Exploiting Language Models for Entity Alignment in Knowledge Graphs

Master's Thesis of

Radina Sofronova

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Dr. Ralf H. Reussner
Second reviewer:	Prof. Dr. Harald Sack
Advisor:	M.Sc. Russa Biswas
Second advisor:	Dr. Mehwish Alam

1. December 2021 – 31. August 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 31st August 2022

. . . . . .

(Radina Sofronova)

Abstract

Knowledge Graphs (KGs) are composed of structured information about a particular domain in the form of entities and relations and have been recognized as the foundation for various artificial intelligence applications. These applications often require diverse knowledge which is not present in a single KG.

Entity Alignment (EA) is the task of aligning equivalent entities which represent the same real-world object across different KGs. As a result, heterogeneous knowledge from different data sources is fused into a unified and consistent KG. Prior works in the field incorporate embedding-based methods to learn the graph structure and then perform the alignment by using a set of aligned entity pairs for training. Most of these models are supervised and require already aligned entities. Additionally, they discard a valuable source of information, e.g. textual data which is present in the KG.

On the other hand, Language Models (LMs) have become crucial for achieving state-of-the-art performance in Natural Language Processing (NLP) tasks. These pre-trained LMs generate representations of textual information into a low dimensional vector space. We draw a parallel between LMs and KGs by investigating the ability of an unsupervised multi-lingual word embedding alignment model to perform entity alignment.

This thesis introduces a novel approach for Entity Alignment named ***LM-EA*** which leverages only the side information, namely entity names and descriptions. LMs - Word2vec [25], FastText [6], Wikipedia2Vec [39] and mBERT [13], constitute the basis for the embedding learning process. We propose one supervised and one unsupervised approach for aligning two embedding spaces, inspired by the work of [11]. Our model has been successfully evaluated on two datasets, one monolingual and one multilingual.

Zusammenfassung

Wissensgraphen (Knowledge Graphs, KGs) bestehen aus strukturierten Informationen über ein bestimmtes Gebiet in Form von Entitäten und Relationen und sind als Grundlage für verschiedene Künstliche Inteligenz (KI) Anwendungen anerkannt. Diese Anwendungen erfordern oft vielfältiges Wissen, das nicht in einem einzigen KG enthalten ist.

Entity Alignment (EA) ist die Aufgabe, sich entsprechende Entitäten, die das gleiche reale Objekt in verschiedenen KGs repräsentieren, zu verknüpfen. Das Resultat der EA ist, dass heterogenes Wissen aus unterschiedlichen Datenquellen zu einem einheitlichen und konsistenten KG zusammengeführt ist. Vorläufige Arbeiten in diesem Gebiet basieren auf Embedding-Verfahren, die die Graphstruktur erfassen. Zudem, sind die meisten dieser Modelle überwacht und erfordern bereits abgeglichene Entitäten. Außerdem vernachlässigen sie wertvolle Informationsquellen, z.B. Textinformationen, die in der KG vorhanden sind.

Andererseits sind Sprachmodelle (Language Models, LMs) entscheidend für die Erzielung von State-of-the-Art Leistungen im Bereich der natürlichen Sprachverarbeitung (Natural Language Processing, NLP). Diese vortrainierten Sprachmodelle erzeugen Darstellungen von Textinformationen in einem niedrigdimensionalen Vektorraum. Wir ziehen eine Brücke zwischen LMs und KGs, indem wir die Eignung eines unüberwachten, multilingualen Word Embedding Alignment Modells zur Durchführung von EA untersuchen.

In dieser Abschlussarbeit wird ein neuer EA Ansatz vorgestellt, ***LM-EA***, der nur die Nebeninformationen, nämlich die Entitätsnamen und -beschreibungen, berücksichtigt. Die Sprachmodelle (LMs) Word2vec [25], FastText [6], Wikipedia2Vec [39] und mBERT [13] dienen als Grundlage für den Lernprozess der Einbettung. Wir schlagen einen überwachten und einen unüberwachten Ansatz für die Angleichung zweier Einbettungsräume vor, inspiriert durch die Arbeit von [11]. Unser Modell wurde erfolgreich an zwei Datensätzen evaluiert, einem einsprachigen und einem mehrsprachigen.

Contents

List of Figures

List of Tables

1 Introduction

Recently, Knowledge Graphs (KGs) have been widely applied in the field of Artificial Intelligence (AI). They are becoming essential sources of knowledge for people and AI-related applications, such as question answering, intelligent conversational agents, and recommender systems. Besides open source KGs such as DBpedia [20], Wikidata [35] and Freebase [7], many companies such as Google[1], Amazon[2] and Siemens [18] have constructed their own KGs to improve the results in search and generate business value in real-world applications.

Knowledge graphs represent information about different domains in the form of a directed labeled graph. The smallest building block of a KG is a triple consisting of a subject, a relation and an object. The subject and object are entities which refer to unique objects in the real world and are linked by a relation, e.g. *<Da_Vinci, born_in, Florence>* in which the order is strict.

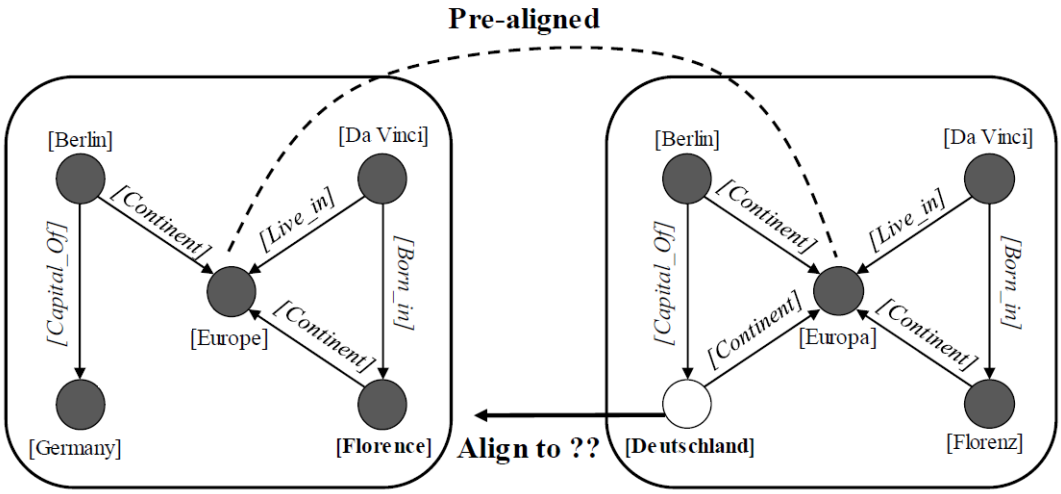
1.1 Motivation

A common characteristic across all of the KGs is that they are seldom complete and lack a large portion of knowledge. Different KGs are constructed independently from different data sources, so they contain complementary facts due to the variations in the sources of information. For instance, DBpedia [20] is constructed by automatically processing Wikipedia infoboxes, whereas Wikidata [35] is assembled based on collaborative efforts from its user base. Another challenge presents multilingual KGs, which are still far from being complete, although they contain some existing links between the same entities in different languages [3][44].

An example containing segments of two multilingual KGs (an English and a German KG) is presented in Fig. 1.1. The two KGs have considerable overlap in their entities and relations and only some of the equivalent entity links exist between them. The dark nodes represent entities that are already aligned in pairs, namely *(Europe; Europa)*, *(Berlin; Berlin)*, *(Da Vinci; Da Vinci)* and *(Florence; Florenz)*. The information that Germany and Deutschland are the same real-world entity is missing. Moreover, the two KGs have complementary information about this entity: the German KG has the information that Germany is in Europe, given by the triple *<Deutschland, continent, Europa>*, but this information is missing in the English KG. The information about this entity can be enriched if we can determine that Deutschland refers to the same real-world entity as Germany, i.e., they are aligned

[1] https://blog.google/products/search/introducing-knowledge-graph-things-not/
[2] https://aws.amazon.com/neptune/knowledge-graphs-on-aws/

entities. As a result of the EA, the information provided by the English KG will be enriched. This example proves that EA is a fundamental task for Knowledge Graph Competion (KGC).



(a) English KG Segment (b) German KG Segment

Figure 1.1: An example of two unaligned KGs (modified by [22]).

Although existing embedding-based EA methods have achieved promising results, they are still challenged by the following two limitations.

Currently, increasing attention has been paid to the utilization of KG embedding-based (KGE) models for tackling the entity alignment task. KGE models capture the structure information and solely depend on the facts in the KG. The KGs are usually sparse, with a large number of long-tail entities whose structural embeddings have low expressiveness. For example, in the Chinese version of DBpedia - $DBpedia_{ZH}$[3], the long-tail entities that appear less than 5 times occupy 74.1% [33] which affects the quality of the KGE drastically. On the other hand, KGE-based EA models are discarding a valuable source of information. In order to extend the KGE models, external information, e.g. textual information, can be utilized. [12] consider how this textual information can help overcome this limitation.

Another challenge for most of the proposed EA models is that they rely on seed alignment as labelled training data, namely pre-aligned entity pairs for supervision. However, in many practical scenarios, such seed entity alignment is not accessible or is very costly to obtain.

[3]https://wiki.dbpedia.org/downloads-2016-10

1.2 Objective

To cope with the above limitations, we propose ***LM-EA***, a new EA model which utilizes the textual information in the KGs. The underlying idea is to use the entity name or description as input to a Language Model (LM) (see Section 2.2.2) which represents the entities of each KG in a low-dimensional vector space. On top of that, an alignment model, inspired by the work of [11], learns a mapping between the two vector spaces and aligns the entities of the two KGs in the same space. The main characteristic of ***LM-EA*** is that it has two settings - supervised and unsupervised, and for the latter no pre-aligned entity pairs are needed which is a novelty among the EA approaches.

The objective of this work is to explore if an unsupervised word embedding alignment model can be modified for the task of entity alignment in KGs.

The main research questions of this thesis are as follows:

- **RQ1**: Is the textual information contained in a KG (entity name or entity description) sufficient for learning good entity representations?
- **RQ2**: Can a word embedding alignment model be modified for the task of entity alignment in KGs?
- **RQ3**: Can unsupervised training of an EA model achieve comparable results with the supervised setting?

