KGs. However, as KG scale increases, this method becomes extremely resource-consuming. Compared with global updates, local updates save computing resource consumption but must obtain the required entity information to update. Recently, the local update method has received more attention [69].

2.6 Summery

In this chapter, we reviewed the literature related to the problems of knowledge graph alignment and KG enhancement. Previous work tackles the problem of KG alignment using various approaches and combinations of embeddings and attributes. However, the accuracy and applicability of existing approaches still need improvements. On the other hand, KG enhancement using attribute completion or enhancement is a relatively new area of research, and integrating multiple KGs for this task is a somewhat unexplored approach. Hence there is significant scope for improvement. In this thesis, we aim to address the issues from the respective fields and proposed methods for improvements. We detail our algorithms to tackle the problem of knowledge graph alignment and attribute enrichment in the following chapters

3

Entity Alignment via Summary and Attribute Embeddings

Entity alignment is the task of integrating heterogeneous knowledge among different knowledge graphs (KGs). Knowledge Graph (KG) is a popular way of storing facts about real-world entities. Unfortunately, very limited number of the entities stored in different KGs are aligned. This paper presents an embedding-based entity alignment method that finds entity alignment by measuring the similarities between entity embeddings. Existing methods mainly focus on relational structures and attribute information for the alignment process. Such methods fail when the entities have a fewer number of attributes or when the relational structure couldn't capture the meaningful representation of the entities. To address this problem, we propose EASAE, an Entity Alignment method using Summary and Attribute Embeddings. We exploit the entity summary information available in KGs for entities' summary embedding. To learn the semantics of the entity summary, we employ Bidirectional Encoder Representations from Transformers (BERT). Our model learns the representations

of entities by using relational triples, attribute triples, and summary as well. We perform experiments on real-world datasets, and the results indicate that the proposed approach outperformed the state-of-the-art models for entity alignment.

3.1 Introduction

In recent years, knowledge graphs (KGs) are drawing massive attention and are being extensively used in AI applications, such as question answering, semantic search, information retrieval, and web mining. KGs contain a large amount of structured knowledge/facts. KGs are multi-relational directed graphs consist of entities as graph nodes and relations as edges, which is an efficient way to store real-world facts as structured data in triple format. A relational triple can be denoted as (h, r, t) where h and t are the head entity and the tail entity respectively and r is the relation between the h and t. If the object is literal then we call it attribute triple.

Entity alignment task refers to finding the same real-world entities in multiple KGs. Recently, increasing attention has been paid to leveraging the KG embedding techniques for addressing the entity alignment problem. KG embedding models represent entities and relations in a low-dimensional vector space while preserving the KG semantics.

There are several existing models which discussed the entity alignment task by means of KG embedding [88, 53, 48, 4, 49, 50]. MTransE [88] is a multilingual knowledge graph embedding model that learns the multilingual knowledge graph structure. JE [53] embeds different KGs into a unified space using TransE [20] with the aim that each seed alignment has similar embeddings. Like JE, IPTransE [48] also represents different KGs into a unified embedding space, which is an iterative and parameter sharing method. Another popular entity alignment model BootEA [4] also embeds two KGs in a unified space and iteratively labels new entity alignment as supervision. It achieved very impressive performance in entity alignment task. JAPE [49] learns entity alignment by jointly exploiting attribute and relational embeddings. AttributeE [50] is an extension of JAPE, they exploit large numbers of attribute triples existing in the knowledge graphs and generates attribute character embeddings to align the entities in two KGs. The existing models mostly depend on relational and attribute triples. They require seed alignments (i.e., seed is the set of aligned triples from two

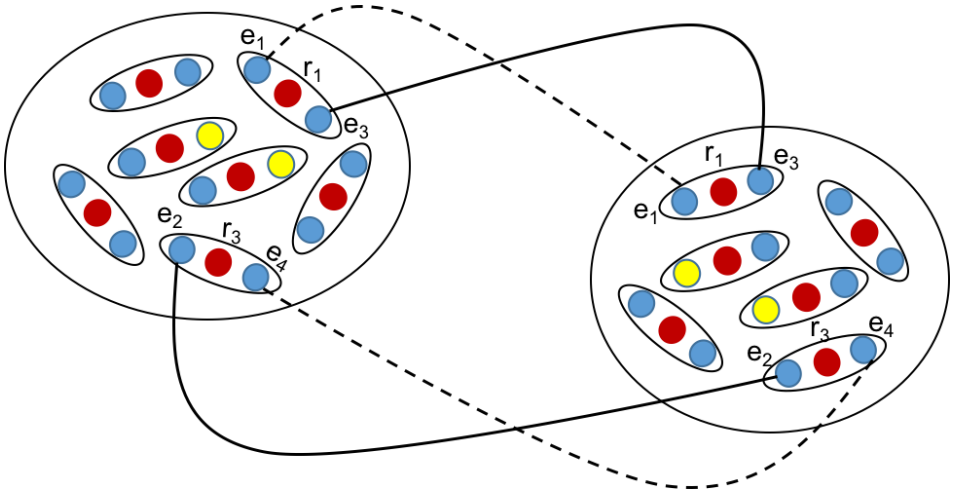


Figure 3.1: KG_1 and KG_2 denote two different knowledge graphs. Triples are represented by the oval shape. Blue, red and yellow circles stand for the entities, relations and attributes respectively. Solid lines are connecting aligned entities, while dashed lines demonstrate equivalent relationship to be discovered.

KGs), which is very limited. It is true that attribute is very helpful in entity alignment task but existing models will fail to align entities in two KGs, where entities have small number of attributes attached to them. To tackle the above challenges, we propose a novel model EASAE for entity alignment which exploits entity summary extracted from the KGs along with the relational and the attribute triples. We exploit BERT [2] to learn the summary embeddings in EASAE. In our model, all the components discussed above are jointly optimized to improve the alignment performance. Our proposed approach is an ensemble method of symbolic system and sub-symbolic system. We have incorporated entity summary as part of symbolic system with embeddings as sub-symbolic system. We summarize the main contributions of this paper as follows:

- We propose a model consists of three components: entity summary embedding, relational embedding and attribute embedding. We also introduce a weighted average technique to combine them.
- We show the effectiveness of entity summary embedding to align entities while very small number of attribute triples exist. Our method can perform well in zero shot scenario.
- We perform experiment in three real world datasets. Our experiments on

those three datasets show that our model largely outperforms the existing state-of-the-art embedding-based entity alignment models.

3.2 Preliminary

In this section we formally define the terms used in this paper and the problem as well.

Definition 3.1 *Knowledge Graph (KG): A knowledge graph KG = (E, R, T) , where E, R, T are the set of entities, relations and triples respectively.*

Definition 3.2 *Relational Triples: $T \subset E \times R \times E$ is a set of relational triples representing the relations between entities, where E and R is the set of all entities and relations respectively.*

Definition 3.3 *Attribute Triples: $A_T \subset E \times A \times L$ is a set of attribute triples representing the attributes of entities, where A is a set of all attributes, and each attribute $A_i \in A$ has a corresponding literal attribute value set $L_i \in L$.*

Definition 3.4 *Entity Alignment : Given two KGs, KG_1 and KG_2, the entity alignment problem aims to find every pair (e_1, e_2) where $e_1 \in KG_1$, $e_2 \in KG_2$, and e_1, and e_2, represent the same real-world entity.*

Given two knowledge graphs, the objective of our model is to jointly learn the relational structure, attribute embedding and summary embedding to find all the pairs of entities between KGs which represent the same real word entities. In our paper, we use bold lowercase letters to represent embedding vectors and bold uppercase letters to denote matrices.

3.3 Proposed Methodology

In this section, we describe our model architecture. We exploit embedding-based techniques in our proposed EASAE model. Figure 3.2 illustrates the EASAE model overview. EASAE can be typically divided into three major components: (1) Predicate alignment between KG_1 and KG_2; (2) Representation learning for entities, relations, and

attributes ; (3) Entity alignment process. For the relational embeddings, we need to have a unified vector space. Therefore, we merge two KGs based on the predicate similarity. We have adopted the same predicate aligning method described in AttributeE [50]. Then we calculate the summary, relational, and attribute embeddings and finally proceed to the entity alignment process. Suppose we have the entity "Washington" in two knowledge graphs KG_1 and KG_2 (we refer them as $Washington_{KG_1}$ and $Washington_{KG_2}$). We retrieve the summary of the entity $Washington_{KG_1}$ and $Washington_{KG_2}$ and calculate the summary embedding using BERT. Similarly we calculate relational and attribute embedding for $Washington_{KG_1}$ and $Washington_{KG_2}$. After that we ensemble them by employing a weighted averaging technique to calculate the overall score for $Washington_{KG_1}$ and $Washington_{KG_2}$. Finally using confine similarity we try to predict whether $Washington_{KG_1}$ and $Washington_{KG_2}$ are aligned or not.

In our model, we use the TransE model to embed the entities and relations. There are two reasons for using TransE: first, entity alignment is 1-to-1 mapping, and TransE shows satisfactory performance in 1-to-1 relations; second, TransE is a very simple but powerful tool, and it is very easy to interpret the model architecture. TransE uses the same embedding space for both relationship and entity embeddings. To learn the embeddings, we merge the similar predicates of the two KGs. For the representation learning part, we have defined three different sub-components: Summary Embedding (SE), Relational Embedding (RE), and Attribute Embedding (AE). We use BERT to learn the SE of the entities [2]. In RE, we exploit the translational embedding model, TransE to embed the structured triples. This part is similar to KG embeddings. Finally, AE uses character level embeddings for attribute triples. For the entity alignment part, we marge all the three major components and predict the alignment using cosine similarity. Here we have extended our previous paper [89] by introducing a weighted average technique for combining three major components and we have conducted extensive experiment to prove EASAE model's performance. We also exploited one more dataset (AttrE-Dataset) in this paper.

3.3.1 Predicate Alignment

For the embedding-based entity alignment, we need to have a unified vector space; therefore, we merge two KGs based on the predicate similarity. We have adopted the

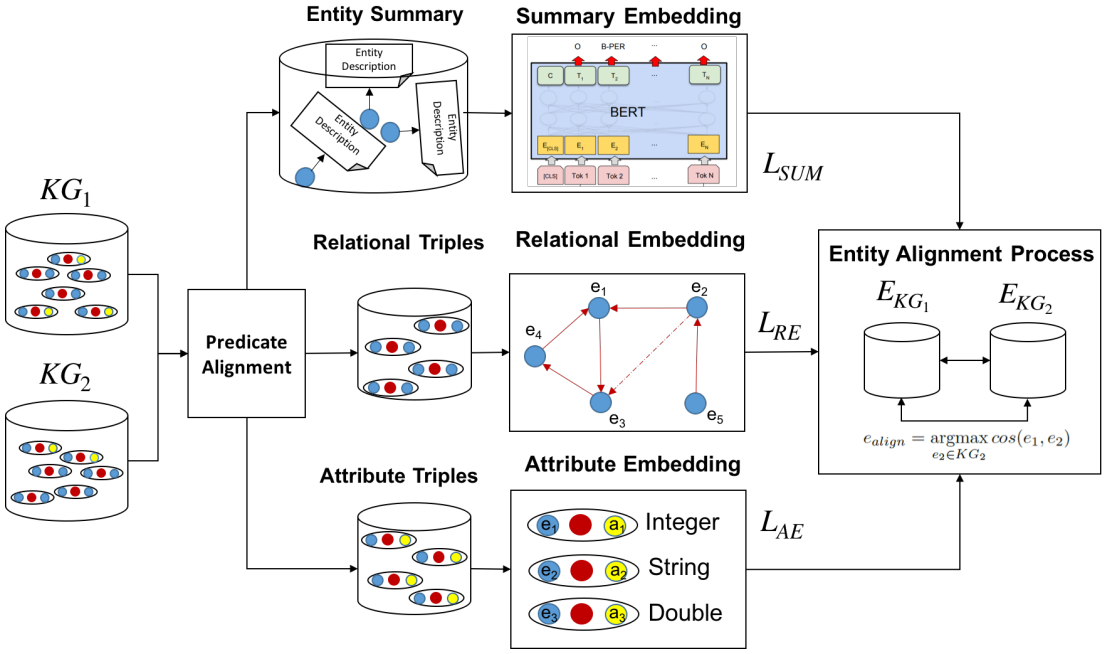


Figure 3.2: EASAE model architecture. The figure of BERT embedding is taken from their original paper [2] .

same predicate aligning method described in AttributeE [50] and also used list of aligned predicates proposed by Zhao, et al [80]. In Figure. 3.3, we show the intuition behind predicate alignment. The predicate alignment module finds partially similar predicates, e.g., dbp:spouse vs. yago:isMarriedTo and renames them with a unified naming scheme (e.g., MARGED:spouse). Based on this unified naming scheme, we merge KG_1 and KG_2 into Merged KG.