We propose a novel EA model named ***LM-EA*** for the task of EA which utilizes only the textual information present in a KG. The main contributions of this thesis are as follows:

- A word embedding alignment model is modified to perform the EA task by aligning the embedding spaces of the source and the target KG.
- To the best of our knowledge, ***LM-EA*** is the first approach which utilizes only textual information present in the KGs and doesn't consider structural information.
- We propose a way to learn a mapping between the vector spaces of the source and the target KG in an adversarial manner which allows our model to be trained without supervision, unlike the existing baseline models which require seed entity pairs.

1.3 Outline

The structure of the thesis is as follows: In chapter 2, background information and relevant terms are introduced, which are important for a good understanding of the rest of the thesis. Chapter 3 analyzes different graph entity alignment techniques by comparing related work. Chapter 4 describes the approach developed in the course of the thesis, followed by a presentation, an evaluation and an analysis of the results in chapter 5. Finally, chapter 6 concludes the thesis with a critical reflection and a short summary and presents future directions of research.

2 Foundations

This chapter presents the basic concepts that are important for the understanding of the rest of the thesis. They are separated into two main fields - semantic web and deep learning. The concepts introduced in this chapter are from those two domains.

2.1 Knowledge Graphs

To understand the development of knowledge graphs (KGs), we first describe the emergence of the *Semantic Web* which represents a new vision about how the Web should be constructed so that its information can be processed automatically by machines on a large scale. The term "Semantic Web" was coined by the inventor of the "World Wide Web", Sir Tim Berners-Lee, who defined it as follows: "Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [2]. Specifically, it is a set of particular standards that allow the data to be easily processed by machines and shared across all the members of the network. The term Linked Data [5] was also coined to refer to the datasets interlinked and built by a set of the recommended standards.

Resource Description Framework (RDF) [1] is one of the standards proposed in the early stage and became the basis of the Semantic Web. RDF is a standard for encoding data and is used for representing information about resources and their relations existing in the real world. It decomposes information into facts and defines the facts as RDF statements which have the form of a triple <subject, predicate, object>, with the order of the elements of a triple being strict. A collection of RDF statements is called an RDF Graph and represents some knowledge as a collection of pieces of information. Since the order of the subject, predicate and object is strict, the resulting graph is a directed and labelled graph.

More precisely, in an *RDF triple*, the subject can be an IRI or a blank node, the predicate should be an IRI, and the object can be an IRI, a literal or a blank node. An IRI (Internationalised Resource Identifier) within an RDF graph is a string that unambiguously identifies a resource [14]. A literal is used for values such as strings, numbers and dates.

In the definition of the Semantic Web, it was stated that it allows machines to understand the meaning (semantics) of the information on the Web. The RDF statements describe resources and relations between them, but the meaning is still missing. A way to introduce semantics in the RDF data is defined by the *RDF Schema (RDFS)*. RDFS provides a

[1] https://www.w3.org/RDF/

simple schema language for RDF, and allows one to declare classes/properties, using the predefined language level class *rdfs:Class*/property *rdfs:Property*. Moreover, RDFS can also specify some dependencies among classes and properties, using the predefined language level properties *rdfs:subClassOf*, *rdfs:subPropertyOf*, *rdfs:domain* and *rdfs:range* [26].

The research on Semantic Web and linked data led to many *open datasets* eventually comprising the Linked Open Data cloud. These datasets, or KGs, are typically cross-domain. Many open KGs are sourced from Wikipedia since it contains a very large factual knowledge spread over multiple domains [15]. In the following, two representative KGs comprising our experimental datasets are presented.

DBpedia
DBpedia [20] is a KG that is derived by extracting structured data from Wikipedia through an open source extraction framework. The extracted information for each Wikipedia page takes the form of an RDF graph. The collection of all RDF graphs forms a large RDF dataset. This dataset can be viewed as Wikipedia's machine-readable version, with the original Wikipedia remaining the human-readable one. The construction process is aided by crowd-sourcing efforts and the ontology is collectively maintained by its user community. Up to August 2022, the full DBpedia data set features 38 million labels and abstracts in 125 different languages [2].

Freebase
Freebase [7] was a collaborative knowledge base launched in 2007. The company that runs Freebase was bought by Google in 2010, and then, the knowledge base has improved Google's KG. Freebase was shut down by Google in 2016 and its knowledge has been incrementally included in Wikidata. The latest dump contained 1.9B facts [3] [15].

2.2 Deep Learning and NLP

Deep learning methods employ multiple processing layers to learn hierarchical representations of data, and have produced state-of-the-art results in many domains. Recently, a variety of methods have emerged in the context of natural language processing (NLP). NLP enables computers to perform a wide range of natural language related tasks, such as parsing, part-of-speech tagging, named entity recognition, semantic role labelling, and synonym detection, as well as machine translation and question answering. In the last few years, neural networks based on dense vector representations have been producing superior results on various NLP tasks. This trend is stimulated by the success of word embeddings [24] and deep learning methods [29][42].

[2] http://wikidata.dbpedia.org/about
[3] https://developers.google.com/freebase/

2.2.1 Word Embeddings

As already mentioned, deep learning currently dominates the benchmarks for various NLP tasks and, at the basis of such systems, words are frequently represented as embeddings – vectors in a low dimensional space – learned from large text corpora. Various algorithms have been proposed to learn distributed word representations in the form of dense vectors.

One of the first proposed algorithms, word2vec [25] provides an efficient way to learn word embeddings from large corpora based on word context and negative sampling. Much research has been put into producing word embeddings, resulting in algorithms like GloVe [27] and fastText [6]. Lately, much effort has focused on neural language models that produce contextual word representations, like ELMo [28], BERT [13], and XLNet [41].

2.2.1.1 Word2Vec

Word2Vec was proposed by Mikolov [25]. It aims to learn the distributed representation for words reducing the high dimensional word representations in a large corpus. There are two settings of Word2Vec, namely CBOW and skip-gram (see Fig. 2.1).

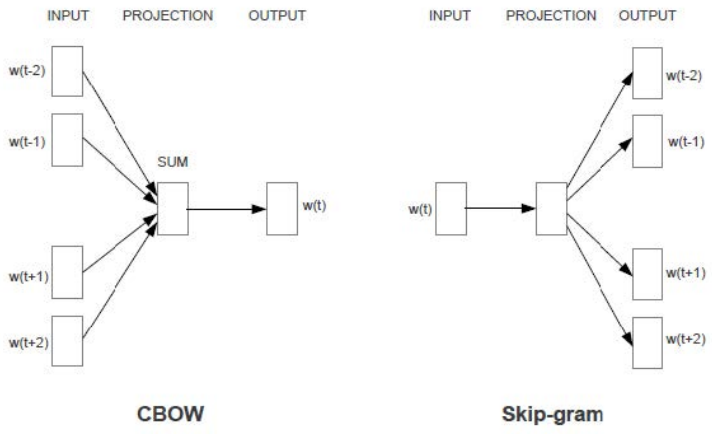


Figure 2.1: Overall architecture of Word2Vec[25].

The CBOW model predicts the current word from a window of context words and the skip-gram model predicts the context words based on the current word. Both consist of an input, an output and a single hidden layer. In general starting from a corpus, one-hot-encoded embeddings for each word w are computed. These vectors form the input to both models that are passed to the hidden units without any activation function (i.e. only computing the dot product between W and input vector(s)), which in turn are directly passed to the output layer. The probability of predicting a word is then computed by passing the hidden layers' outputs to a softmax function.

2.2.1.2 FastText

FastText[6] is an extension of the Word2Vec model, and has both the CBOW and Skip-gram architectures. The main difference with Word2Vec is that each word is represented as a bag of n-gram characters. This is espeially benefitial in capturing representations of shorter and rare words since they can be obtained by breaking down words into n-grams to get the embeddings.

2.2.1.3 Wikipedia2Vec

Wikipedia2Vec[39] is a LM which jointly learns the embeddings of words and entities from Wikipedia and places semantically similar words and entities close to each other in the vector space. Wikipedia2Vec extends the Word2Vec skip-gram model.

Wikipedia2Vec learns embeddings by jointly optimizing word-based skip-gram, anchor context, and link graph models (see Fig. 2.2). The word-based skip-gram model learns word embeddings by predicting the neighboring words of a given word in a Wikipedia page. The anchor context model places similar words and entities close to each other in the vector space using hyperlinks and their neighboring words in Wikipedia. The link graph model learns entity embeddings by predicting the neighboring entities of each entity in the Wikipedia's link graph which is an undirected graph with entities in the nodes and edges that represent the presence of hyperlinks between them. Each of the models defines a loss function. The loss functions are linearly combined and optimized using stochastic gradient descent (SGD).

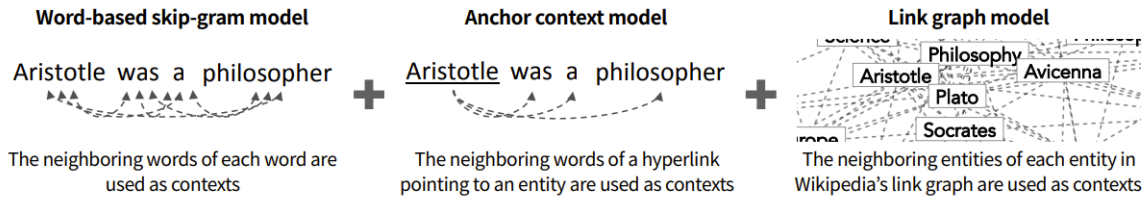


Figure 2.2: Overall architecture of Wikipedia2Vec[39].

2.2.2 Pre-Trained Language Models

The LMs presented in Sec. 2.2.1 are capable of generating latent representations of the words which capture the semantic meanings but they do not dynamically change according to the context they appear in. They are static in nature and are context independent. However, contextual LMs such as BERT, etc., encode the semantics of the words differently based on the context they appear in. All the language models are trained on huge unlabelled text corpora.

2.2.2.1 mBERT

Bidirectional Encoder Representations from Transformers (BERT) is a contextual LM designed to learn deep bidirectional representations by jointly conditioning on left and right context in all layers. Multilingual-BERT (mBERT) is trained on text from Wikipedia with a shared vocabulary across all the languages.

In order to learn the bidirectional representations BERT uses a masked language objective - some of the input tokens are randomly masked, and the objective is to predict the original vocabulary id of the masked word relying only on its context. The masked language model allows joint processing of the left and right context. BERT also has a next sentence prediction learning objective and because of that the model supports tasks dealing with sentence pairs.

The BERT model architecture is a multi-layer bidirectional transformer encoder. The input representation supports a single sentence or sentence pairs. BERT uses word-piece embeddings with a 30K token vocabulary. The first token of every sequence is a special [CLS] token, which stands for classification. Sentence pairs are concatenated in a single sentence using the [SEP] special token as separator. In addition, a learned embedding is added to each token indicating the sentence it belongs. Thus, the token representations are the addition of the token, sentence, and position embeddings.

2.3 Embedding Space Alignment Techniques

Most of the work in the area of embedding space alignment focuses on aligning multilingual word embeddings. However, the same techniques can be applied to a multitude of alignment problems, including entity alignment across different KGs.

2.3.1 Linear Alignment

In one of the original Word2Vec papers [23], Mikolov et al. propose a way for aligning embeddings learned from corpora in different languages. This approach relies on the assumption that the two embedding spaces can be aligned via a linear transformation. They align the embedding spaces by using a translation matrix W such that $z = Wx$, where z is a vector belonging to the target vector space and x is a vector belonging to the source vector space. To calculate the translation matrix, a dictionary that provides mappings for a subset of the words is needed. Then, existing linear algorithms are used to calculate the pseudo-inverse. In order to achieve optimal results, it is recommended to use a very large seed vocabulary.

2.3.2 Non-linear Alignment

Unfortunately, the linear alignment approach does not always scale, since the number of parameters is limited to the translation matrix W. It is possible to follow the same approach, but instead of deriving a pseudo-inverse matrix, a neural network is trained to learn a non-linear translation function. The non-linearities can be introduced by using activation functions such as ReLUs.

An example for a non-linear alignment method is MUSE [11]. It maps two word embedding spaces onto each other via an orthogonal projection. MUSE assumes pre-trained embedding spaces are available and learns the mapping between them.

2.4 Entity Alignment

Entity alignment aims to find entities located in different KGs, which are equivalent and represent the same real-world object. A KG $G = (E, R, T)$ is a directed graph comprising a set of entities E, relations R, and triples $T \subseteq E \times R \times E$. Given two KGs $G_1 = (E_1, R_1, T_1)$ and $G_2 = (E_2, R_2, T_2)$, and a train set of already aligned entities $S = \{(u, v) \mid u \in E_1, v \in E_2, u \leftrightarrow v\}$, where $\leftrightarrow$ represents equivalence, the task of entity alignment is to find the entity pairs $(e1; e2)$ of equivalent entities $e_1 \in E_1$ and $e_2 \in E_2$ of the test set.

By examining the frameworks of current entity alignment approaches, [44] identifies a general structure comprising the four main components, illustrated in Fig. 2.3. A KG representation learning method, combined with KG structure information or external information, is used to represent the entities of the KG as low-dimensional vectors. Then the two vector spaces are aligned by the alignment module and the similarity between vectors is calculated in the prediction module to find equivalent entity pairs.

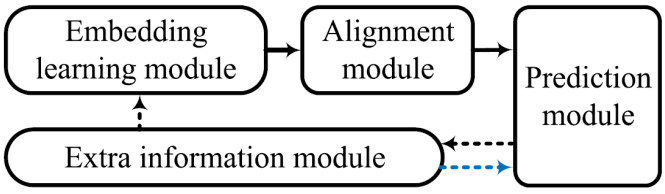


Figure 2.3: A general EA framework [44].

Embedding learning module
The embedding learning module aims at learning a low-dimensional vector representation of the entities. In recent times, KG embedding models have been successfully used to model KGs. This has led to the introduction of embedding-based models, which leverage the KG structure for the entity alignment task. Most of the alignment models are built on top of one of the two basic KG embedding models TransE [8] and Graph Convolutional

Network (GCN) [19].

Alignment module
The alignment module aims at mapping the entity embeddings of different KGs, which are learned by the previous module, into a unified space. There are two main strategies - using a margin-based loss function or using a corpus fusion. The first one uses the train set of aligned entities as positive pairs and generates negative pairs by corrupting the positive ones. Then, the margin-based loss function requires that the distance between the positive pairs is small, the distance between the negative pairs is large, and there should exist a margin between the distances of positive and negative pairs. The other strategy is to use a corpus fusion, namely seed entity pairs, to bridge the training corpora of the different KGs.

Prediction module
The prediction module takes as input the unified embedding space of the two KGs returned by the alignment module. It aims at finding the most likely target entity for each source entity of the test set. For this purpose, distance measures, such as the Euclidean or Manhattan distance, or similarity measures, such as cosine similarity, are used. For each source entity, a ranked list of target entities is returned according to the chosen metric. As a result, the target entity with the highest matching probability is aligned to the source entity.

Extra information module
Some entity alignment methods utilize extra information to improve performance. One strategy is to use bootstrapping or iterative training, for which likely entity pairs are labelled as training pairs for the next iteration. This has proven to improve the alignment results progressively. Another strategy is to use textual information, such as entity names or descriptions, as input features for learning the entity embeddings in the first module. Models which implement this strategy are thoroughly introduced in Chapter 3.

3 Related Work

Entity Alignment for KGs has been an active research topic in the data mining and semantic web communities since the beginning of the 21^{st} century. Some of the proposed methods leveraged terminological structure, exploited a set of heuristics or were based on relational clustering techniques [21].

The past decade has witnessed a large amount of research on representation learning for KGs. Motivated by the success of embedding-based methods for link prediction, researchers adapted these methods to address the EA problem [21]. Hence, the research direction of EA methods shifted and focused on EA via embedding-based methods. The motivation is that if the neighbouring structure and the values of the attributes of counterpart entities from different KGs are similar, then their low-dimensional representations should also be similar. The EA approaches consist of integrating two or more KGs into the same source of knowledge by aligning nodes that refer to the same entity.

A large variety of embedding-based approaches have been proposed for the task of EA. Their performance is systematically compared by multiple recent benchmark studies. The work by Sun et al. [32] provided the first in-depth analysis and comparison of EA methods. This paper also released OpenEA, an open-source library[1] which contains an implementation of 13 recent EA methods and some small-scale benchmark datasets. Another similar benchmark study is provided by Zhao et al. [44]. Last, but not least, Zhang et al. [43] evaluate state-of-the-art methods in an industrial context by exploring the impact of seed alignments and different biases. They also contribute by proposing a new industrial benchmark dataset that is extracted from two heterogeneous KGs under deployment for medical applications.

We divide the existing entity alignment methods into two categories according to the information they utilize - entity alignment with structure information (Section 3.1) and entity alignment with multiple sources of information (Section 3.2). Most of the EA models learn the vectors of the entities from the triple information by considering the KG structure, whereas only a handful of them consider the textual information available in the KGs. Since the LM-EA model proposed in this thesis utilized only the textual information available in a KG, we take a close look at the EA models which utilize textual KG data.

On the other hand, regarding training, ***LM-EA*** is the only unsupervised model, except for AttrE [34], which performs EA without using any seed entity pairs for the alignment. However, the AttrE approach differs from ours since they leverage structural and character

[1] https://github.com/nju-websoft/OpenEA

embeddings, whereas we use language models. Besides, their approach aligns two embedding spaces by minimizing the embedding distance between entities with similar attribute character embeddings. In contrast, we learn a mapping between the vector spaces of the source and the target KG in an adversarial manner.

We divide the existing entity alignment methods into two categories according to the information they use. Section 3.1 presents entity alignment methods with structure information, whereas section 3.2 presents entity alignment methods which use textual sources of information.

3.1 Entity alignment methods with structure information

The structure information, i.e. relation triples, is the most commonly used information in the exising EA approaches. These models utilize either a translation-based embedding method, or a graph neural networks (GNNs)-based embedding method.

EA models that use Translation-based embedding methods
Most of the translation-based EA approaches utilize TransE[8] as the underlying embedding method. It represents both entities and relations in the same vector space. TransE regards a relation r as the translation from the head entity h to the tail entity t, that is $h + r \approx t$.

MTransE
An example for such a model is MTransE [9]. It is one of the first neural approaches that successfully adapts an embedding-based method for link prediction. MTransE minimizes the loss function:

$$J = S_K + \alpha S_A, \tag{3.1}$$

where S_K is the loss function of the embedding module, S_A is the loss function of the alignment module, and α is a factor that weights S_K and S_A.

EA models that use GNN-based embedding methods
Translational methods can not cope with various complex graph structures. On the contrary, GNNs learn the embeddings by aggregating the representations of neighboring nodes. In other words, GNNs use the adjacent information to represent KGs. They rely on message passing, according to which, each graph node recursively receives and aggregates node representations from its neighbors in order to represent the local graph structure. Many recent studies use GNNs as the basis for their EA models.

GCN
GCN [19] takes as input the randomly initialized entity embeddings of the KG. Then, it learns a set of layer-specific weights (filters) that are multiplied with the input embeddings. It acts as a sliding window across the KG that learns entity features while preserving useful structural information from the neighborhoods.

3.2 Entity alignment methods with textual information

In addition to structure information, there are other kinds of information present in a KG which are helpful for EA, e.g., entity name information, and entity description information. In this section we present EA models that take advantage of one or more of the above mentioned sources of textual data information and divide them into models that leverage the entity name, and such that use the entity description.

EA models that use entity names
Some models that used entity names as external information are, e.g., HGCN[36], RDGCN[37] and GM-Align[38].

HGCN
HGCN jointly learns entity and relation predicate embeddings in three stages as follows.

The *first stage* computes entity embeddings using a GCN variant named the Highway-GCN (see Fig. 3.2), which embeds entities into a unified vector space. HGCN computes the entity embeddings for the two KGs separately and then maps them into a shared vector space.

The *second stage* gets relation embeddings based on their head and tail entity representations. This stage first computes the average embeddings of all the head and tail entities connected to the relation. Then, the two averaged embeddings are concatenated and used as the embedding of the relation.

The *third stage* uses Highway-GCN with the input being the concatenation of the entity embeddings computed in the first stage and the sum of all the relation predicate embeddings related to the entity (see Fig. 3.2). The alignment module maps the output of the Highway-GCN for the two KGs into a shared vector space using a loss which minimizes the distances between pairs of entities in the seed entity alignments, and maximizes the distances between negative samples.