3.3.2 Summary Embedding

For summary embedding, we consider the textual description associated with each entity as summary. As example, in DBpedia [90] most of the entities have a short abstract; similarly Wikidata [14] provides a summarize textual description of entities which contain the basic information about those entities. We employ BERT to generate a set of word vectors from the summary of each specific entity which is usually capable of capturing the main ideas of entities. Recently, BERT achieved the state-of-the-art performance in the embedding tasks in comparison with the other well established

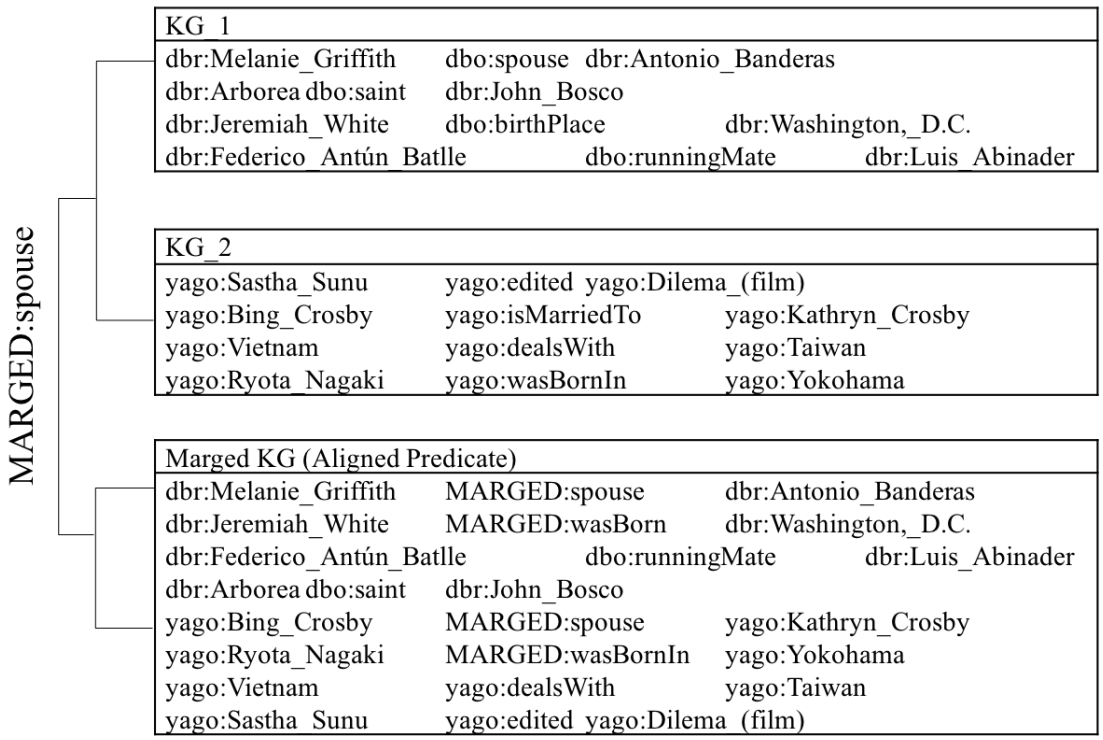


Figure 3.3: Predicate Alignment of KG_1 and KG_2

techniques e.g., CNN, GRU based models. We discuss the process of obtaining the summary embeddings using BERT in the following section.

With summary or short description of an enitity, a set of keywords can be generated which are usually capable of capturing the main ideas of entities. We assume that similar entities might have slightly different descriptions, but they should have similar keywords.

Bidirectional Encoder Representations from Transformers

Word embeddings are dense vector representations of words in lower dimensional space. For word embeddings pre-trained language representation models are used and they can be divided into two categories: feature-based and fine tuning approaches. Traditional word embedding methods such as Word2Vec [91, 36] and Glove [92] aimed at adopting feature-based approaches to learn context-independent words vectors. In contrary fine tuning approaches like BERT works well in capturing the contextual

representation of the texts. Bidirectional Encoder Representations from Transformers (BERT) model [2] has built on a multilayer bidirectional transformer encoder as the name suggest. It is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context. It is a state-of-the-art model for a wide range of tasks, such as word embedding, question answering and language inference, without substantial task specific architecture modifications. BERT framework consists of two basic steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabelled data over different pre-training tasks. BERT is using two unsupervised tasks for pre-training named as: Masked LM and Next Sentence Prediction (NSP) . In the first task, BERT predicts randomly masked input tokens. In the second task, BERT predicts whether two input sentences are consecutive or not during pre-training. It can be seen as a special case of Question Answering and Natural Language Inference. For fine tuning, we first initialize BERT with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters [2].

$$\mathbf{e}_{sum} = \mathbf{w}_1 + \mathbf{w}_2 + \mathbf{w}_3 + \cdots + \mathbf{w}_n \tag{3.1}$$

For each entity we feed the summary to the BERT model and we get the embeddings of each word. Then we sum up the embeddings of each word to get the entity embedding based on the summary of that specific entity.

BERT base model uses 12 layers of transformer encoders, each output from each layer of these (12 layers) can be used as a word embedding. As per the suggestion of the BERT authors, one of the best performance can be found by summing the last 4 layers. We follow their suggestion in our implementation.

Embedding Learning

In our model, we leverage the entity summary to align the entities between two KGs. Xie et. al. already showed that entity summary/description can be a powerful tool for KG completion task [93] and KDCoE [54] followed them in their paper. Summary Embedding is effective while the aligning entities have less number of attributes in the KG. There are many entities in KGs with no attribute values at all, in such

cases summary embeddings might be an efficient tool to bridge the gap. We use three different scoring functions to learn the entity summary embeddings. They are as follows:

$$\begin{aligned} f_s &= || \mathbf{h}_{sum} + \mathbf{r} - \mathbf{t}_{sum} ||_{l1/l2} \\ f_{st} &= || \mathbf{h}_{sum} + \mathbf{r} - \mathbf{t} ||_{l1/l2} \\ f_{hs} &= || \mathbf{h} + \mathbf{r} - \mathbf{t}_{sum} ||_{l1/l2} \end{aligned} \tag{3.2}$$

In Eq. (1), for the scoring function f_s: the $\mathbf{h}_{sum}$ and the $\mathbf{t}_{sum}$ are the representations of head and tail entity considering the summary as described in previous section; f_{st} captures the representations of head entity based on summary and tail entity uses relational embedding based representation and f_{hs} is the exact opposite of f_{st} for head and tail entities. Here, $l1/l2$ denotes the $L1/L2$ norm. The scoring functions: f_s, f_{st} and f_{ts} also help our method to perform in the zero-shot scenario. Zero-shot scenario focuses on the situation when at least one of entities in test triples is out of the training sample. Structure-based representations cannot deal with this situation because they have no representations for entities which are out of training set. We have used these three scoring functions to deal with such situation. Therefore we can clearly understand, when we face zero shot scenario, where we don' t have structure-based representations on those cases we consider the summary based representations using summary vector generated by BERT. The summary embedding model learns the entity embeddings as follows:

$$L_{SUM} = f_s + f_{st} + f_{hs} \tag{3.3}$$

where L_{SUM} is the loss function for summary based representations of entities and the objective is to minimize L_{SUM}.

3.3.3 Relational Embedding (RE)

The entities are linked by relations which characterize the structure of KGs. In RE model we analyze the relation structures of the triples. To preserve relational structures, we adopt TransE [20] to interpret a relation as a translation vector from its head entity to tail entity. For Relational Embedding (RE), we have used the similar method described in JSAE [89].

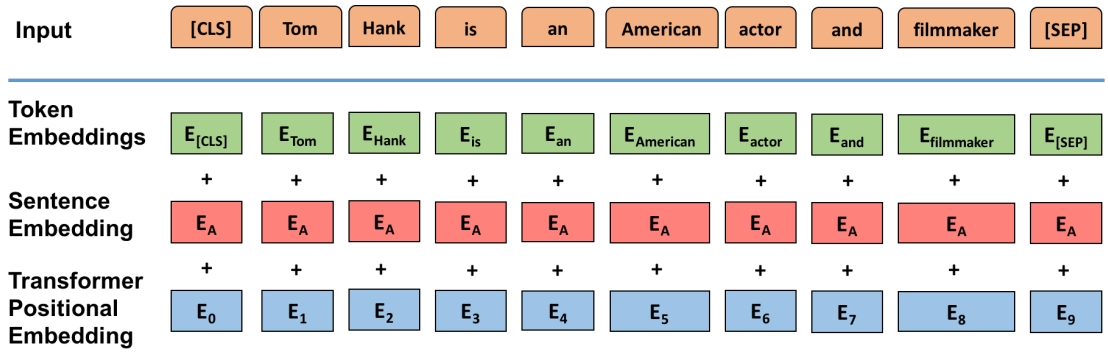


Figure 3.4: BERT Input Embedding Representation. The input sentence we use here is copied from Dbpedia abstract of Tom Hanks.

Similar to AttributeE [50], we have also exploited the weight α to control the embedding learning over the triples with aligned predicates. To learn the structure embedding, in our model, we minimize the following objective function L_{RE} :

$$L_{RE} = \sum_{(h,r,t)\in T_r} \sum_{(h',r,t')\in T'_r} max(0, \gamma + \alpha(f_r(h,t) - f_r(h',t'))) \tag{3.4}$$

where $\alpha = \frac{count(r)}{|T|}$, γ is the margin, $f_r(h,t)$ and $f_r(h',t')$ denote the scoring function for valid triples and negative triples respectively, T_r is the set of valid relationship triples, T'_r is the set of corrupted relationship triples, f_r is the plausibility function for relational triples, count(r) is the number of occurrences of relationship r, and $|T|$ is the total number of triples in the merged KGs.

3.3.4 Attribute Embedding (AE)

We have employed the basic of TransE for attributes and literals embedding, but unlike relational embedding we interpret predicate r as a translation from the head entity h to the attribute a. For structural differences in various KGs, the same attribute can have multiple representations, as an example one KG might stores the decimal points up to 4 digits where others might keep more (e.g. 23.7000 vs. 23.70000001 as the latitude value of an entity). Therefore to encode the attribute value, we use a compositional function $\phi(a)$ where a is a sequence of the characters of the attribute value $a = c_1, c_2, c_3, \ldots, c_l$

and define the relationship of each element in an attribute triple as $h + r \approx \phi(a)$. The compositional function encodes the attribute value into a single vector and maps the similar attribute values to a similar vector representation. We have applied an N-gram-based compositional function in our proposed model. We use the summation of n-gram combination of the attribute value. The equation is given below:

$$\phi(a) = \sum_{n=1}^{N} \left(\frac{\sum_{i=1}^{l} \sum_{j=i}^{n} c_j}{l - i - 1} \right) \tag{3.5}$$

To learn the attribute embedding, we minimize the following objective function L_{AE}:

$$L_{AE} = \sum_{(h,r,a) \in T_a} \sum_{(h',r,a') \in T_a'} max(0, \gamma + \alpha(f_a(h,a) - f_a(h',a'))) \tag{3.6}$$

Here, $f_a(h,a)$ and $f_a(h',a')$ denote the scoring function for valid attribute triples and negative attribute triples respectively. T_a is the set of valid attribute triples from the training dataset, while $T_a^{'}$ is the set of negative attribute triples (A is the set of attributes in G). The negative samples are constructed by replacing the head entity with a random entity or the attribute with a random attribute value. Here, $f_a(h,a)$ is the plausibility score that based on the embedding of the head entity h, the embedding of the relationship r, and the vector representation of the attribute value that computed using the compositional function $\phi(a)$. We adopted the idea of Attribute Embedding (AE) from our previous model JSAE [89].