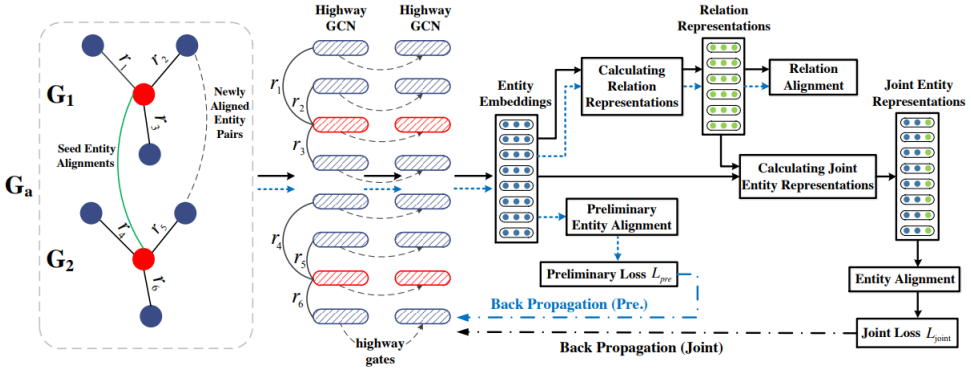


Figure 3.1: Overall architecture of HGCN [36].

RDGCN

RDGCN is a competitive neural EA model which can be used for KGs that have attribute triples. Fig. 3.2 presents the overall architecture of the Relation-aware Dual-Graph Convolutional Network (RDGCN). It differs from HGCN in that it incorporates relation information by attentive interaction. It utilizes relation information and extends GCNs (see Sec. 3.1) with highway gates (see Fig. 3.2) to capture the neighborhood structural information.

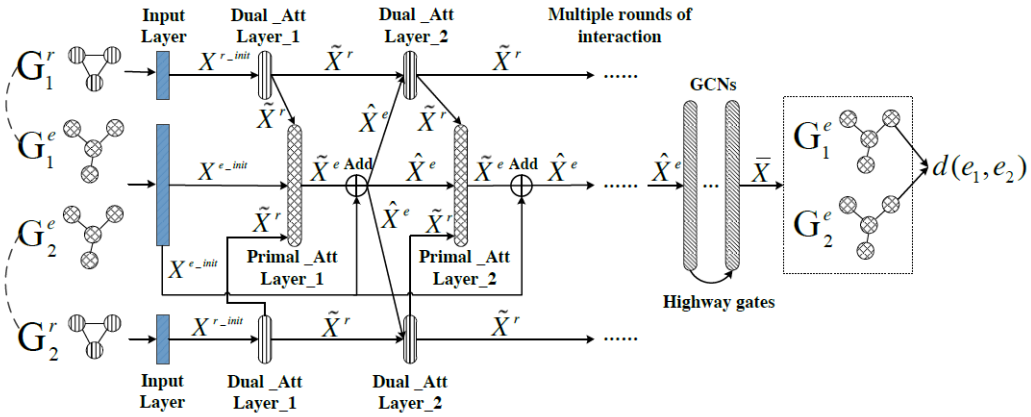


Figure 3.2: Overall architecture of RDGCN [37].

Given two KGs, G_1 and G_2, RDGCN constructs a primal entity graph G^e by merging G_1 and G_2, and its dual relation graph G^r, by creating a node in G^r for every relation type of G^e, and connecting two nodes in G^r if the corresponding relations in G^e share the same head or tail entities. Then, it uses a graph attention mechanism (a dual attention layer (see Fig. 3.2) that assigns different importance to each neighbor's contribution) to make interactions between G^e and G^r, so that the resulting entity representations in G^e capture the relation information. Then, the result is fed to a GCN which captures the structure of the neighborhood (see Fig. 3.2). In the end, RDGCN minimizes a loss function which uses a distance function and entity pairs from the seed alignment.

GM-Align

GM-Align introduces the topic entity graph, a local sub-graph of an entity, to represent entities with their contextual information. GMAN-Align uses entity names as input features. GM-Align is outperformed by HGCN and RDGCN. The architecture of the model is presented in Fig. 3.3.

GM-Align first utilizes a GCN to encode two KGs, resulting in a list of entity embeddings for each graph of them. Then, it compares each entity from the one KG against all entities from the other KG by using a matching method, which generates matching vectors for all entities in the two graphs. Then, another GCN is applied to propagate the local matching information throughout the entire graph which results is a global matching vector for each topic graph that is used for the final prediction.

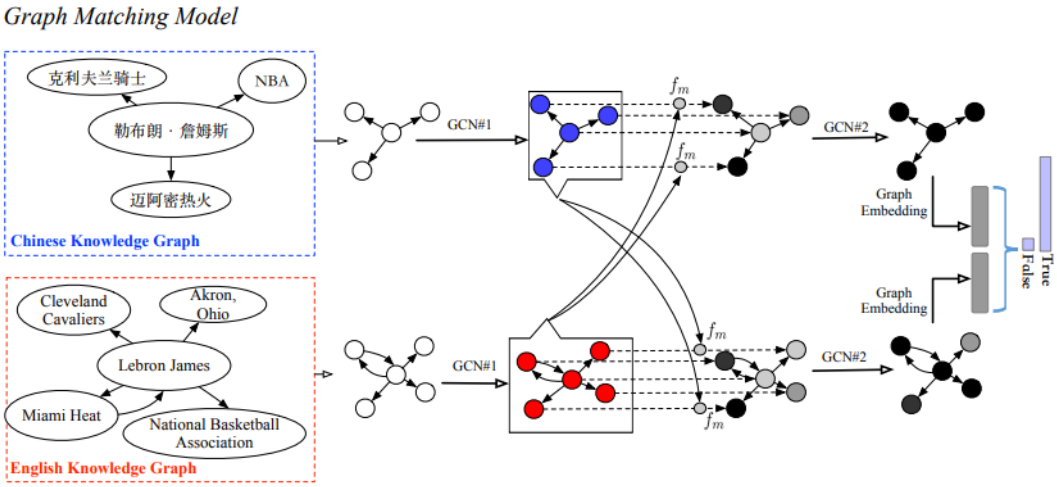


Figure 3.3: Overall architecture of GM-Align [38].

EA models that use entity descriptions

To the best of our knowledge, only the following EA models utilize entity descriptions from KGs: KDCoE [10], HMAN [40] and BERT-INT [33].

KDCoE

KDCoE builds on top of MTransE (see Sec. 3.1) by shifting the entity embeddings by the embeddings of the entity descriptions, which are treated as a type of special attribute triples where the attribute value is a literal description for the entity.

HMAN

HMAN takes into account diverse types of information such as relation predicates, attribute values, and entity descriptions besides the structural information. HMAN employs a pre-trained BERT model [Devlin et al., 2019] to capture the semantic relatedness of the descriptions of two entities.

The architecture of HMAN is given in Fig. 3.4. HMAN uses a GCN, as well as fully connected (FC) layers and highway network layers to encode the topological structure, relational features, and attribute features of the knowledge graph. For encoding the entity descriptions it incorporates the pre-trained BERT [13] model into the framework which further improves the model performance. HMAN is trained in a supervised manner, meaning that during the training phase, entities are embedded into the same low-dimensional vector space and equivalent entities are places close to each other. HMAN optimizes a margin-based loss function, which incorporates positive, as well as negative entity pairs.

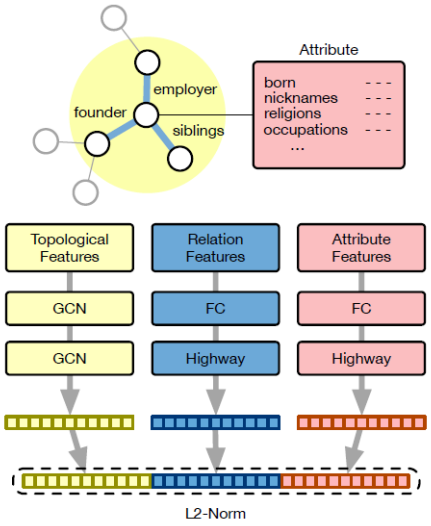


Figure 3.4: Overall architecture of HMAN [40].

BERT-INT
BERT-INT outperforms all existing neural EA models on a collection of datasets derived from DBpedia (see Sec. 5.3) and utilizes the highly successful LM BERT [13]. Different from HMAN that directly uses the embeddings of the descriptions to align entities, BERT-INT uses these embeddings as basic units to compose an interaction model.

Fig. 3.5 presents the BERT-INT architecture. The BERT model is used in the basic BERT unit to embed the name, description, attribute and value of an entity, and an interaction model is built upon the BERT embeddings to compute the interactions between these embeddings. The interactions are further divided into the neighbor-view interactions, the attribute-view interactions and the already mentioned name/description-view interaction (see Fig. 3.5).

The entity alignment is performed as follows. The basic BERT unit (see Fig. 3.5) obtains an embedding for each entity and computes the cosine similarity between the embedding of a source entity and the embedding of each entity in the target KG, and returns the top-K similar entities as candidates of the source entity. Then for the source entity and each candidate from the target KG, the BERT-based interaction model infers a matching score

between them and ranks all the candidates for evaluation. The candidate selection process can considerably enhance the alignment efficiency by the interaction model.

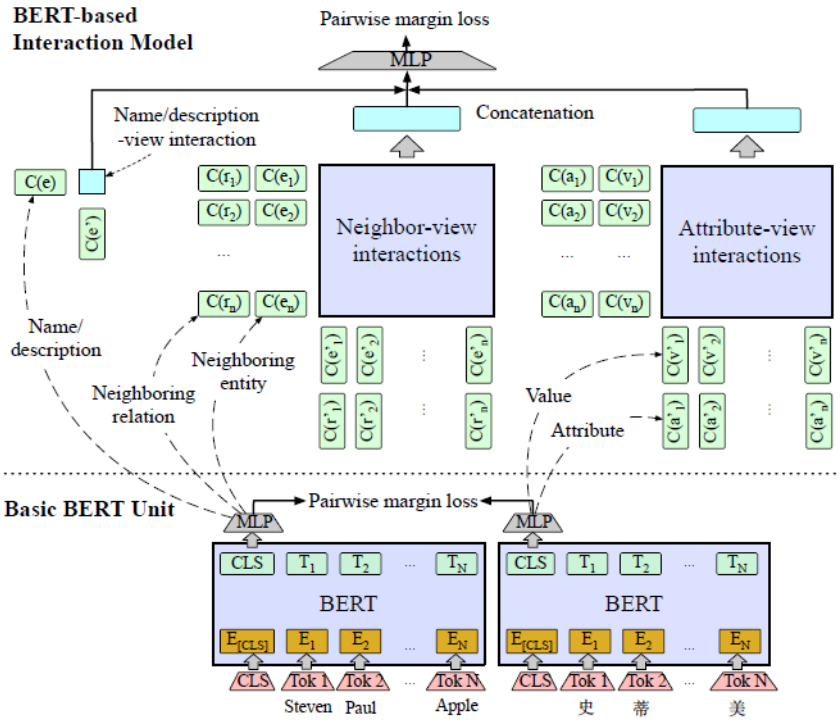


Figure 3.5: Overall architecture of BERT-INT [33].