3.3.5 Entity Alignment Process

We combine the score from three embeddings models into an ensemble method to achieve better predictive performance. Here, we employ a weighted averaging technique to get the overall score of our model. We denote, $\boldsymbol{e}_{cmb}$ as the combined embedding for e. M is the number of embeddings which is three in our case and $\boldsymbol{e}^{(i)}$ be the embedding from each component.

$$\boldsymbol{e}_{cmb} = \sum_{i=1}^{M} \omega_i \boldsymbol{e}^{(i)} \tag{3.7}$$

Instead of straightforward linear combination of the three embeddings of EASAE

model we have assigned weights ω_i to each entity embeddings to emphasize on important component. To calculate ω_i we took the average embedding from the three embeddings (Summery, Relational and Attribute) and compute the deviation of each embedding from the average. Let, $\overline{\boldsymbol{e}}$ is the average of the embeddings.

$$\overline{\boldsymbol{e}} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{e}^{(i)} \tag{3.8}$$

After that we have applied the following equation to find ω_i.

$$\omega_i = \frac{cos(\boldsymbol{e}^{(i)}, \overline{\boldsymbol{e}})}{\sum_{j=1}^{M} cos(\boldsymbol{e}^{(j)}, \overline{\boldsymbol{e}})} \tag{3.9}$$

This would ensure if one embedding is far away from its average embedding, it would have a lower weight because the impact this embedding is not significant. This is a kind of late combination, because it aggregates embeddings after they have been learned independently.

Finally, we compute the following equation for entity alignment as described in AttributeE [50].

$$e_{align} = \underset{e_2 \in KG_2}{\operatorname{argmax}}\, cos(e_1, e_2) \tag{3.10}$$

Given an entity $e_1 \in KG_1$, we compute the similarity between e_1 and all entities $e_2 \in KG_2$ to find the aligned entity pair. For each query entity, we expect the rank of its' truly-aligned target entity to be at the top of the rank list.

3.4 Experiments

3.4.1 Datasets

To evaluate our model, we have used total three datasets, which are shown in Table 1. Two of them are considered to be the primary datasets for our method evaluation namely DBP-WD and DBP-YG. They are recently introduced by BootEA [4]. These two datasets were sampled from DBpedia [90], Wikidata [14] and YAGO [8], each of which contains 100,000 aligned entity pairs. The statistics of the datasets is given on Table 1.

In our model we exploit the aligned predicates to align the entities between KGs.

Table 3.1: Dataset Statistics.

Datasets		Entity	Relation Triples	Attribute Triples
DBP-WD	DBpedia	100,000	463294	381166
	Wikidata	100,000	448774	789815
DBP-YG	DBpedia	100,000	428952	451646
	YAGO	100,000	502563	118376
AttrE-Dataset	DBpedia	33627	36,906	184672
	YAGO	30628	38451	173309

DBP-WD and DBP-YG datasets contain 52 and 30 aligned predicates respectively. Each dataset provides 30% reference of entity alignment data as seeds and left the remaining as testing data.

For Summary embedding model, we have extracted the entities' summaries as follows: (1) For DBpedia we have considered the abstract (en) as the summary; (2) similarly for Wikidata we have considered the Wikipedia (en) abstracts (for the entities in our dataset) as the summary; (3) YAGO does not provide the summary/description on their data dump but it provides Wikilinks to the corresponding entities , so we have extracted the Wiki pages associated with the YAGO entities (for the entities in our dataset) and then copied the abstracts from the Wikipedia links as summaries. Although, DBpedia, Wikidata and YAGO are mostly inspired by Wikipedia, but due to the heterogeneity of KG ontology the summaries are not the same in these KGs. Here we present the example of the entity 'Achilles' (Greek mythological character, Achilles or Achilleus was a hero of the Trojan War) .

In the following table, we have added the DBpedia short abstract of 'Achilles' and Wikidata entity summary of 'Achilles'. You can see the clear difference in their representations and it motivates us to use summary embedding to utilize these information to enhance the performance of entity alignment task.

We have also used another dataset AttrE-Dataset which was introduced by AttributeE [50]. This dataset is used for conducting a comparison analysis in terms of the dependency on the number of attribute triples. The dataset contains 15,000 aligned entities and 72 aligned predicates from DBpedia and YAGO.

Table 3.2: Summary from Dbpedia and Wikidata for the entity 'Achilles'

Dbpedia Short Abstract	Wikidata Summary
Achilles In Greek mythology, Achilles was a Greek hero of the Trojan War, the central character and the greatest warrior of Homer's Iliad. Achilles also has the attributes of being the most handsome of the heroes assembled against Troy. Later legends (beginning with a poem by Statius in the first century AD) state that Achilles was invulnerable in all of his body except for his heel. Since he died due to an arrow shot into his heel, the Ächilles' heel has come to mean a person's principal weakness.	In Greek mythology, Achilles was a hero of the Trojan War, the greatest of all the Greek warriors, and is the central character of Homer's Iliad. He was the son of the Nereid Thetis and Peleus, king of Phthia. Achilles' most notable feat during the Trojan War was the slaying of the Trojan prince Hector outside the gates of Troy. Although the death of Achilles is not presented in the Iliad, other sources concur that he was killed near the end of the Trojan War by Paris, who shot him in the heel with an arrow. Later legends (beginning with Statius' unfinished epic Achilleid, written in the 1st century AD) state that Achilles was invulnerable in all of his body except for his heel, because when his mother Thetis dipped him in the river Styx as an infant, she held him by one of his heels. Alluding to these legends, the term "Achilles' heel" has come to mean a point of weakness, especially in someone or something with an otherwise strong constitution. The Achilles tendon is also named after him due to these legends.

3.4.2 Experimental Settings

To analyze the performance of our model we have conducted three experiments. We have executed Experiment 1 and 2 using DBP-WD and DBP-YG datasets. The Experiment 1 is performed to compare the individual performances of the embedding models of EASAE to identify the contribution of each embedding model.

In Experiment 2, we have compared our model with the state-of-the-art models to evaluate the contribution of EASAE. We compared our proposed method with the following models: MTransE [88], IPTransE [48], JAPE [49], BootEA [4], KDCoE [54], AttributeE [50] and MultiKE[51]. For getting best model performance we tune the hyperparameters by grid search. We select the margin γ from {1, 5, 10}, the embedding dimension d of vectors among {50, 75, 100, 150, 200}, the learning rate α from {0.001, 0.01, 0.1}, the batch size B from {20, 50, 100, 200}. For scoring function $L1$ norm was selected. EASAE takes maximum 500 epochs to converse in the experimental datasets.

Experiment 3 is conducted to identify the contribution of our summary embedding in such cases where the number of attribute triples are limited. For this experiment we have used AttrE- Dataset. Because the number of attributes is very high in AttrE-Dataset.

To evaluate the performance of the models in all three experimental settings, we employ Hits@1 and Hits@10 (Hits@k indicates the proportion of correctly aligned entities ranked in the top k predictions), and the mean rank (MR) which denotes the mean the rank of the correct entities for alignment. Higher Hits@k and lower MR indicate better performance.

3.4.3 Experiment 1

In our first experiment, we have examined the effectiveness of summary embeddings for entity alignment problem. We have compared the individual performances of the embedding models of EASAE to identify the contribution of our summary embedding model. So, we performed experiments for aligning the entities by the summary embedding, attribute embedding and relational embedding models independently on the datasets. The results are shown in the Table 2. As each embedding model focuses on different features so individually they cannot utilize the other feature hence the result is not so promising. We have also performed experiment using two more

Table 3.3: Results of entity alignment using different embeddings combination

Model	DBP-WD			DBP-YG		
	MR	Hits@1	Hits@10	MR	Hits@1	Hits@10
Summary Embedding (BERT)	927	70.40	80.34	1108	64.89	78.49
Relation Embedding	152	54.24	65.08	58	33.49	59.89
Attribute Embedding	6,774	61.13	72.08	1,001	62.52	73.89
Attribute + Relation Embedding	145	67.83	77.98	138	56.34	69.63
Summary + Relation Embedding	207	74.40	85.34	110	74.39	83.18
Summary +Attribute + Relation Embedding	**43**	**92.04**	**97.83**	**29**	**90.86**	**96.64**

combinations: (1) using summary embedding and relational embedding (2) using attribute embedding and relational embedding. Relational embedding plays a vital role in translation based entity alignment models, therefore we didn't compare with summary embedding and attribute embedding combination. The combinations perform better than the individual models (SE, AE and RE). However, when we combine all three embeddings in proposed EASAE, we have observed that it can capture the meaningful representation of the entities and therefore give us the best outcome.

3.4.4 Experiment 2

In Experiment 2, the experimental results show that proposed EASAE model consistently outperforms the baseline models (see Table 3). MTransE, IPTransE, and JAPE rely on the number of the seed alignments, therefore their experiment mainly focused on the cross-lingual datasets of same knowledge graph (e.g. DB-en, DB-fr). BootEA [4] efficiently used their method for aligning two different KGs, but they used 100,000 aligned entities for defining the unified space which is a large number of pre-aligned data. Though DBpedia and Wikidata contain large numbers of already aligned entities but other KGs usually do not provide such a large number of aligned entities. KDCoE [54] used entity description for their alignment method. But their focus on the paper was to align cross-lingual entities in the same KG, so their method can not show good performance for aligning entities between different knowledge graphs. AttributeE [50] tactfully handled this issue by using predicate alignment instead of relying on seed

Table 3.4: Comparison with the state-of-the-art embedding-based entity alignment models. The results of MtransE, IPTransE, JAPE and BootEA were directly copied from BootEA [4]. We reproduced the others results using their source code.

Model	DBP-WD			DBP-YG		
	MR	Hits@1	Hits@10	MR	Hits@1	Hits@10
MTransE [88]	656	28.12	51.95	512	25.15	49.29
IPTransE [48]	265	34.85	63.84	158	29.74	55.76
JAPE [49]	266	31.84	58.88	189	23.57	48.41
BootEA [4]	109	74.79	89.84	34	76.10	89.44
KDCoE [54]	182	57.19	69.53	137	42.71	48.30
AttributeE [50]	142	68.77	80.78	108	57.05	70.64
MultiKE [51]	114	91.45	95.19	35	88.03	96.63
JSAE [89]	112	81.48	92.34	84	79.56	91.45
EASAE	**43**	**92.04**	**97.83**	**29**	**90.86**	**96.64**

alignment. However, for their experiment they have used the benefits of huge attribute triples. Their dataset contains attribute triples, which is three times larger than the relational triples and in our observation, it is not happened in the all real cases. In this experiment, we have used the predicate alignment so EASAE does not require pre-aligned seeds in the training phase. MultiKE uses three kinds of information such as name, attribute, and entity structure. Entity name information can certainly add leverage for alignment and they achieved latest results. However, entity name can be sometimes ambiguous as for different KGs might have different naming convention and two different entities can also have same name. This problem only can be solved by entity summary.

In our dataset the number of attribute triples is almost equal to relational triples, so attribute embedding alone can not achieve reasonable performance, but EASAE leverages summary embedding to utilize the textual information of the entities and outperform the results of the existing state-of-the-art models. Moreover, predicate alignment module overcomes the dependency of pre-aligned seeds. Among the state-of-the-art models BootEA and MultiKE achieved the better results in DBP-WD and DBP-YG datasets respectively. We can see from the results, EASAE succeed to achieve significant improvement in the performance with the respect to JASE as well.

Hits@k is the most significant metric to evaluate KG embedding models. In the

Table 3.5: Result Analysis using AttrE-Dataset

Dataset	Model	MR	Hits@1	Hits@10
AttrE-Dataset	TransE	24809	1.22	3.54
	MTransE	7105	33.46	34.32
	JAPE	5296	33.35	33.37
	AttributeE	26	91.02	92.17
	EASAE	**24**	**93.35**	**97.89**
AttrE-Dataset (30% reduced attribute Triple)	AttributeE	1655	49.11	32.91
	EASAE	**57**	**87.22**	**92.75**

DBP-WD dataset, EASAE achieved the Hits@1 score of 92.04% and Hits@10 score of 97.83% . For DBP-YG dataset our model achieved the Hits@1 score of 90.86% and Hits@10 score of 96.64%. The results clearly outperform the latest state-of-the-art models.

Here we also discuss some very recent studies based on the key differences between those models and EASAE. COTSAE [52] showed high accuracy on DWY100K datasets but for DBP-WD dataset EASE achieved competitive performance compare to COTSAE while for DBP-YG their performance is much more satisfactory. Although we haven't conducted the experiment but we can surely claim that our method is simple enough to extend it to an iterative manner and if we add iterative method of using newly predicted aligned entities as training set for next iteration our result will certainly improve. Our results are already quite close so we strongly believe our model will work well in the iterative setting. CEAFF [94] showed very high accuracy on all datasets of DWY100K because entity names in DBpedia, YAGO and Wikidata are nearly identical, where string-level feature is extremely effective. In contrast, although semantic information is also useful, not all words in entity names can find corresponding entries in external word embeddings, which hence limits its effectiveness. They presented the result of CEAFF, where string-level feature is removed and our result is still comparable with that. BERT-INT [55] used the name as the basic representation of an entity for monolingual dataset experiment and CEAFF [94] also used a string based similarity for entity names. However, they both admit the names for most of the aligned entities are exactly the same in the datasets, therefore the accuracy is very high in both cases. We also managed to achieved 100% at Hits@10 while adding entity name matching feature

with our proposed EASAE model. For this particular dataset incorporating name information is very useful but in real world scenario it is not always practical to rely on the name only. For instance, in the field of Bio-informatics it is quite common for different knowledge bases to have the same things with different names. Another example can be, the entity "Washington" can refer to a person/state/capital. In such cases summary or textual description can lead to better performance as name matching could not add any significance. Our idea is not only useful for the above case but also efficient for sparse dataset according to the experimental results. Moreover, the structural characteristics of KGs and zero-shot cases are totally ignored in their model.

3.4.5 Experiment 3

EASAE model introduces summary embedding which overcomes the dependency of having large amount of attribute triples in the dataset. We were very interested to further investigate this impact. For that reason, we have designed another experiment using a different dataset (AttrE-Dataset) provided by AttributeE. We analyzed that dataset and observed each entity on AttrE-Dataset is associated with satisfactory number of attributes. In AttributeE, the authors showed a very efficient way to utilize attribute information to model entity alignment problems and they achieved state-of-art performance compare to recent models like MTransE and JAPE. EASAE model also utilizes the attribute information so having large number of attribute data is also helpful for our model as well. EASAE achieved better performance on AttrE-Dataset than all the baselines. However, we cannot guarantee large amount of attribute information for every datasets. In such cases, AttributeE may not meet the expected performance. We have seen this trail in our previous experiment (Experiment 2) as well. To further investigate the impact of summary embedding we randomly reduce 30% attribute triples from AttrE-Dataset and run the experiment again on EASAE and AttributeE model. As AttributeE already superseded the results of other mentioned models (TransE, MTransE and JAPE), so we think it is reasonable to compare our model with AttributeE only. In the new dataset (30% reduced attribute triples), the performance of AttributeE sharply falls down by almost 46% and 65% in terms of Hits@1 and Hits@10 respectively. We can see the impact on EASAE models' performance as well but in terms of Hits@1 and Hits@10 the performance is decreased by only around 6% and

5% respectively. Summary embedding is contributing to neutralize the effect of this attribute triples reduction. Furthermore, EASAE model uses a weighted averaging technique when merging three embedding models. It also helps to alleviate the model performance. When we could not get enough information from attribute embedding , EASAE emphasizes summary and relational embeddings. This results indicate that summary embedding can help to alleviate the model performance. The detailed results of the experiment is shown on Table 3.5.

3.5 Conclusion

In this paper, we introduced a joint entity summary and attribute embedding model along with relational structural embedding technique for entity alignment in between KGs. Our proposed model uses the BERT embedding model for entity summary embeddings and it is very effective in zero-shot scenario. We also discussed how our approach can overcome the flaws of the recently proposed entity alignment methods. We compared the performance of EASAE with the most recent state-of-the-art models. Our experimental results demonstrated that our approach achieved superior results than the baseline embedding approaches. For the future work, we intend to include more datasets to determine the effectiveness of our proposed model. Moreover, we plan to study the cross-lingual entity alignment in the same KG and between the different KGs e.g., DBpedia to Wikidata corss-lingual entity alignment as well. Recently, convolutional network based KG embedding achieved promising performance in KG embedding. We want to leverage the neural network based models for modelling the complex semantics of KGs and want to apply them in entity alignment task.

4

Attribute Enhancement using Aligned Entities between Knowledge Graphs

Knowledge graph (KG) is a structural form of semantic network that integrates triplets into a graph to support knowledge processing and reasoning. Popular KGs like DBpedia, YAGO, Freebase are mostly constructed based on crowd-sourced content and automatic extraction methods, therefore they are not always complete and error-free. In this study, we propose an Attribute Enhancement Framework (AEF) to enrich the attributes of entities by integrating multiple KGs based on their aligned information. Usually, similar entities from different KG contain different set of attributes. AEF exploits representation learning based ranking model to determine the significant attributes. Later it employs a similarity mapping method to integrate new attributes to the target KG. AEF also determines the attribute value inconsistency between two KGs. In our experimental dataset, 46% new attribute properties and 29% new values from DBpedia are proposed for the YAGO entities. With this study, we aim to include all the important attributes to the existing KGs towards more robust and complete knowledge

graph.

4.1 Introduction

The last decade has witnessed a surge of research, discussions and applications on knowledge graphs (KGs). Unlike traditional knowledge representation methods, a KG effectively and intuitively expresses the relation between entities and infers new knowledge about them. Therefore, KGs are very popular in various fields, such as semantic search, question and answering systems, personalized recommendation systems and decision support systems [69]. In knowledge graph, real world entities are represented in the form of a triple and triples either indicates relationship between other entities or the property of that entity which is called attribute. Entity's attribute property plays an important role in knowledge bases and attribute completion is a very important task for KG completion and quality control. However, only a handful work has been done in this field because of its complexity. This paper discusses about entity attribute property enrichment based on important attribute ranking and missing value inclusion using multiple KGs. Perhaps, this might be the first attempt to incorporate multiple Knowledge Graphs for attribute completion task.

Knowledge Graphs are constantly growing, providing more and more information as structured data. Most data sources have their roots in unstructured or semi-structured information available throughout the web. So it is very understandable that extracting data from unstructured or semi-structured information is error-prone. This is the reason we always experience missing values or inconsistency in data. Combining the information in different KGs can enrich the quality of knowledge graph though it is very challenging task.

Entity attributes carry meaningful information about that particular entity. As an example, in Figure 4.1, we can see the entity 'Hollywood' has been searched in three different search platforms (Google, Yahoo, Bing). In the search results, we get three different set of attributes. We can clearly understand, if the 'Area' or 'Population' property can be be added to all of the KG then it would be beneficial. However adding 'Incorporated' or 'Merged with Los Angeles' properties from Bing can make the entity information more complete but they are not as significant in compare to previous two properties. To deal with this challenge, our proposed method ranks the impactful

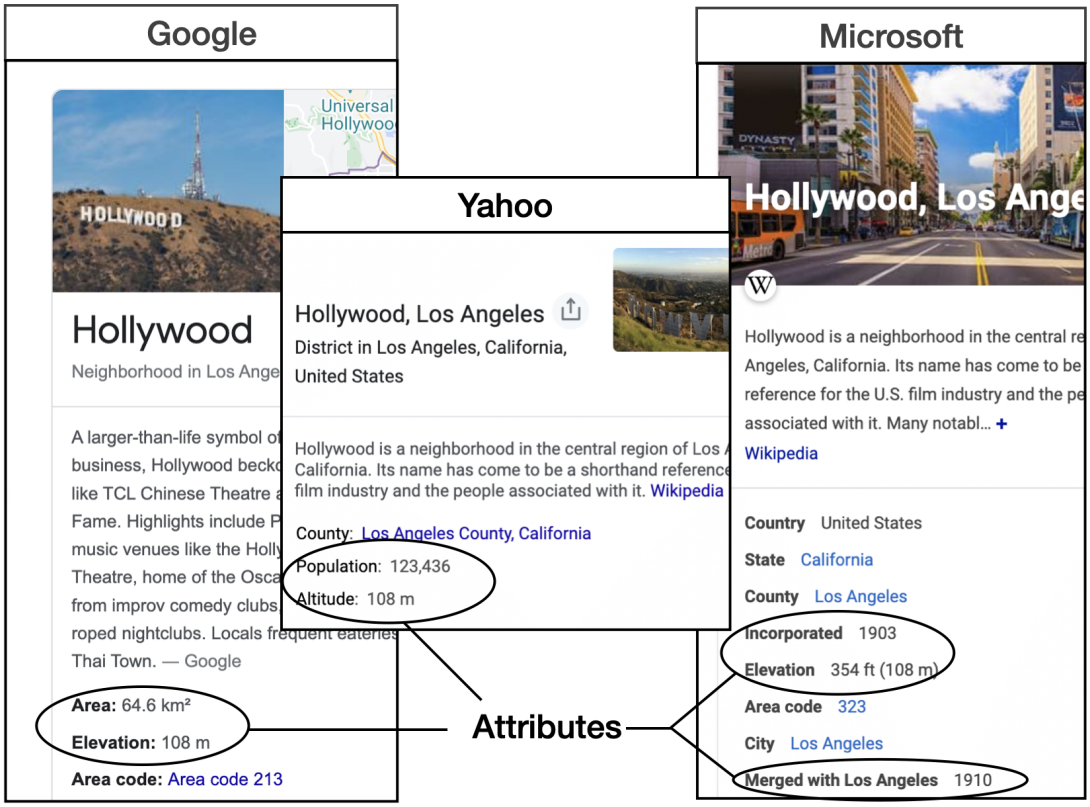


Figure 4.1: Different set of attributes from different KGs.

attributes and induced a way to find out missing attributes to enrich the KGs.

Attribute enhancement is a complicated task compared to entity type and relation prediction. One of the biggest challenges is to detect the missing attributes and values. Suppose, for a person type entity we can assume that it should contain an attribute similar to *birthdate*. However, we cannot claim that it may have marriage date or death date information. A living person will not have the property *deathdate* or likewise for an unmarried person entity. But at the same time it is possible that, the entity actually referring to an death person who was married as well and that information is missing to the KG. By using aligned entities from multiple KGs we can get an idea of the attribute which might be missing. Figure 4.2. describes a missing attribute scenario from YAGO. The *deathPlace* of 'Leonardo da Vinci' is present in YAGO but *birthdate* and *deathdate* are missing but his connecting entities like 'Albert Einstein' have these attributes.

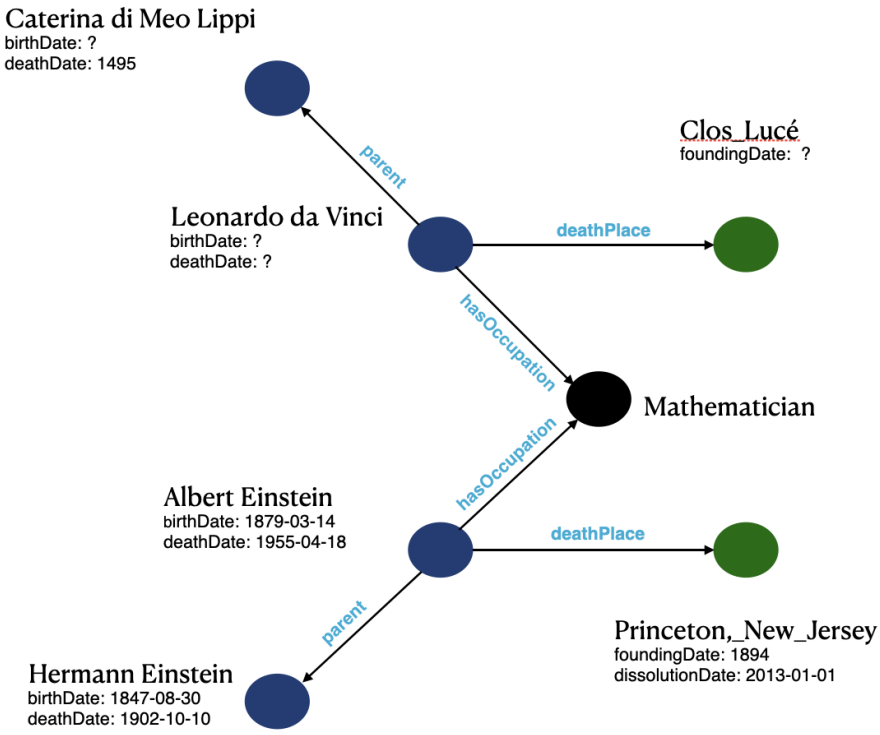


Figure 4.2: Missing attributes scenario in YAGO.

In this paper, we propose a method to rank entity attributes of a reference KG and based on that we can include new or missing attributes to the targeted KG. Our ranking method sorts the attributes based on importance and then adds the top-ranked attributes to the specific entity if they are missing.

We propose embedding and probabilistic approaches to tackle the ranking problem and later prepare an ensemble method to rank the important attributes. In this study, we address two downstream tasks. 1) New or Missing attribute proposing from reference KG to targeted KG. 2) Inconsistency detection of attributes. The first task utilizes both the ranking method and similarity match function together to recommend new or missing attributes. While our second task only leverages the similarity match function to detect the value inconsistency between two KGs.