4 Approach

This chapter introduces the proposed **LM-EA**, a **L**anguage **M**odel-based model for **E**ntity **A**lignment which exploits the textual information contained in a KG, namely the entity names and descriptions. ***LM-EA*** leverages a LM that embeds the name/description of an entity, on top of which an alignment module is applied, followed by a prediction module which outputs a list of aligned entity pairs. Figure 4 shows the whole framework.

Our model follows the general EA framework presented in Sec. 2.4 comprising an embedding learning module (component Ⓐ), an alignment module (component Ⓑ) and a prediction module (component Ⓒ). In the following, each module of our framework is presented in detail.

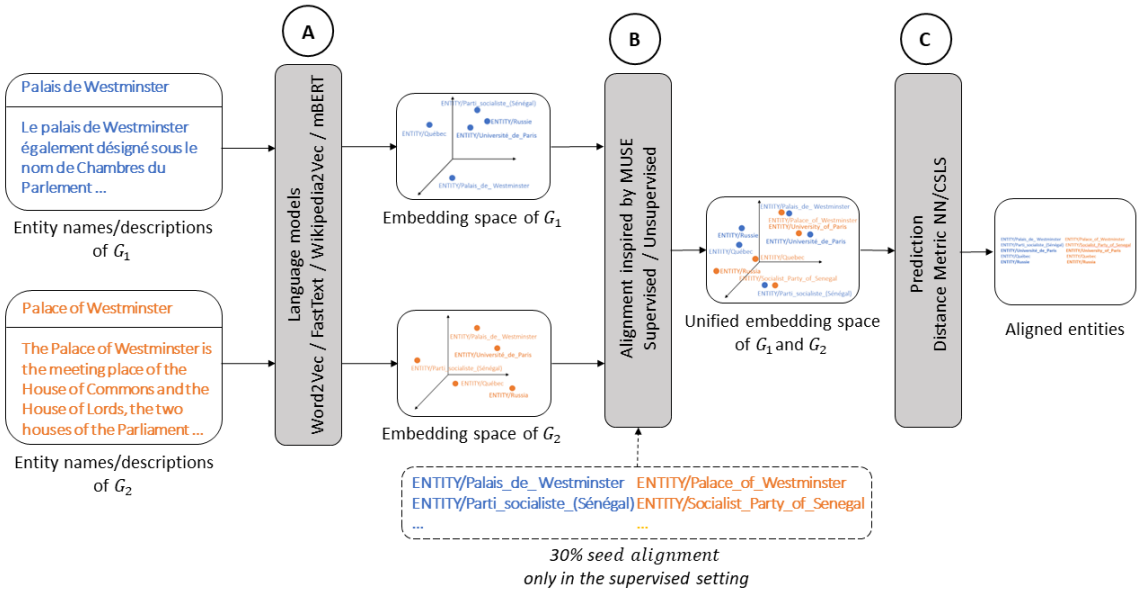


Figure 4.1: The framework of ***LM-EA***.

4.1 Embedding learning module utilizing LMs

The embedding learning module (see component Ⓐ in Fig. 4.1) aims at learning a low-dimensional vector representation of the entities. As it is the first module in the framework,

it takes as input the textual information for each entity for both the source and the target KGs. **LM-EA** has two settings - the one utilizes the entity name ***LM-EA***^{name}, whereas the other uses the entity description ***LM-EA***^{description}. In the second case, we use the name when the description is missing.

Our model focuses on uncovering the rich semantic information encoded in the entity name/description using different LMs (see Sec. 2.2.2). LMs are neural network based models that learn the distributed representation of words into a continuous low-dimensional vector space. ***LM-EA*** deploys four different LMs - Word2Vec [25], FastText [6], Wikipedia2Vec [39] and mBERT [13] which results in four settings of our model - ***LM-EA***$_{Word2Vec}$, ***LM-EA***$_{FastText}$, ***LM-EA***$_{Wikipedia2Vec}$ and ***LM-EA***$_{mBERT}$. They are presented in detail in this section.

LM-EA$_{Word2Vec}$ & ***LM-EA***$_{FastText}$
Word2Vec and FastText aim at learning the distributed representation of words in a large corpus to a low dimensional vector space. In ***LM-EA***$_{Word2Vec/FastText}$, the name/description representations of the entities are generated from the Word2Vec[1] model pre-trained on Google News dataset or by fastText[2] trained on Wikipedia corpora.

Given the name of the i^{th} entity $Name_{e_i}$ or a description of the i^{th} entity $Description_{e_i}$, $(w_1, w_2, ..., w_n)$ represents the sequence of n words in the entity name/description respectively. Their representation is given by

$$Name_{e_i}^{Word2Vec/FastText} = \frac{1}{n}\sum_{j=1}^{n} W_j \tag{4.1}$$

or

$$Description_{e_i}^{Word2Vec/FastText} = \frac{1}{n}\sum_{j=1}^{n} W_j, \tag{4.2}$$

where W_j is the the word embedding of the j^{th} word in $Name_{e_i}$ or $Description_{e_i}$ extracted from the pre-trained Word2Vec/FastText model. For the embedding of each entity x_{e_i} we use $Name_{e_i}$ or $Description_{e_i}$, depending on the chosen setting of ***LM-EA***.

LM-EA$_{Wikipedia2Vec}$
Wikipedia2Vec is a skip-gram based LM in which the entities and words from Wikipedia are jointly learned and optimized. In ***LM-EA***, the name/description representations of the entities are generated from the Wikipedia2Vec[3] model trained by simultaneously iterating over pages in Wikipedia and entities in the link graph in random order.

Given an entity e_i, if a pre-trained Wikipedia2Vec entity vector x_{e_i} for this entity exists, it is used as the embedding of e_i. Otherwise, the pre-trained word vectors of the words in

[1] https://code.google.com/archive/p/word2vec/
[2] https://fasttext.cc/docs/en/crawl-vectors.html
[3] https://wikipedia2vec.github.io/wikipedia2vec/

each entity name/description are used in the following manner. Given the name of the i^{th} entity $Name_{e_i}$ or a description of the i^{th} entity $Description_{e_i}$, $(w_1, w_2, ..., w_n)$ represents the sequence of n words in the entity name or description respectively. Their representation is given by

$$Name_{e_i}^{Wikipedia2Vec} = \frac{1}{n}\sum_{j=1}^{n} W_j \tag{4.3}$$

or

$$Description_{e_i}^{Wikipedia2Vec} = \frac{1}{n}\sum_{j=1}^{n} W_j, \tag{4.4}$$

where W_j is the the word embedding of the j^{th} word in $Name_{e_i}$ or $Description_{e_i}$ extracted from the pre-trained Wikipedia2Vec model. For the embedding of each entity x_{e_i} (for which no entity vector was present) we use $Name_{e_i}$ or $Description_{e_i}$, depending on the chosen setting of ***LM-EA*** - ***LM-EA***$^{\boldsymbol{name}}$ or ***LM-EA***$^{\boldsymbol{description}}$.

LM-EA$_{\boldsymbol{mBERT}}$
mBERT is a multi-layer bidirectional Transformer encoder for word representations which takes a sequence of words as an input and generates their vector representations. The pre-trained mBERT[4] model is trained on text from Wikipedia content with a shared vocabulary across all the 104 supported languages. In ***LM-EA***, the name/description are used as an input which allows the mBERT model to capture the semantics based on the sequence of words contained in them.

Given the name of the i^{th} entity $Name_{e_i}$ or a description of the i^{th} entity $Description_{e_i}$, $(w_1, w_2, ..., w_n)$ represents the sequence of n words in the entity name or description respectively. The mBERT encoder adds special tokens *[CLS]* and *[SEP]* at the beginning and at the end of $Name_{e_i}$ or $Description_{e_i}$. Therefore, the input sequence to the mBERT model is given by $([CLS], w_1, w_2, ..., w_n, [SEP])$. The output of the model is a sequence of contextualized embeddings of the tokens in the input sequence together with the added special tokens and is given by, $g(\hat{C_{e_i}}) = (h_{[CLS]}, h_1, h_2, ..., h_n, h_{[SEP]})$, where h_i is the hidden representation of the i^{th} token of the input sequence. ***LM-EA*** exploits the feature-based approach of the mBERT model in which fixed features are extracted from the mBERT model, similar to [4]. The mean-pooling is considered as the representation of the input sequence. It takes the average of k hidden layers. Therefore, the final entity representation of e_i generated using $Name_{e_i}$ or $Description_{e_i}$ is given by

$$Name_{e_i}^{mBERT} = \frac{1}{k}\sum_{j=1}^{k}(h_{[CLS]}^{j}, h_1^j, ...h_n^j, h_{[SEP]}^{j}), \tag{4.5}$$

or

$$Description_{e_i}^{mBERT} = \frac{1}{k}\sum_{j=1}^{k}(h_{[CLS]}^{j}, h_1^j, ...h_n^j, h_{[SEP]}^{j}), \tag{4.6}$$

[4]https://github.com/google-research/bert/blob/master/multilingual.md

where $(h^j_{[CLS]}, h^j_1, ...h^j_n, h^j_{[SEP]})$ is the representation of the j^{th} hidden state. Our model uses k=4 - the mean of the last 4 hidden layers. For the embedding x_{e_i} of each entity we use $Name_{e_i}$ or $Description_{e_i}$, depending on the chosen setting of ***LM-EA*** - ***LM-EA***$^{\boldsymbol{name}}$ or ***LM-EA***$^{\boldsymbol{description}}$.

4.2 Alignment module

The embedding learning module (see component Ⓐ in Fig. 4.1) produced two separate embedding spaces for the source and the target KG. The alignment module (see component Ⓑ in Fig. 4.1) takes as input those two vector spaces and applies the method proposed in Conneau et al.[11] to map them into a shared vector space. MUSE[5] is a word translation method, which we apply to the entity embeddings to perform entity alignment.