The main contributions of this paper are:

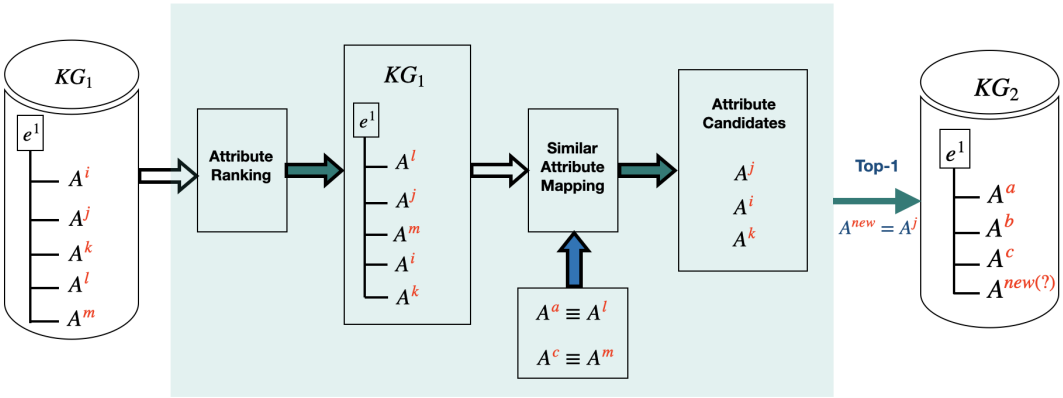


Figure 4.3: Model architecture of Attribute Enhancement Framework(AEF)

- We propose a framework (AEF) for attribute completeness using multiple KGs.
- Our method recommends attributes for target entities based on their aligned information from other KG. These recommended attributes might be new attributes that are not present in the target KG or missing attribute properties that are present in the KG structure but not present in the targeted entity.
- Our method can also detect the attribute value inconsistency between two KGs based on their aligned attributes.

4.2 Preliminary

In this section, we formally define the terms used in this paper and the problem as well.

Definition 4.1 *Knowledge Graph (KG): A knowledge graph KG = (E, R, T) , where E, R, T are the set of entities, relations and triples respectively.*

Definition 4.2 *Relational Triples: $T \subset E \times R \times E$ is a set of relational triples representing the relations between entities, where E and R is the set of all entities and relations respectively.*

Definition 4.3 *Attribute Triples: $A_T \subset E \times A \times L$ is a set of attribute triples representing the attributes of entities, where A is a set of all attributes, and each attribute $A_i \in A$ has a corresponding literal attribute value set $L_i \in L$.*

Definition 4.4 *Attribute Ranking: Given the pair composed by an entity and an attribute, we use the $attribute_{property}$ relation to generate a set of triples (e, $attribute_{property}$, a). The task is to identify a ranking function $f(e, a)$ that assigns a score to each entity-entity attribute pair and then sorts the attributes based on the scores computed for a specific entity.*

In our paper, we use bold lowercase letters to represent embedding vectors and bold uppercase letters to denote matrices.

4.3 Proposed Methodology

We propose an attribute recommendation framework for enhancing the attribute properties of an entity using multiple KGs. The overall methodology we propose is depicted in Figure 4.3. AEF proposes two major steps - (1) Attribute Ranking and (2) Similar Attribute Mapping. To recommend attribute property for an entity of KG_2 we take the aligned entity from KG_1 as the reference. We first rank the attribute properties of the entity from KG_1. Then we group the similar attribute property for KG_1 and KG_2. The purpose of this group is to avoid recommending duplicated properties that are already present in KG_2. Finally, the model would recommend *top@k* attributes as the potential candidate which KG_2 can add to enhance its entity.

Our model can also contribute for detecting erroneous or missing attribute values between the Similar Attribute property group.

4.3.1 Attribute Ranking

Attribute ranking method would rank the attribute properties of the entity from the reference KG. Entity attribute carry information about the different properties of an entity. As an example, in DBpedia entity 'Barack Obama' has 51 different attributes. Using those attribute we can understand the characteristic of the entity 'Barack Obama'. Out of all possible attributes attached to an entity in reference KG, we want to sample

out the most significant attributes for recommending them to it's similar entity from target KG. Thus, the task of ranking entity attributes by their importance becomes essential.

Here we have applied two different ranking methods - (1) Embedding Based Ranking Method (EBR); (2) Popularity Based Ranking Method (PBR).

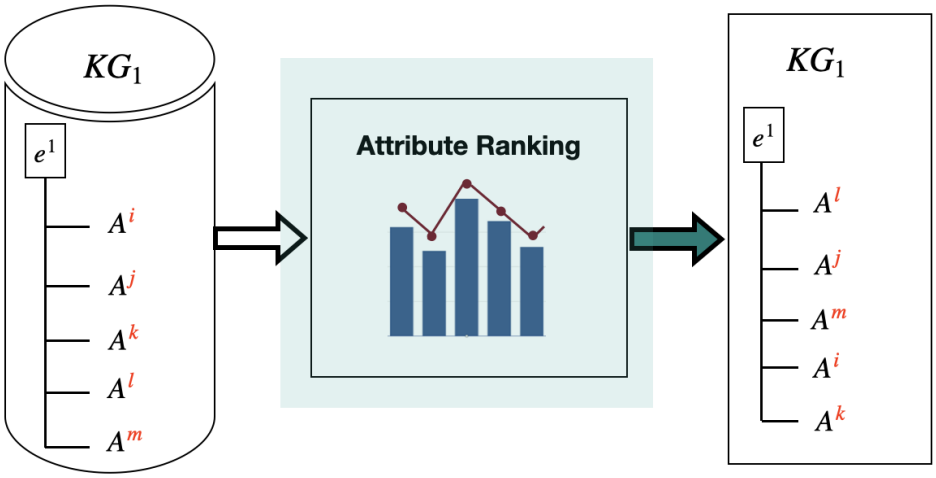


Figure 4.4: Attribute Ranking

Embedding Based Ranking (EBR)

To characterize the structure information in KG, we interpret a relationship as the translation from the head entity to the tail entity [20].

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 [95]. In our embedding-based ranking model, we employ the most basic translation distance based models for simplicity. This model ranks the entity attributes as follows:

TransE learns embedding as $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$ where (h, r, t) holds. Hence, $\boldsymbol{(h + r)}$ is very close to $\boldsymbol{t}$ [20]. The score function of TransE is:

$$f(\boldsymbol{h}, \boldsymbol{t}) = -|| \boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t} ||_{l_{1/2}} \tag{4.1}$$

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

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

$$f_{rel}(\boldsymbol{e}, \boldsymbol{a}) = -|| \boldsymbol{e} + \boldsymbol{attribute_{property}} - \boldsymbol{a} ||_{l_{1/2}} \tag{4.2}$$

In this paper, we leverage knowledge graph embeddings to model the entity attribute ranking task. 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 [20]. We show how the loss function based on the $attribute_{property}$ is the same for all the relations in the set of R.

$$L_{EBR} = \sum_{(e, attribute_{property}, a) \in S} \sum_{(e', attribute_{property}, a') \in S'} max\big(0, f_{rel}(\boldsymbol{e}, \boldsymbol{a}) + \gamma - f_{rel}(\boldsymbol{e}', \boldsymbol{a}')\big) \tag{4.3}$$

$f_{rel}(e, a)$ is the energy function score of the positive triple and $f_{rel}(e^{'}, a^{'})$ 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.

$$\begin{aligned} S^{'} = \{(e^{'}, attribute_{property}, a) | e^{'} \in E \wedge e^{'} \neq e \wedge \\ (e, attribute_{property}, a) \in S\} \cup \\ \{(e, attribute_{property}, a^{'}) | a^{'} \in A \wedge a^{'} \neq a \wedge \\ (e, attribute_{property}, a) \in S\} \end{aligned}$$

As the equation suggests, we randomly replace the heads and tails of positive triples with other entities in E and attribute in A, 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.

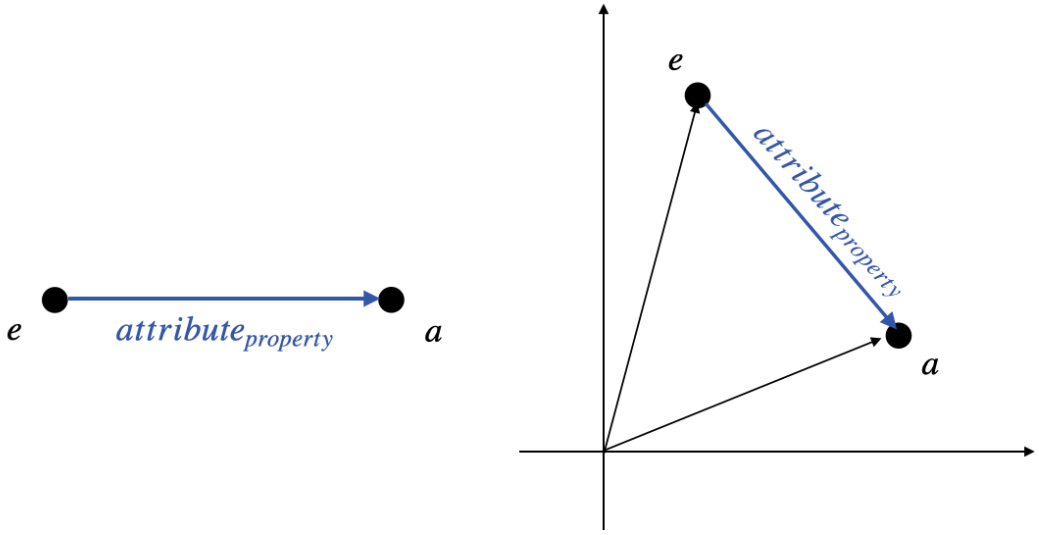


Figure 4.5: TransE based Attribute Property Embedding

Popularity Based Ranking (PBR)

In PBR model, we take the advantage of the probabilistic modeling [96, 97] of KGs but we keep the score function f simple. From the EBR models we learn that the simplest way to measure the plausibility of a triple $(e, attribute_{property}, a)$, by defining the score function, $f(\boldsymbol{e}, \boldsymbol{a}) = -|| \boldsymbol{e} + \boldsymbol{attribute_{property}} - \boldsymbol{a} ||_{l_{1/2}}$.

Our PBR model defines the conditional probability on triples as follows:

$$P_r(a|e, attribute_{property}) = \frac{exp\{f(\boldsymbol{e}, \boldsymbol{a})\}}{\sum_{\bar{a} \in T} exp\{f(\boldsymbol{e}, \bar{\boldsymbol{a}})\}} \tag{4.4}$$

Eq. (4) states the conditional probability of an entity attribute a where the $attribute_{property}$ relation and an entity are given over a fact/triple. In the same way, we can define $Pr(e|attribute_{property}, tp)$ and $Pr(attribute_{property}|e, a)$.

So the objective function is given as:

$$L_{PBR} = -\sum_{(e,attribute_{property},a)\in S} \big(\log P_r(e|attribute_{property}, a) + \log P_r(attribute_{property}|e, a) + \log P_r(a|e, attribute_{property}) \big) \tag{4.5}$$

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

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

$$L = L_{EBR} + L_{PBR} \tag{4.6}$$

L_{EBR} and L_{PBR} are loss functions for the embedding based ranking and the probabilistic ranking respectively. They are independent of each other and hence are optimized separately. We adopt stochastic gradient descent (SGD) to optimize the above loss functions.

We defined the entity attribute ranking problem formally in Definition 4. The task is to rank a set of n entity attributes 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 attribute pairs with a " $attribute_{property}$" property to the knowledge graph together with other existing relations.

Once this is achieved, the model learns the knowledge graph embedding, which includes all the triples as well as the " $attribute_{property}$" property triples. This model is evaluated over queries of the form $(e, attribute_{property}, ?)$ as a task of ranking a based on the rank of gold entity type $a*$. 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 attributes.

4.3.2 Similar Attribute Mapping

The purpose of our model is to enhance attribute property on an entity with the help of its corresponding entity from the other KG. In the above section we have ranked the candidate attribute. However we do not intend to recommend the attributes which are already present for the targeted entity. As we are dealing with two different KGs and they have ontological differences, so we cannot match similar attributes by simple string matching or such method.

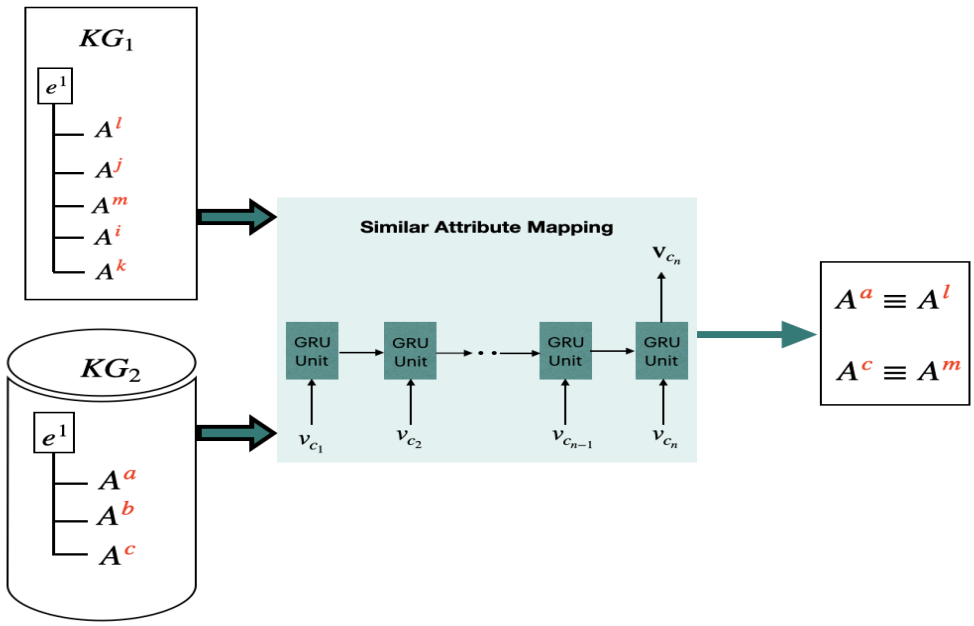


Figure 4.6: Attribute Similarity Mapping

Well designed ontologies are used to construct Knowledge Graphs and the terms of ontology properties are mostly meaningful individual words [80]. Word embedding is better to deal with such words rather than simple string matching.

Continuous Bag-of-Words (CBOW)

Continuous Bag-of-Words (CBOW) model is one of the most popular neural network models for learning distributed representations of words, in other words, embedding words in a vector space (word2vec) [91]. In this paper we use CBOW to learn the representation of words. Given a sequence of training words w_1, w_2, ... , w_n, and a context window c, the learning function of CBOW model is to maximize the following average log probability:

$$\frac{1}{n}\sum_{t=1}^{n} log\ p(w_t|w_{t-c}, .., w_{t-1}, w_{t+1}, ..., w_{t+c})$$

where $p(w_t|w_{t-c}, .., w_{t-1}, w_{t+1}, ..., w_{t+c}) = \frac{exp(\bar{v} \cdot v_{w_t})}{\sum_{w=1}^{|V|} exp(\bar{v} \cdot v_w)}$

Here, v_w is the distributed vector representation of word w, $\bar{v}$ is an average of the distributed representation of words in the context c, and V is the set of vocabularies.

Based on our observation, we found that the terms of the ontology properties are mostly compound words, e.g., *birthDate* or *dateOfBirth*. The estimation of the representation of compound words using representations of individual words is required. Since distributed representation of a compound word cannot be directly represented as a sum of individual vectors, we apply Recurrent Neural Networks (RNN) based approach, specifically, GRU-based [98] approach to represent compound words as introduced in the study.

Gated Recurrent Unit (GRU)

Recurrent Neural Networks (RNN) has shown promising results in processing arbitrary sequences of input. For a given sequence of input RNN model learns the current latent state with the input data at time t and the previous latent state at time $t-1$. Then the current latent state is used to predict the output.

Although RNN is able to handle a variable-length sequence input, long-term dependencies are difficult to be captured due to the gradients tend to either vanish or explode. The long short-term memory (LSTM) unit and gated recurrent unit (GRU) are able to handle long-term dependencies and perform better than using traditional $tanh$ unit [37]. We choose the GRU instead of LSTMs for training compound words from Wikipedia articles because GRU use fewer parameters and they are almost similar in terms of performance.

Given a compound word w_c, which is a sequence of individual words w_{c_1}, w_{c_2}, ..., w_{c_n}, the GRU-based model predicts the distributional representation of the compound word $\hat{w_c}$, which is the output of the last state. In the learning process, we need to minimize the following error function $L(w_c, \hat{w_c})$:

$$L(w_c, \hat{w_c}) = ||\mathbf{v}_{w_c} - \hat{\mathbf{v}}_{w_c}||_2$$

where $\mathbf{v}_{w_c}$ is the distributed representation of the input compound word w_c, and $\hat{\mathbf{v}}_{w_c}$ is the predicted representation of the compound word using GRU-based approach.

Table 4.1: Dataset Statistics.

Datasets		Entity	Relation Triples	Attribute Triples	Relational Property	Attribute Property
DBP-WD	DBpedia	100,000	463294	381166	330	351
	Wikidata	100,000	448774	789815	220	729
DBP-YG	DBpedia	100,000	428952	451646	301	334
	YAGO	100,000	502563	118376	29	23

The GRU unit we adopted is defined as follows:

$$
\begin{aligned}
z_t &= f(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= f(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= g(W_h x_t + U_h(r_t \circ s_{t-1}) + b_h) \\
s_t &= z_t \circ s_{t-1} + (1 - z_t) \circ h_t
\end{aligned}
$$

where x_t is the input vector at time t, z_t is the update gate vector at time t, r_t is the reset gate vector at time t, h_t is the hidden layer vector at time t, s_t is the output vector at time t, W and U are parameter matrices, b is bias parameter, f and g are activation functions, and $\circ$ is the Hadamard product operation.

Since we use the output of last state $\hat{\mathbf{v}}_{w_c}$ as the predicted distributed representation of compound word w_c, $\hat{\mathbf{v}}_{w_c}$ is calculated as follows:

$$
\hat{\mathbf{v}}_{w_c} = s_{Last} = z_{Last} \circ \ s_{Last-1} + (1 - z_{Last}) \circ h_{Last}
$$

In most of the cases, GRU-based approach assigns higher cosine similarity to the semantically similar attribute pairs than CBOW. For example, for the pair of *dbo:birthDate* and *wiki:Date of birth*, GRU-based similarity is 0.8542 while the CBOW-based similarity is 0.5178. Therefore, we have used GRU-based for our final evaluation.

Table 4.2: Recommended Attribute From DBpedia

New Attributes		
	Top@1	Top@2
DBpedia -> YG	19%	28%
DBpedia -> Wikidata	8%	12%
Missing Attributes		
	Top@1	Top@2
DBpedia -> YG	21.85%	31.74%
DBpedia -> Wikidata	17.43%	23.65%

4.4 Experiments

4.4.1 Datasets

To evaluate our model, we have used total two datasets, which are shown in Table 1. They are recently introduced by BootEA [4].These two datasets were sampled from DBpedia [6], Wikidata [99] and YAGO [8], each of which contains 100,000 aligned entity pairs. The statistics of the datasets is given on Table 1.

Although the datasets were introduced for entity alignment task. But they were perfect fit for our experiment as well. For our experiment we have considered DBpedia as the reference KG and predict the attributes for the entities from other KG.

4.4.2 Implementation

SPARQL Protocol and RDF Query Language (SPARQL) [100] is used to access the KGs to retrieve the existing attributes of the entities. We also use SPARQL queries to get attribute values for inconsistency checking.

Wikipedia articles are used to train the distributed representation of individual words and compound words using GRU-based word embedding. The average length of compound words is 3.06 tokens and 16,369,076 compound words were used for

the GRU-based approach. We used word2vec provided by gensim [101] to learn the distributed representation of words and used tensorflow [102] to implement RNN model using GRU unit. We used CBOW with dimension size as 200 and window size as 5. We discarded words that appear less than 25 times and used the token “UNK" to replace out of vocabulary words.

4.4.3 Attribute Enhancement

We have considered DBpedia as the reference Knowledge graph for the experiment. AEF runs on the whole DBpedia dataset and generate the ranked lists of attributes for each entities with the help of our ranking module. Then each aligned entity pair (e_{KG_1}, e_{KG_2}) goes through the similar attribute matching module to generate the list of aligned attributes between two entities. Please note that the datasets exploited in this paper consist of aligned entities between multiple KGs (refer to Sect. A). Finally attributes other than the aligned set are considered as candidates for attribute recommendation for the targeted entity. This way our method avoid recommending the attributes which are already present in the targeted entity. From the candidate list top K attributes are recommended. In this experiment we present the number of new attributes proposed by the model with $K = 1$ and $K = 2$ in Table 4.2. From the Table 4.2 we can observe for YAGO our model was able to enrich 19% and 28% new attributes for Top@1 and Top@2 recommendations respectively. On the other hand, for Wikidata we achieved 8% and 12% new attributes enrichment for Top@1 and Top@2 recommendations. Wikidata has more rubost attributes than DBpedia, therefore YAGO got more number of attribute recommendations than Wikidata.

Data extraction and KG enrichment are very completed tasks to build an effective KG. All the KGs itertively etract and collect information which separate to each other (KGs). Entity attribute information can help the KG towards completeness interms of entity related critical facts. Our study inspires gathering important attribute values from multiple knowledge graphs which may lead to attribute completeness in traditional KGs without reinventing the wheel over and over again.

Apart from the new attributes we have found the entities where recommended attributes that are present in KG but not present for that particular entity. As shown in Figure **??**, the entity ’Leonardo da Vinci’ from YAGO does not have deathDate as its

attribute. But YAGO contains property 'schema:deathDate'. In such cases our model get deathDate property as recommended property from DBpedia for the 'Leonardo da Vinci' entity. Table 4.5 present a sample of attributes properties that falls into this criteria. Using our model we found 21.85% and 31.74% missing attributes for Yago for Top@1 and Top@2 recommendations respectively. In the same way we have observed 17.83% and 23.65% missing attributes for Wikidata entities respectively. Our proposed model can address new attributes recommendation and missing attributes filling at the same time.

One of the major challenges in this field is to evaluate the result due to lack of gold standard datasets. In order to evaluate our system we have selected 'birthDate' attributes based on its importance and tried to evaluate the performance of our system based on those attribute property. We have found that for person type entities, the most frequent attribute is birthDate. Among 2,230,135 dbo:person type entities in DBPedia, 2,131,931 entities contains birthDate. To test the quality of the recommendation for missing attribute, we remove YAGO:Birthdate from person type entities from our $DBpedia \rightarrow YG$ dataset and using proposed method we could predict 98% correctly. Evaluating new recommended attributes are even more difficult as we don't have any reference for that. So we have tried to show the statistical importance of the new attribute. As instance for Location type entities one of highest ranked recommended attribute is “area" which is intuitively very important feature to describe the property of a particular location. So undoubtedly the recommending this attribute can help to enrich the target KG by a lot.

We have further analyze the new recommended attributes based on different entity types. In Table III, we have showed entity type wise distributions of the recommended new attributes. For the experiment dataset, we observed that location type entities were required more attributes recommendations than the other types. The proper explanation of it might be: In our experimental dataset, location type entities from YAGO lacks many important attributes which DBpedia uses to describe the property of location type entities. We have gathered some examples of the newly suggested attributes for better understanding.

Table 4.3: Type wise distribution of recommended attributes

Entity Types	Recommended New Attributes (%)	
	Top@1	Top@2
Place / Location	24%	37%
Person	11%	18%
Film /Drama	15%	23%
Sport	5%	11%
Organization	5%	13%

4.4.4 Inconsistency Detection of Attributes

We have introduced a similarity mapping function for the attributes mapping between reference KG and target KG. This way we avoid recommending duplicated attributes for target entity. We also found a similar group of attributes between two KG. In theory, for aligned entity pairs, similar attribute properties should have exact similar values. But due to the ontological differences between two Knowledge Graphs attributes values are not always aligned mostly due to the format differences. However, even if both KG kept the attribute values in the same format, we still can find some inconsistency in values between two KGs. In Table 4.5. we have sampled few aligned attributes from DBpedia and YAGO. They all have similar format for the attribute value but there is some inconsistencies in their values. This phenomenon occurs when the information is extracted from different sources and one of those source are erroneous. For example, the entity "Willem Schellinks" (former Dutch painter) appears in DBpedia and YAGO. In DBpedia his **birthDate** is reported as $1623-02-07$ while in YAGO it says his **birthDate** is $1627-02-02$. Evidently, two values are very different. Using our proposed model we can detect such attribute value inconsistency as well. Although this is not our primary goal but still this is very useful discovery for overall KG completion and quality enhancement. In our sampled dataset, we have found such phenomenon as well and table V showed the percentage of value inconsistencies for some of the attributes.