Let $X = \{x_{e_1}, ..., x_{e_n}\}$ and $Y = \{y_{e_1}, ..., y_{e_m}\}$ be two sets consisting of n and m entity embeddings of a source and a target KG, respectively. X and Y are trained independently. The aim of the alignment module is to learn a mapping such that for every x_{e_1}, $f(x_{e_1})$ corresponds to the equivalent entity in Y.

Depending on the availability of a labelled data set, i.e. aligned entity pairs, the ***LM-EA*** approach can be categorized as supervised or unsupervised. The pipeline consists of the following steps. For the first one, there are two variations depending on which setting is used - supervised or unsupervised.

1. Learn a transformation matrix W.
   1.1. Orthogonal transformation in the supervised setting.
   1.2. Adversarial training in the unsupervised setting.
2. Iterative refinement of the initial mapping through the Procrustes solution.

4.2.1 Learn a transformation matrix W

4.2.1.1 Learn a transformation matrix W in the supervised setting

In the orthogonal transformation, we learn the orthogonal matrix W between the source KG embedding space and the target KG embedding space, by leveraging the pre-aligned entity pairs. Suppose that there is a seed dictionary, that is, $D = (x_{e_i}, y_{e_i})$, $i = 1, ...t$, where x_{e_i} is the embedding of an entity in the source KG and y_{e_i} is the corresponding embedding of an entity in the target KG and there are t entity pairs in the seed alignment. Through iterative training, as shown in equation 4.7, we obtain the orthogonal transformation matrix W:

$$W^* = \underset{W \in M_d(\mathbb{R})}{\arg\min} \|WX - Y\|, \; WW^T = I. \tag{4.7}$$

[5] https://github.com/facebookresearch/MUSE

Once the mapping W is learned, WX and Y are in the same shared vector space.

4.2.1.2 Learn a transformation matrix W in the unsupervised setting

Different from the orthogonal transformation used in the supervised setting, in the unsupervised setting the mapping W is learned in an adversarial manner. This approach is inspired by Generative Adversarial Networks (GANs)[17]. GANs use two models: a generator that captures the data distribution, and a discriminator that estimates the probability that a sample came from the training data rather than the generator.

The adversarial training aims to learn the representation projection in an unsupervised way. Suppose $X = \{x_{e_1}, ..., x_{e_n}\}$ and $Y = \{y_{e_1}, ..., y_{e_m}\}$ represent the entity vector sets of the source KG and the target KG, respectively. A model is trained to discriminate between elements randomly sampled from $WX = \{Wx_{e_1}, ..., Wx_{e_n}\}$ and Y. This model is called the discriminator. W is trained to prevent the discriminator from making accurate predictions. As a result, this is a two-player game, where the discriminator aims at maximizing its ability to identify the origin of an embedding, and W aims at preventing the discriminator from doing so by making WX and Y as similar as possible.

Suppose the discriminator parameter is defined as θ_D, and $P_{\theta_D}(source = 1|z)$ indicates the probability that the discriminator vector z belongs to the source embedding space. The loss function of the discriminator is shown in eq. 4.8.

$$\mathcal{L}_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}(source = 1|Wx_i) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}(source = 0|y_i) \tag{4.8}$$

and the loss function of the transformation matrix W is shown in eq. 4.9.

$$\mathcal{L}_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}(source = 0|Wx_i) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}(source = 1|y_i) \tag{4.9}$$

According to the training process for GANs, for each input sample, the discriminator and transformation matrix W are updated by the stochastic gradient descendent (SGD) to minimize the two loss functions.

4.2.2 Iterative refinement using the Procrustes analysis

Procrustes [1] is a supervised method for word translation, which learns the translation in a bootstrapping way, i.e iteratively. In **LM-EA**, the learned transformation matrix W is used to build a small set of entity pairs. A new translation matrix W that maps the vector spaces X and Y of only these selected entities is induced by solving the Orthogonal Procrustes problem. By doing this, the mapping is refined. The Procrustes problem offers

a closed form solution obtained from the singular value decomposition (SVD) of YX^T (see eq. 4.10)

$$W^* = \underset{W \in M_d(\mathbb{R})}{\arg\min} \|WX - Y\| = UV^T, \; s.t. \; U\Sigma V^T = SVD(YX^T) \tag{4.10}$$

This step is used iteratively by using the new matrix W to create new seed entity pairs.

4.3 Prediction Module

The alignment module (see component Ⓑ in Fig. 4.1) resulted in a transformation matrix W which aligns the embedding spaces of the two KGs. The prediction module (see component Ⓒ in Fig. 4.1) aims at finding the equivalent target entity for each source entity and builds the aligned entity pairs. To do this, the learned transformation matrix W is used to create the shared vector space and distance metrics are used to find the nearest neighbours of a source entity.

For computing the nearest neighbours, two distance metrics are used. The first distance metric used for determining the k-nearest neighbours is cosine similarity, shown in eq. 4.11.

$$\cos(x_{e_i}, y_{e_i}) = \frac{x_{e_i} y_{e_i}}{\|x_{e_i}\| \|y_{e_i}\|} \tag{4.11}$$

In high-dimensional spaces there is a phenomenon called **hubness**. A hub is a high-density area of points in which some vectors are with high probability nearest neighbours of many other points, while others (anti-hubs) are not nearest neighbours of any point. In other words, the hubness problem indicates that some points (known as hubs) frequently appear as the top-1 nearest neighbours of many other points in the vector space [16].

Conneau et. al [11] propose a metric named Cross-domain similarity local scaling (**CSLS**) to reduce the hubness problem in high-dimensional spaces. It is used to expand high-density areas and condense low-density ones, for more accurate nearest neighbour calculation. For two entity vectors mapped to the same space – namely, x_{e_i} and y_{e_i} – the CSLS score between them is calculated as shown in eq. 4.12

$$CSLS(x_{e_i}, y_{e_i}) = 2\cos(x_{e_i}, y_{e_i}) - sim_k(x_{e_i}) - sim_k(y_{e_i}), \tag{4.12}$$

where $sim_k(x_{e_i})$ and $sim_k(y_{e_i})$ represent the average cosine similarity of x_{e_i} and y_{e_i} with their k nearest neighbours, respectively, and $\cos(x_{e_i}, y_{e_i})$ represents the cosine similarity of two entity embedding vectors.

5 Evaluation

This section gives details about the benchmark datasets, experimental setup, and analysis of the results obtained.

5.1 Datasets

The datasets used by EA methods generally derive from large-scale open-source data sources such as DBpedia [20], Wikidata [35] and Freebase [7]. For the experimental evaluation, we adopt two frequently utilized and representative benchmark datasets, including four KG pairs in total. We choose one monolingual dataset with 3 KG pairs and one multilingual dataset with 1 KG pair for the evaluation of our model since it exploits textual data and both settings are worth considering.

5.1.1 Multilingual dataset DBP15K

The DBP15K dataset is the most popular dataset for the evaluation of EA approaches. An interesting observation is that all papers that claim to evaluate their model on *DBP15K* are not based on the full DBP15K dataset (which we refer to as *DBP15k (full)*), but use a smaller subset provided by the authors of JAPE [31] (which we refer to as *DBP15K*). The smaller dataset was created by selecting a subset of entities that are popular, i.e., appear in many triples as head or tail. Since only the statistics of the *DBP15k (full)* dataset are given in the JAPE paper and the downsizing of the dataset is not mentioned at all, subsequent papers also report the statistics of the larger dataset, although they conduct their experiments with the smaller version *DBP15K*. The statistics of *DBP15K (full)* are given in Table 5.1.

The dataset that we use, as well as all the other benchmark models [44], is the *DBP15K*
The dataset that we use, as well as all the other benchmark models [44], is the *DBP15K*[1] dataset. It is proposed by JAPE [31] and consists of three multilingual KG pairs extracted from DBpedia: French to English dataset. It is proposed by JAPE [31] and consists of three multilingual KG pairs extracted from DBpedia: French to English ($DBP15K_{FR-EN}$), Japanese to English ($DBP15K_{JA-EN}$) and Chinese to English ($DBP15K_{ZH-EN}$). Each KG pair contains 15 thousand inter-language links (ILLs) as gold standards which is almost three-quarters of the entities in the respective graphs. Table 5.2 shows the statistics of the *DBP15K* dataset.

[1] https://github.com/nju-websoft/JAPE/tree/master/data

Table 5.1: Details of the *DBP15K (full)* dataset

Dataset		Entities	Relations	Rel. Triples
$DBP15K_{FR-EN}$	French	66,858	1,379	192,191
	English	105,889	2,209	278,590
$DBP15K_{JA-EN}$	Japanese	65,744	2,043	164,373
	English	95,680	2,096	233,319
$DBP15K_{ZH-EN}$	Chinese	66,469	2,830	153,929
	English	98,125	2,317	237,674

Table 5.2: Details of the *DBP15K* dataset

Dataset		Entities	Relations	Rel. Triples
$DBP15K_{FR-EN}$	French	19,661	903	105,998
	English	19,993	1,208	115,722
$DBP15K_{JA-EN}$	Japanese	19,814	1,299	77,214
	English	19,780	1,153	93,484
$DBP15K_{ZH-EN}$	Chinese	19,388	1,701	70,414
	English	19,572	1,323	95,142

5.1.2 Monolingual dataset DBP-FB

Special precaution must be taken when considering which monolingual dataset to be used for evaluating an EA model. E.g., DWY100K [2] comprises two monolingual KG pairs, $DWY100K_{DBP-WD}$ and $DWY100K_{DBP-YG}$, which are extracted from DBpedia[20], Wikidata[35] and YAGO[30], in which equivalent entities across different KGs possess identical names from the entity identifiers, and just by comparing the names ground-truth results can be achieved. Therefore, such datasets don't represent the real-life challenge of ambiguous entity names.

To bridge this gap, [44] propose a new monolingual dataset *DBP-FB* which uses DBpedia[20] as the source KG and Freebase[7] as the target KG. Freebase represents entities with incomprehensible identifiers (Freebase MIDs), and therefore different entities might share the same name. Besides, *DBP-FB* is challenging because each entity in the source KG has more than one corresponding equivalent entity in the target KG. This is a real-life scenario since KGs contain entities that other KGs do not contain. Details on the dataset construction can be found in [44]. Table 5.3 contains details about the statistics of the dataset. *DB-FB* contains 25,542 ILLs (aligned entity pairs).

[2] https://github.com/nju-websoft/BootEA/blob/master/dataset/DWY100K_raw_data.zip

Table 5.3: Details of the *DBP-FB* dataset

Dataset		Entities	Relations	Rel. Triples
DBP-FB	DBpedia	29,861	407	96,414
	Freebase	25,542	882	111,974

5.2 Experimental Setup

Textual Data from the KGs
LM-EA uses textual data in the form of entity names/descriptions. Nevertheless, the presented datasets (see Sec. 5.1) don't contain entity descriptions. Since *DBP15K* contains entities from DBpedia, we used the DBpedia dump files[3] to get the entity descriptions, in particular the short abstracts for the four DBpedia versions ($DBpedia_{EN}$, $DBpedia_{FR}$, $DBpedia_{JA}$, $DBpedia_{ZH}$). If no description is available for some entity, the entity name is used in that case.

In the case of *DBP-FB*, we again used the short abstracts from the DBpedia dump file to obtain entity descriptions for the source KG. In the target KG, Freebase, only 406 out of 25,542 source entities have English descriptions. Therefore, we skip the $\textbf{\textit{LM-EA}}^{\textbf{\textit{description}}}$ setting when evaluating our model using the *DBP-FB* dataset.

Experimental Setup of the LMs
Following the explanation of the embedding learning method (see component Ⓐ in Fig. 4.1) which uses the language models (LMs) described in Section 2.2.2, we present the experimental setup of the pre-trained LMs.

- Word2Vec was pre-trained on roughly 100 billion words from a Google News dataset, using the CBOW setting with window size 5, negative sampling 3, and embedding dimension 300. A pre-trained model for English is available.
- FastText models were pre-trained for different languages on CommonCrawl[4] and Wikipedia, using CBOW with position-weights, character n-grams of length 5, a window of size 5, 10 negatives, and embedding dimension 300. A pre-trained model for English, French, Japanese and Chinese is available.
- Wikipedia2vec was pre-trained on Wikipedia 2018 version with window size 10, epochs 10, negative sampling 15, and embedding dimension 300. A pre-trained model for English, French, Japanese and Chinese is available.
- mBERT is a single language model supporting 104 languages, trained on monolingual text from Wikipedia content with a shared vocabulary across all the languages. mBERT is a 12 layer transformer with 768 hidden layers, 12 attention heads, and 110 million parameters. Since the embedding learning module of ***LM-EA*** outputs the average of the last 4 hidden layers in mBERT, the dimension of the output vector is

[3]http://downloads.dbpedia.org/wiki-archive/downloads-2016-10.html
[4]https://commoncrawl.org/

768. mBERT supports all the languages that are relevant for the evaluation of our model with the chosen datasets.

Train/test split for the supervised *LM-EA*

Following the setting of previous methods, we take 30% of ILLs (aligned entity pairs) for training and keep the other 70% for testing. This means that for *DBP15K* we use 4,500 entities to train ***LM-EA*** in the supervised setting and 10,500 entities to evaluate it. Moreover, the precise 30/70% split is included in the dataset, to ensure a consistent evaluation of the different models on that dataset. DBP-FB on the contrary doesn't provide a train/test split, so we shuffle the entities randomly and take 30% of the ILLs (7,662) for training and the rest (17,880) for testing.

Parameter configurations of the alignment module of *LM-EA*

Table 5.4 presents the parameter configurations of the alignment module (see component Ⓑ in Fig. 4.1) of ***LM-EA***. For both settings, 5 refinement Procrustes iterations are performed and the maximum vocabulary size is disabled. In the unsupervised setting, a batch size of 32 is used and the discriminator is trained for 5 epochs with 1M iterations per epoch. For every input sample, the discriminator and the mapping matrix W are trained successively with stochastic gradient descent (SGD) with a learning rate update 0.1 to minimize the two loss functions.

Table 5.4: Parameter configurations of Alignment module

Setting	Parameter	Value
Supervised	n_refinement	5
	max_vocab	-1
Unupervised	batch_size	32
	epochs	5
	iterations per epoch	1,000,000
	map_optimizer	SGD
	dis_optimizer	SGD
	n_refinement	5
	max_vocab	-1

Evaluation Metrics

Following the evaluation of existing EA approaches, we utilize *Hits@k*, k=1,10 as the evaluation metrics. *Hits@k* indicates the percentage of correctly mapped entities in the top-k closest target entities. The closest entities are obtained by the prediction module of ***LM-EA*** (see component Ⓒ in Fig. 4.1). *Hits@1* represents the accuracy of the alignment results, which is the most important performance indicator. The results of *Hits@k* are given in percentages. Word2Vec is pre-trained only for English and the DB_FB dataset contains no entity descriptions. In those cases, **X** is set in the tables.

5.3 Results

In this section we consider the performance of ***LM-EA***. For the results of the baseline models, we rely upon the experiment results reported in [44]. They directly use the provided source codes, and the results are obtained by executing the models with the set of parameters reported in the original papers. Moreover, this is also the paper which introduces the *DB-FB* dataset. BERT-INT[33] was proposed after the study[44] was published, so the results for BERT-INT are taken from the original paper.

To the best of our knowledge, ***LM-EA*** is the first EA model which utilizes solely the textual information in the KGs. Moreover, the unsupervised setting of ***LM-EA*** is the only model, except for AttrE [34], which performs EA without using any seed entity pairs for the alignment. Therefore, no direct comparison is possible. Nevertheless, we present the results of two baseline models that use solely structural information (MTransE [9] and GCN[19]) for comparison with our method that only leverages textual information. To the best of our knowledge HMAN [40], BERT-INT [33] and KDCoE [10] are the only models, which use entity descriptions as external information, and RDGCN [37], HGCN [36] and GM-Align [38] - such that encode entity names into vector representations. No comparison is possible with KDCoE since it's not evaluated on either of the two datasets.

Table 5.5 presents the comparison of the Hits@1 and Hits@10 results of the supervised setting of ***LM-EA*** with *CSLS* as a distance measure - LM-EA$^{\text{name}}_{\text{Word2Vec}}$, LM-EA$^{\text{name}}_{\text{FastText}}$, LM-EA$^{\text{name}}_{\text{Wikipedia2Vec}}$, LM-EA$^{\text{name}}_{\text{mBERT}}$ using names as textual information. This is the best performing setting of ***LM-EA*** and that's why it was chosen for the comparison with the baselines. Since the best results when using entity descriptions were obtained in combination with *Wikipedia2Vec*, we include only this setting in the comparison table LM-EA$^{\text{description}}_{\text{Wikipedia2Vec}}$.

Table 5.6 and table 5.7 present all results obtained using the CSLS and the cosine similarity distance metric respectively. Those scores prove that CSLS is the better suited distance metric for determining the nearest neighbours. As can be observed, it produces better results in all the settings.

We ran the unsupervised setting of ***LM-EA*** with the best performing combination of textual information and language model. Table 5.8 presents the comparison of the supervised and the unsupervised setting. The two variants of ***LM-EA*** provide comparable results with only small differences for the $DBP15K_{FR-EN}$, $DBP15K_{ZH-EN}$ and $DBP-FB$ datasets. Those results prove that learning the transformation matrix W in an adversarial manner without seed alignment is highly efficient.

Table 5.5: Comparison of the supervised and the unsupervised setting of *LM-EA*.

Model	LM-EA$^{\text{name}}_{\text{Wikipedia2Vec}}$		LM-EA$^{\text{name}}_{\text{Wikipedia2Vec}}$		LM-EA$^{\text{name}}_{\text{mBERT}}$		LM-EA$^{\text{name}}_{\text{FastText}}$	
Dataset	$DBP15K_{ZH-EN}$		$DBP15K_{JA-EN}$		$DBP15K_{FR-EN}$		$DBP15K_{DB-FB}$	
Evaluation Metric	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10
LM-EA supervised	28.209	57.447	36.257	66.266	76.190	85.133	62.846	84.479
LM-EA unsupervised	24.609	57.104	18.809	41.657	73.552	82.047	62.651	83.635

Table 5.6: Results of *LM-EA* obtained using the CSLS distance metric.

Dataset	$DBP15K_{ZH-EN}$		$DBP15K_{JA-EN}$		$DBP15K_{FR-EN}$		$DBP15K_{DB-FB}$	
Evaluation Metric	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10
$\text{LM-EA}^{\text{name}}_{\text{FastText}}$	2.961	9.457	4.085	13.085	37.066	54.638	62.846	84.479
$\text{LM-EA}^{\text{name}}_{\text{Wikipedia2Vec}}$	28.209	57.447	36.257	66.266	53.533	79.219	21.118	40.123
$\text{LM-EA}^{\text{name}}_{\text{mBERT}}$	16.285	33.142	14.361	32.504	76.190	85.133	57.645	72.276
$\text{LM-EA}^{\text{name}}_{\text{Word2Vec}}$	X	X	X	X	X	X	11.0346	26.375
$\text{LM-EA}^{\text{description}}_{\text{FastText}}$	0.685	2.971	0.742	3.733	11.034	26.375	X	X
$\text{LM-EA}^{\text{description}}_{\text{Wikipedia2Vec}}$	28	56.790	35.895	66.076	53.028	77.819	X	X
$\text{LM-EA}^{\text{description}}_{\text{mBERT}}$	-	-	-	-	26.885	51.523	X	X
$\text{LM-EA}^{\text{description}}_{\text{Word2Vec}}$	X	X	X	X	X	X	X	X

Table 5.7: Results of *LM-EA* obtained using the cosine similarity distance metric.

Dataset	$DBP15K_{ZH-EN}$		$DBP15K_{JA-EN}$		$DBP15K_{FR-EN}$		$DBP15K_{DB-FB}$	
Evaluation Metric	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10
$\text{LM-EA}^{\text{name}}_{\text{FastText}}$	1.971	7.333	2.266	9	25.8	44.161	61.029	81.621
$\text{LM-EA}^{\text{name}}_{\text{Wikipedia2Vec}}$	22.914	46.609	27.942	54.923	42.438	68.657	20.973	40.016
$\text{LM-EA}^{\text{name}}_{\text{mBERT}}$	11.923	27.066	9.790	24.228	73.323	82.704	56.068	68.540
$\text{LM-EA}^{\text{name}}_{\text{Word2Vec}}$	X	X	X	X	X	X	10.950	26.319
$\text{LM-EA}^{\text{description}}_{\text{FastText}}$	0.580	2.990	0.685	3.876	10.950	26.319	X	X
$\text{LM-EA}^{\text{description}}_{\text{Wikipedia2Vec}}$	22.657	46.390	28.019	54.561	42.285	67.419	X	X
$\text{LM-EA}^{\text{description}}_{\text{mBERT}}$	-	-	-	-	20.4	44.361	X	X
$\text{LM-EA}^{\text{description}}_{\text{Word2Vec}}$	X	X	X	X	X	X	X	X

Table 5.8: Comparison of the *LM-EA* with benchmark datasets.