Table 4.4: Type wise example of new attributes recommended for YAGO

Type	Example of New Attributes
Place/Location	area populationTotal postalCode utcOffset ...
Person	title activeYearsStartYear ...
Film	runtime releaseDate
Sport	squadNumber
Organization	office

4.5 Conclusion

Attribute enhancement is an important problem in knowledge graph community, and solving it would improve the quality and the usefulness of the KGs to a great extent. In this paper, we presented an Attribute Enhancement Framework (AEF) to enrich entity attribute information using aligned entities between multiple knowledge graphs. We also designed two kinds of strategies to rank the important attributes before recommending new attributes to the target KG. Proposed AEF can also detect the attribute value inconsistency between two KGs based on their aligned attributes. Our experiments on two real-world datasets demonstrated the effectiveness of our

Table 4.5: Statistics of similar attribute group

Similar Attribute Group	Value Inconsistency(%)
birthDate	5.02%
deathDate	9.45%
synonym	7.32%
foundingDate	6.7%
elevation	3.43%

framework. In future work, we plan to investigate more larger dataset and propose graph neural network based attribute ranking method to improve the quality of proposed AEF.

5 Discussion

In this Chapter, the achievements, discussion, and some limitations of the entity alignment task and attribute enhancement task are discussed, respectively. Firstly, EASAE, which tackles aligning entities' tasks between multiple KGs resources, is discussed in Section 5.1. Then, the discussion of AEF for the attribute enhancement task is presented in Section 5.2.

5.1 Entity Alignment

We introduced EASAE, a joint entity summary and attribute embedding model along with relational structural embedding technique for entity alignment in between KGs. The generic approach of embedding-based entity alignment typically consists of three components, a merging module, an embedding module, and an alignment module. The embedding module and the alignment module can be trained separately or jointly, and these two together compose the training modules for entity alignment. To run the embedding module, it is necessary to bring the entities and relations from both KGs

Table 5.1: Latest Entity Alignment Models and their generic variations.

Technique	KG embedding	KG structure structure	Attributes as input features	Merging feature for embedding module
MTransE (2017)	TransE	Triple	-	Seed entity alignments;Seed relation predicate alignments
IPTransE (2017)	PTransE	Path	-	Seed entity alignments;Seed relation predicate alignments
JAPE (2017)	TransE	Path	Data type of attribute value	Seed entity alignments;Seed relation predicate alignments
BootEA (2018)	TransE	Triple	-	Seed entity alignments
KDCoE (2018)	TransE	Triple	Entity description	Seed entity alignments
TransEdge (2019)	TransEdge	Triple	-	Seed entity alignments
AttributeE (2019)	TransE	Triple	Attribute triple as relation triple; String of attribute predicate/value;	Predicate alignments
MultiKE (2019)	TransE	Triple	Attribute triple as relation triple; String of attribute predicate/value; Entity name	Seed entity alignments; Relation/attribute predicate alignments
COTSAE (2020)	TransE	Triple	Character sequence of attribute value/predicate	Seed entity alignments
EASAE (2021)	TransE	Triple	Attribute triple as relation triple; String of attribute predicate/value; Entity description	Predicate alignments
GCN-Align (2018)	GCN	Neighborhood	Attribute triple as relation triple	Seed entity alignments
RDGCN (2019)	DPGGNN	Neighborhood	Entity name	Seed entity alignments
GMNN (2019)	GCN	Neighborhood	Entity name	Seed entity alignments
MuGNN (2019)	GCN	Neighborhood	-	Seed entity alignments;Seed relation predicate alignments
CEA (2020)	GCN	Neighborhood	Entity name	Seed entity alignments
AliNet (2020)	GAT	Path		
AttrGNN (2020)	GAT	Neighborhood	String of attribute value; Entity name	Seed entity alignments

Table 5.2: Comparative Analysis with Latest Entity Alignment Models

Technique	Prior alignment	Zero-shot handling	Robustness
Translation Based Models			
JAPE	Seed and Predicate	No	Dataset dependent
BootEA	Seed	No	Dataset dependent
TransEdge	Seed	No	Dataset dependent
AttributeE	Predicate	No	Dataset dependent
MultiKE	Seed	Partial[1]	Robust when entity names are synonymous
COTSAE (2020)	Seed	No	Dataset dependent
EASAE	Predicate	Yes	Robust when entity textual information is available
Neural Network Based Models			
GCN-Align	Seed	No	Robust when entity names are synonymous
RDGCN	Seed	No	Mostly relies on entity name similarity
GMNN	Seed	No	Mostly relies on entity name similarity
MuGNN	Seed and Predicate	Partial[1]	Robust when entity names are synonymous
CEA	Seed	Partial[1]	Robust when entity names are synonymous
AttrGNN	Seed	No	Dataset dependent

into a unified vector space. Recent studies rely on either pre-aligned seed between KGs or predicate alignment. Some papers proposed customized combination of both seed and predicate alignment. However, previously aligned seeds between KGs is very limited. Predicate alignment is a good substitute for entity alignment task and we applied it in EASAE. For the embedding module mostly translation based embedding or GNN based embedding are used. However, TransE is the most common translation based model which is widely used for entity alignment due to its simplicity and efficiency. There is another important variation exist, in the way the attributes are used in embedding module. Many studies have been conducted on how to exploit attribute information efficiently for entity alignment task. But, the heterogeneity between KGs makes it challenging. Because for similar entity pair from different KGs usually contains different attribute sets and which makes the alignment task very challenging. Recent paper are proposed using additional properties like entity name or textual description to deal with this problem. For some particular datasets incorporating name information is very useful but in real world scenario it is not always practical to rely on the name only. For instance, in the field of Bio-informatics it is quite common for different knowledge bases to have the same things with different names. Another problem is, name can be very ambiguous, many people or place or object can share the same name. Textual description is a safer choice but relying only description is not enough, that's why we include relation, attribute and description together to get the best outcome. For alignment model there are few variations exist. Some models use bootstrapping while other choose not to because it adds more complexity. Another problem with embedding based module is the zero shot scenario. When the entity is missing from the training set any embedding module suffers from its performance. Table 5.1 presents an elaborated comparison between recent studies and Table 5.2 presents a comparative analysis based on their performance w.r.t to exiting challenges.[1]

The entity alignment task mostly relies on entities' relational structure and attribute property. However, considering only relations and attributes fail when the entities have a fewer number of attributes or when the relational structure can't capture the meaningful representation of the entities. Also, synonymous entity pairs in different KG usually contain a different set of relations and attributes. Adding summary information can be very helpful in tackling these problems. Our method shows where

[1] When entity names are synonymous

Table 5.3: Key statistics for popular KGs. (2016) [5]

	DBpedia	Freebase	Wikidata	YAGO
Number of triples	411885960	3124791156	748530833	1001461792
Number of relations	58776	70902	1874	106
Number of entities	4298433	49947799	18697897	5130031
Unique non-literals as attribute	83284634	189466866	101745685	17438196
Unique literals as attribute	161398382	1782723759	308144682	682313508

a summary is available, it can help in completing the drawbacks of relations and attributes. Also based on the dataset, our proposed method can dynamically adapt to perform accordingly and thus making it more robust. We considered the textual description of entities as the summary and applied BERT embedding model to these summaries for entity summary embedding. We adapt TransE-based translation model for attribute embedding and relational structural embedding. EASAE is very effective in zero-shot scenarios by utilizing the description. EASAE shows the effectiveness of entity summary embedding to align entities while a very small number of attribute triples exist. Additionally, our method can dynamically adapt based on the dataset features to perform accordingly and thus making it more robust. We have performed experiments with two different datasets which are different in nature. One dataset is very dense in relation, and the other is rich with attributes. In both cases, our model performs well. Moreover, in our observation, textual descriptions are fairly common properties that KG contains for its entities. For example, financial domain datasets are quite different from benchmark datasets, but usually, they contain textual descriptions with them. So our model can easily be extended for these kinds of datasets as well. If both or either relation and attribute properties are not so well defined in those datasets, still our summary feature would complement such scenarios. So it is understandable that our idea is not limited to certain datasets or scenarios. We compared the performance of EASAE with the most recent state-of-the-art models. Our experimental results demonstrated that our approach achieved superior results than the baseline embedding approaches.

5.2 Attribute Enhancement

Attribute enhancement is an important problem in knowledge graph community, and solving it would improve the quality and the usefulness of the KGs to a great extent. We have listed some key statistics of popular KGs in Table 5.3. We can clearly see the difference in various aspects of different KGs. Thus integrating existing KG can complement each other significantly. However, transferring knowledge from one KG to another is very challenging due to the heterogeneity of KGs. Only a handful of work has been done for attribute enhancement because of its complexity. In this paper, we presented an Attribute Enhancement Framework (AEF) to enrich entity attribute information using aligned entities between multiple knowledge graphs. We also designed two strategies to rank the important attributes before recommending new attributes to the target KG. Proposed AEF can also detect the attribute value inconsistency between two KGs based on their aligned attributes. Our experiments on two real-world datasets demonstrated the effectiveness of our framework. In future work, we plan to investigate larger datasets and propose a graph neural network based attribute ranking method to improve the quality of the proposed AEF.

We believe that our work will foster ongoing research works on entity alignment and attribute enhancement. A complete KG can improve the search engine along with other downstream applications of KG e.g., recommender system, customer profiling, etc., which is the ultimate goal of the research around KG.

6 Conclusion

In this thesis, we investigated, designed, and evaluated a number of methods and algorithms to exploit multiple knowledge graphs and increase the quality of their data. Our work contributed to advancing the state-of-the-art in several tasks related to knowledge graph alignment and knowledge graph enhancement. We also studied how multiple KGs can benefit the overall quality control of knowledge graphs.

6.1 Summary

The vision of our research is to integrate knowledge from various knowledge graphs that can improve interoperability among different data sources. However, due to the lack of aligned information between KGs and their heterogeneous properties, we cannot readily integrate multiple knowledge graphs. The goal of our research is to solve the two proposed problems This research aims to address two well-known problems to construct a more complete and robust KG by utilizing already existing KGs from various domains. In this thesis, we introduced an entity alignment framework

that can deal with the problem of limited existing aligned information and propose a more efficient framework for aligning entities from different KGs by applying an embedding-based method. Moreover, we propose an attribute enhancement framework for adding new and missing attributes between the aligned entities. Because, even though the entities are aligned, their relation, type, and structure are different. It is due to their respective KG architecture. Therefore, we need an impactful system that can deal with this heterogeneity problem between aligned entities and enhance the entities' attributes.

6.2 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 sections, we present some compelling ideas that could be pursued as an extension of this work and that can help in advancing the current state of knowledge graph technologies and semantic web applications.

- **Future Work for Entity Alignment**
  - Our proposed model is built on top of translation-based KG embedding models. Recently, convolutional network-based KG embedding achieved promising performance in KG embedding. We want to leverage the neural network-based models for modeling the complex semantics of KGs and want to apply them in entity alignment task.
  - For future work, we intend to include more versatile datasets to determine the effectiveness of our proposed model. Moreover, we plan to study the cross-lingual entity alignment in the same KG and between the different KGs e.g., DBpedia to Wikidata cross-lingual entity alignment as well.
  - Another intriguing future work is to adopt a bootstrapping training procedure. For instance, the resulting alignments may be used as additional data for an iterative training method to improve overall performance.
- **Future Work for Attribute Enhancement**

- In our proposed framework, we have exploited very simple ranking approaches. In the future, we would like to expand the ranking models using neural network-based models. Current method focuses more on embeddings and limited features. Other important features like "type" information might be helpful here. Entity type plays an important role on entity attributes. We intend to incorporate type-embodied attribute ranking models in your proposed framework.
- AEF exploited word2vec and GRU-based techniques for aligning attributes between KGs which is the simple way to tackle this problem. In many cases, word2vec or GRU-based system might fail to identify the similarity. We have an improvement opportunity here to work on. In near future, we want to do more study in this module and propose an efficient model for it.
- Similar kinds of ideas that we proposed for attributes also can be applied for type ranking and entity type completion. Entity type expresses an entity more meaningfully. Ranking entity types and including various types in the KG can benefit many downstream tasks, e.g., entity profiling, recommendations, etc.

6.3 Outlook

Knowledge graphs are becoming an increasingly popular way of thinking about and organizing data within significant 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. Knowledge Graph plays an important role in many modern applications, e.g. question answering, browsing knowledge, structured search, and data visualization. Integrating knowledge from Multiple knowledge graphs is helpful to supplement the knowledge that is missing in the knowledge source. This will improve both the quality of the KG and the performance of the application it's using. In this dissertation, we

aimed to achieve a model that can align two heterogeneous KGs and then integrate knowledge based on the aligned information to complement each other.