Dataset	$DBP15K_{ZH-EN}$		$DBP15K_{JA-EN}$		$DBP15K_{FR-EN}$		$DBP15K_{DB-FB}$	
Evaluation Metric	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10	Hits@1	Hits@ 10
MTransE	20.9	51.2	25	57.2	24.7	57.7	8.5	23
GCN	39.8	68.9	40	72.9	38.9	74.9	17.8	42.3
HMAN	56.1	85.9	55.7	86	55	87.6	25.9	54.2
BERT-INT	96.8	99	96.4	99.1	99.2	99.8	-	-
RDGCN	69.7	84.2	76.3	89.7	87.3	95	67.5	84.1
HGCN	70.8	84	75.8	88.9	88.8	95.9	77.9	92.3
GM-Align	59.5	77.9	63.5	83	79.2	93.6	72.1	85.5
$\text{LM-EA}^{\text{name}}_{\text{FastText}}$	2.96	9.45	4.08	13.08	37.06	54.63	62.84	84.47
$\text{LM-EA}^{\text{name}}_{\text{Wikipedia2Vec}}$	28.20	57.44	36.25	66.26	53.53	79.21	21.11	40.12
$\text{LM-EA}^{\text{name}}_{\text{mBERT}}$	16.28	33.14	14.36	32.50	76.19	85.13	57.64	72.27
$\text{LM-EA}^{\text{name}}_{\text{Word2Vec}}$	X	X	X	X	X	X	11.03	26.37
$\text{LM-EA}^{\text{description}}_{\text{Wikipedia2Vec}}$	28	56.79	35.89	66.07	53.02	77.81	X	X

6 Conclusion

6.1 Summary

In this thesis, an embedding-based machine learning approach for entity alignment has been presented. The proposed model named LM-EA utilizes LMs to learn the embeddings of the entities in the source and the target KG respectively and performs EA in a supervised as well as an unsupervised setting. The LMs used for the task are Word2Vec, FastText, Wikipedia2Vec and mBERT. The alignment is inspired by the work of [11] which proposes a word alignment method for machine translation.

After introducing foundations in the field of Semantic Web and NLP in Chapter 2 and summarizing relevant literature in Chapter 3, this thesis further explored the idea and intuition behind ***LM-EA***. The pipeline and the implementation were explained in Chapter 4. Then, the datasets used, and their generation, were discussed in Chapter 5. ***LM-EA*** was evaluated on real-world benchmark datasets, including three multilingual KG pairs and one monolingual KG pair. The performance of ***LM-EA*** is compared to existing benchmark studies.

6.2 Future Work

At present, the alignment of KG entities based on representation learning is still the mainstream research method. Models based on semantic matching are efficient, but the results often rely on pre-aligned entity pairs to a large extent. This thesis proposes an unsupervised EA approach and sets the beginning for further research in this direction.

Another challenge for the EA task is the heterogeneity of the KGs. Our method is the first which doesn't leverage the KG structure and overcomes this challenge, but we believe the exploration of EA models which don't take into account the KG structure but instead use external information is a subject of further research.

Bibliography

[1] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. "Learning bilingual word embeddings with (almost) no bilingual data". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 451–462. DOI: 10.18653/v1/P17-1042. URL: https://aclanthology.org/P17-1042.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (May 2001), pp. 34–43. URL: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[3] Russa Biswas, Mehwish Alam, and Harald Sack. "Is Aligning Embedding Spaces a Challenging Task? An Analysis of the Existing Methods". In: *CoRR* abs/2002.09247 (2020). arXiv: 2002.09247. URL: https://arxiv.org/abs/2002.09247.

[4] Russa Biswas et al. "Cat2Type: Wikipedia Category Embeddings for Entity Typing in Knowledge Graphs". In: *Proceedings of the 11th on Knowledge Capture Conference*. K-CAP '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 81–88. ISBN: 9781450384575. DOI: 10.1145/3460210.3493575. URL: https://doi.org/10.1145/3460210.3493575.

[5] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked Data - The Story So Far". In: *Int. J. Semantic Web Inf. Syst.* 5 (2009), pp. 1–22.

[6] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://aclanthology.org/Q17-1010.

[7] Kurt Bollacker et al. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. Vancouver, Canada: Association for Computing Machinery, 2008, pp. 1247–1250. ISBN: 9781605581026. DOI: 10.1145/1376616.1376746. URL: https://doi.org/10.1145/1376616.1376746.

[8] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-Relational Data". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 2787–2795.

[9] Muhao Chen et al. "Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-lingual Entity Alignment". In: *IJCAI*. 2018.

[10] Muhao Chen et al. *Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-lingual Entity Alignment*. 2018. DOI: 10.48550/ARXIV.1806.06478. URL: https://arxiv.org/abs/1806.06478.

[11] Alexis Conneau et al. *Word Translation Without Parallel Data*. 2017. DOI: 10.48550/ARXIV.1710.04087. URL: https://arxiv.org/abs/1710.04087.

[12] Daniel Daza, Michael Cochez, and Paul Groth. "Inductive Entity Representations from Text via Link Prediction". In: *CoRR* abs/2010.03496 (2020). arXiv: 2010.03496. URL: https://arxiv.org/abs/2010.03496.

[13] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[14] Martin J. Dürst and Michel Suignard. "Internationalized Resource Identifiers (IRIs)". In: *RFC* 3987 (2005), pp. 1–46.

[15] Dieter Fensel et al. *Knowledge Graphs - Methodology, Tools and Selected Use Cases*. Ed. by Dieter Fensel. Springer, 2020.

[16] Yunjun Gao et al. "ClusterEA: Scalable Entity Alignment with Stochastic Training and Normalized Mini-batch Similarities". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022. DOI: 10.1145/3534678.3539331. URL: https://doi.org/10.1145%2F3534678.3539331.

[17] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[18] Thomas Hubauer et al. "Use Cases of the Industrial Knowledge Graph at Siemens". In: *SEMWEB*. 2018.

[19] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[20] Jens Lehmann et al. "DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web, vol. 6* (2015). DOI: 10.3233/sw-140134.

[21] Manuel Leone et al. "A Critical Re-Evaluation of Neural Methods for Entity Alignment". In: *Proc. VLDB Endow.* 15.8 (2022), pp. 1712–1725. ISSN: 2150-8097. DOI: 10.14778/3529337.3529355. URL: https://doi.org/10.14778/3529337.3529355.

[22] Xin Mao et al. "MRAEA: An Efficient and Robust Entity Alignment Approach for Cross-Lingual Knowledge Graph". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. WSDM '20. Houston, TX, USA: Association for Computing Machinery, 2020, pp. 420–428. ISBN: 9781450368223. DOI: 10.1145/3336191.3371804. URL: https://doi.org/10.1145/3336191.3371804.

[23] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. *Exploiting Similarities among Languages for Machine Translation*. 2013. DOI: 10.48550/ARXIV.1309.4168. URL: https://arxiv.org/abs/1309.4168.

[24] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546 [cs.CL].

[25] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[26] Jeff Z. Pan et al. *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 3319456520.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[28] Matthew E. Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[29] Richard Socher et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *EMNLP*. 2013.

[30] 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*. WWW '07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 697–706. ISBN: 9781595936547. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[31] Zequn Sun, Wei Hu, and Chengkai Li. *Cross-lingual Entity Alignment via Joint Attribute-Preserving Embedding*. 2017. DOI: 10.48550/ARXIV.1708.05045. URL: https://arxiv.org/abs/1708.05045.

[32] Zequn Sun et al. "A Benchmarking Study of Embedding-based Entity Alignment for Knowledge Graphs". In: *CoRR* abs/2003.07743 (2020). arXiv: 2003.07743. URL: https://arxiv.org/abs/2003.07743.

[33] Xiaobin Tang et al. "BERT-INT:A BERT-based Interaction Model For Knowledge Graph Alignment". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, July 2020, pp. 3174–3180. DOI: 10.24963/ijcai.2020/439. URL: https://doi.org/10.24963/ijcai.2020/439.

[34] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. "Entity Alignment between Knowledge Graphs Using Attribute Embeddings". In: AAAI Press, 2019. ISBN: 978-1-57735-809-1. URL: https://doi.org/10.1609/aaai.v33i01.3301297.

[35] Denny Vrandečić and Markus Krötzsch. "Wikidata: A Free Collaborative Knowledgebase". In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489.

[36] Yuting Wu et al. *Jointly Learning Entity and Relation Representations for Entity Alignment*. 2019. DOI: 10.48550/ARXIV.1909.09317. URL: https://arxiv.org/abs/1909.09317.

[37] Yuting Wu et al. "Relation-Aware Entity Alignment for Heterogeneous Knowledge Graphs". In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019. DOI: 10.24963/ijcai.2019/733. URL: https://doi.org/10.24963%2Fijcai.2019%2F733.

[38] Kun Xu et al. *Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network*. 2019. DOI: 10.48550/ARXIV.1905.11605. URL: https://arxiv.org/abs/1905.11605.

[39] Ikuya Yamada et al. "Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 23–30.

[40] Hsiu-Wei Yang et al. *Aligning Cross-Lingual Entities with Multi-Aspect Information*. 2019. DOI: 10.48550/ARXIV.1910.06575. URL: https://arxiv.org/abs/1910.06575.

[41] Zhilin Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020. arXiv: 1906.08237 [cs.CL].

[42] Tom Young et al. *Recent Trends in Deep Learning Based Natural Language Processing*. 2018. arXiv: 1708.02709 [cs.CL].

[43] Ziheng Zhang et al. *An Industry Evaluation of Embedding-based Entity Alignment*. 2020. DOI: 10.48550/ARXIV.2010.11522. URL: https://arxiv.org/abs/2010.11522.

[44] Xiang Zhao et al. "An Experimental Study of State-of-the-Art Entity Alignment Approaches". In: *IEEE Transactions on Knowledge and Data Engineering* 34.6 (2022), pp. 2610–2625. DOI: 10.1109/TKDE.2020.3018741.