Bibliography

[1] Xin Rong. word2vec parameter learning explained. In *Computing Research Repository (CoRR) abs/1411.2738*, 2014.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[3] Rui Zhang, Bayu Distiawan Trisedya, Miao Li, Yong Jiang, and Jianzhong Qi. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *The Very Large Data Bases Journal*, 31(5):1143–1168, 2022.

[4] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4396–4402. ijcai.org, 2018.

[5] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9(1):77–129, 2018.

[6] 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.

[7] 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*, pages 1247–1250. ACM, 2008.

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

[9] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.*, pages 601–610. ACM, 2014.

[10] M. Färber. The microsoft academic knowledge graph: A linked data source with 8 billion triples of scholarly data. In *Proceedings of the 18th International semantic web conference*, pages 13–129. Springer, 2019.

[11] Kathuria M, Nagpal C, and Duhan N. Journey of web search engines: Milestones, challenges and innovations. *International Journal of Information Technology and Computer Science*, 12:47–58, 2016.

[12] Rumana Ferdous Munne and Ryutaro Ichise. Entity alignment via summary and attribute embeddings. *Logic Journal of the IGPL*, 2022.

[13] John P. McCrae Paul Buitelaar Anja Jentzsch Andrejs Abele and Richard Cyganiak. Linking open data cloud diagram. https://lod-cloud.net/, November 3 2022.

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

[15] 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.

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

[17] 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*, pages 1419–1428. ACM, 2016.

[18] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A

web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data-mining*, pages 601–610, 2014.

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

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

[21] 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 Conference on Artificial Intelligence*, volume 15, pages 2181–2187. AAAI Press, 2015.

[22] 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 Conference on Artificial Intelligence*, volume 14, pages 1112–1119. AAAI Press, 2014.

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

[24] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of The 32nd Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence,*, pages 1819–1826. AAAI Press, 2018.

[25] 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*. OpenReview.net, 2019.

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

[27] 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, pages 271–280. ACM, 2012.

[28] 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.

[29] 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*, 18(1):4735–4772, 2017.

[30] 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*, pages 5185–5194. Association for Computational Linguistics, 2019.

[31] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 27th Annual Conference of Advances in neural information processing systems*, pages 926–934, 2013.

[32] 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.

[33] 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 conference on artificial intelligence*. AAAI Press, 2018.

[34] 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*, pages 978–987, 2019.

[35] 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 Conference on Artificial Intelligence*, pages 3009–3016. AAAI Press, 2020.

[36] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations,*, 2013.

[37] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Computing Research Repository (CoRR) abs/1412.3555*, 2014.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. volume 12, pages 2451–2471, 2000.

[40] S. Francois, L. Y. Francois, and Z. Chuguang. Rdf-ai: an architecture for rdf datasets matching, fusion and interlink. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop*, 2009.

[41] Volz Julius, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk-a link discovery framework for the web of data. In *Proceedings of Linking Data on the Web Workshop*, page 538, 2009.

[42] Y. Raimond, C. Sutton, and M. B. Sandler. Automatic interlinking of music datasets on the semantic web. In *Proceedings of Linking Data on the Web Workshop*, 2008.

[43] Suchanek FM, Abiteboul S, and Senellart P. Paris: Probabilistic alignment of relations, instances, and schema. In *Proceedings of International Conference on Very Large Data Bases*, 2011.

[44] Ngomo ACN and Auer S. Limes: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop*, 2011.

[45] M. Pershina, M. Yakout, and K. Chakrabarti. Holistic entity matching across knowledge graphs. In *Proceedings of International Conference on Big Data*, page 1585–1590, 2015.

[46] E. Rivas and S. R. Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *Journal of Molecular Biology*, 285(5):2053–2068, 1999.

[47] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In

Proceedings of the 26th International Joint Conference on Artificial Intelligence, pages 1511–1517. ijcai.org, 2017.

[48] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4258–4264. ijcai.org, 2017.

[49] Zequn Sun, Wei Hu, and Chengkai Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of Proceedings of the 17th International semantic web conference*, pages 628–644. Springer, 2017.

[50] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the 33rd Association for the Advancement of Artificial Intelligence conference on artificial intelligence*, pages 297–304. AAAI Press, 2019.

[51] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. Multi-view knowledge graph embedding for entity alignment. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5429–5435. ijcai.org, 2019.

[52] Kai Yang, Shaoqin Liu, Junfeng Zhao, Yasha Wang, and Bing Xie. Cotsae:co-training of structure and attribute embeddings for entity alignment. In *Proceedings of the 34th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, pages 3025–3032. AAAI Press, 2020.

[53] Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. A joint embedding method for entity alignment of knowledge bases. In *Proceedings of the 1st China Conference on Knowledge Graph and Semantic Computing*, pages 3–14, 2016.

[54] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3998–4004. ijcai.org, 2018.

[55] Xiaobin Tang, Jing Zhang, Bo Chen, Yang Yang, Hong Chen, and Cuiping Li. Bert-int: A bert-based interaction model for knowledge graph alignment. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 3174–3180. ijcai.org, 2020.

[56] Anna Lisa Gentile, Petar Ristoski, Steffen Eckel, Dominique Ritze, and Heiko

Paulheim. Entity matching on web tables: a table embeddings approach for blocking. In *Proceedings of the 20th International Conference on Extending Database Technology*, pages 510–513. OpenProceedings.org, 2017.

[57] Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*, page 349–357. ACL, 2018.

[58] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[59] K. Xu, liwei wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. In *Proceedings of Association for Computational Linguistics*, page 3156–3161, 2019.

[60] Y. Cao, Z. Liu, C. Li, J. Li, and T.-S. Chua. Multi-channel graph neural network for entity alignment. In *Proceedings of Association for Computational Linguistics*, page 1452–1461, 2019.

[61] L. Galarraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Exploring and evaluating attributes, values, and structures for entity alignment. In *Proceedings of the Very Large Data Bases*, page 707–730, 2015.

[62] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *Proceddings of International Conference on Learning Representations*, page 707–730, 2018.

[63] Z. Sun, C. Wang, W. Hu, M. Chen, J. Dai, W. Zhang, and Y. Qu. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *Proceedings of the 34th Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, page 222–229. AAAI Press, 2020.

[64] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of International Joint Conference on Artificial Intelligence*, page 5278–5284, 2019.

[65] Weixin Zeng, Xiang Zhao, Jiuyang Tang, and Xuemin Lin. Collective entity alignment via adaptive features. In *Proceedings of International Conference on Data Engineering*, pages 1870–1873, 2020.

[66] A.E Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *International Journal of Game Theory*, 36(3):537–569, 2008.

[67] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. A vectorized relational graph convolutional network for multi-relational network alignment. In *Proceedings of International Joint Conference on Artificial Intelligence*, page 4135–4141, 2019.

[68] Zhiyuan Liu, Yixin Cao, Liangming Pan, Juanzi Li, Zhiyuan Liu, and Tat-Seng Chua. Exploring and evaluating attributes, values, and structures for entity alignment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6355–6364, 2020.

[69] Xiangyu Wang, Lyuzhou Chen, Taiyu Ban, Muhammad Usman, Yifeng Guan, Shikang Liu, Tianhao Wu, and Huanhuan Chen. Knowledge graph quality control: a survey. *Fundamental Research*, 1(5):607–626, 2021.

[70] Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539, 2011.

[71] André Melo and Heiko Paulheim. Detection of relation assertion errors in knowledge graphs. In *Proceedings of the Knowledge Capture Conference*, pages 1–8, 2017.

[72] Heiko Paulheim and Christian Bizer. Improving the quality of linked data using statistical distributions. In *Proceedings of the International Journal on Semantic Web and Information Systems*, pages 63–86, 2014.

[73] Heiko Paulheim and Aldo Gangemi. Serving dbpedia with dolce–more than just adding a cherry on top. In *Proceedings of the 14th International semantic web conference*, pages 180–196, 2015.

[74] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in DBpedia. In *Proceedings of the 11th European Semantic Web Conference*, pages 504–518, 2014.

[75] Fleischhacker Daniel, Heiko Paulheim, Volha Bryl, Johanna Völker, and Christian Bizer. Detecting errors in numerical linked data using cross-checked outlier detection. In *Proceedings of the 13th International semantic web conference*, pages 357–372, 2014.

[76] Md-Mizanur Rahoman and Ryutaro Ichise. Automatic erroneous data detection over type-annotated linked data. *IEICE TRANSACTIONS on Information and Systems*, 99(4):969–978, 2016.

[77] Lukasz Golab, Flip Korn Howard Karloff, Avishek Saha, and Divesh Srivastava.

Sequential dependencies. In *Proceedings of the 35th International Conference on Very Large Data Bases Endowment 2.1*, pages 574–585, 2009.

[78] Nick Koudas, Avishek Saha, Divesh Srivastava, and Suresh Venkatasubramanian. Metric functional dependencies. In *the IEEE 25th International Conference on Data Engineering*, pages 1275–1278, 2009.

[79] Grace Fan, Wenfei Fan, and Floris Geerts. Detecting errors in numeric attributes. In *Proceedings of the 15th International Conference on Web-Age Information Management*, pages 125–137, 2014.

[80] Lihua Zhao, Natthawut Kertkeidkachorn Rumana Ferdous Munne, and Ryutaro Ichise. Missing rdf triples detection and correction in knowledge graphs. In *Proceedings of the 7th Joint International Semantic Technology Conference*, pages 164–180. Springer, 2017.

[81] Huiying Li, Feifei Xu Yuanyuan Li, and Xinyu Zhong. Probabilistic error detecting in numerical linked data. In *Proceedings of the 26th International Conference on Database and Expert Systems Applications*, pages 61–75, 2015.

[82] Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui. Towards time-aware knowledge graph completion. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1715–1724, 2016.

[83] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

[84] Yang Chen and Daisy Zhe Wang. Towards time-aware knowledge graph completion. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 649–660, 2014.

[85] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. In *Proceedings of the 32nd Association for the Advancement of Artificial Intelligence conference on artificial intelligence*. AAAI Press, 2018.

[86] Razniewski Simon, Fabian Suchanek, and Werner Nutt. But what do we actually know?. In *the 5th Workshop on Automated Knowledge Base Construction*, pages 40–44, 2016.

[87] Jiaqing Liang, Yanghua Xiao, Haixun Wang, Yi Zhang, and Wei Wang. Probase+: Inferring missing links in conceptual taxonomies. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1281–1295, 2017.

[88] Muhao Chen and Carlo Zaniolo. Learning multi-faceted knowledge graph embeddings for natural language processing. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 5169–5170. ijcai.org, 2017.

[89] Rumana Ferdous Munne and Ryutaro Ichise. Joint entity summary and attribute embeddings for entity alignment between knowledge graphs. In *Proceedings of the 15th Hybrid Artificial Intelligent Systems Conference*, pages 107–119. Springer, 2020.

[90] 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.

[91] 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 Annual Conference on Advances in neural information processing systems*, pages 3111–3119, 2013.

[92] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 19th conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. ACL, 2014.

[93] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the 30th Association for the Advancement of Artificial Intelligence conference on artificial intelligence*, page 2659–2665. AAAI Press, 2016.

[94] Weixin Zeng, Xiang Zhao, Jiuyang Tang, and Xuemin Lin. Ceaff:collective entity alignment via adaptive features. In *Proceedings of IEEE 36th International Conference on Data Engineering*, pages 1870–1873, 2020.

[95] Md Mostafizur Rahman and Atsuhiro Takasu. Exploiting knowledge graph and text for ranking entity types. *ACM SIGAPP Applied Computing Review*, 20(3):35–46, 2020.

[96] 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*, pages 260–269, 2016.

[97] Zhen Wang, Jianwen Zhang, Jianlin Feng, , and Zheng Chen. Knowledge graph

and text jointly embedding. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1591–1601, 2014.

[98] Natthawut Kertkeidkachorn and Ryutaro Ichise. Estimating distributed representations of compound words using recurrent neural networks. In *Proceedings of the 22nd International Conference on Applications of Natural Language to Information Systems*, pages 235–246. Springer, 2017.

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

[100] Eric Prud' hommeaux and Alexandre Bertails. A mapping of sparql onto conventional sql. In *World Wide Web Consortium*, pages 697–706, 2007.

[101] Radim Řehůřek and Petr Sojka. Gensim—statistical semantics in python. In *Retrieved from genism. org*, 2011.

[102] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, and Ghemawat S. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In *Computing Research Repository (CoRR) abs/1603.04467*, 2016.