Link Prediction in Knowledge Graphs Using Graph Embedding

Emil S. Bækdahl
ebakda16@student.aau.dk
Group d806f20

Abstract—Knowledge bases are becoming increasingly popular in both academia and industry since they provide an intuitive way to reason about data and thus can support decision making. Recently, an interest in inferring new knowledge from existing knowledge bases has seen a rise. For instance, in the medical domain, inference can be used to propose new drugs or treatments for known diseases. In this project, we examine a state-of-the-art approach to this problem that uses graph embeddings to predict missing links in knowledge bases. Particularly, we focus on a recent approach called the hierarchy-aware knowledge graph embedding and evaluate its performance on existing and new datasets. As a conclusion, we are not able to reproduce the results of the associated paper, and experiments on a new domain-specific dataset exhibit significantly poor performance.

1 Introduction

Representation of knowledge in a way that suits computation is at the core of many machine intelligence tasks and agents. A collection of such knowledge is called a knowledge base (KB) and it describes how different entities in a domain are related. A KB can be constructed either manually or automatically by computer programs that extract knowledge from existing sources. Either way, during the construction, there might be a trade-off between completeness and correctness [1]. Completeness describes how well a KB captures all knowledge in the desired domain while correctness describes the degree of truth of the knowledge. When creating a KB by hand, it is possible to manually verify the correctness of the knowledge. However, it can be an infeasible task to manually cover all knowledge in larger domains, resulting in an incomplete KB. To speed up this process, automatic methods, such as building computer programs that extract knowledge, can be used. However, while automatically extracting knowledge from many different sources can give a complete KB, it may result in creating false knowledge. The trade-off lies in finding a reasonable balance between correct and complete knowledge.

As a consequence of this issue, it is attractive to develop solutions that improve correctness or completeness of KBs. This task is called KB refinement and approaches to it are either concerned with (a) increasing completeness by inferring new or hidden knowledge from existing knowledge, or (b) increasing correctness by finding and removing false knowledge. Especially the former sees a gain in popularity since it can provide otherwise undiscovered insight into existing data and thus help humans in decision making.

One approach to computationally increasing completeness is through link prediction. The goal of this task is to infer new knowledge by identifying missing links between existing entities in a KB. For link prediction, the current most popular methods are based on vector embeddings of KBs. In this project, we take a closer look at this state-of-the-art technique. Specifically, we focus on a recent technique that uses a hierarchy-aware knowledge graph embedding (HAKE) of KBs to infer missing links between entities in a hierarchical graph structure [2]. We explain the underlying concepts for this technique and conduct experiments with existing datasets and a new domain-specific dataset.

The rest of this report is structured as follows. Section 2 discusses existing work on KBs and techniques for graph embedding and link prediction. In Section 3, we introduce preliminary concepts in the field which will be the foundation for understanding the HAKE technique in Section 4. Using the technique, we conduct experiments and present their results in Section 5. In Section 6, we discus the results and highlight some key ideas that can point in directions of future work. We end with a conclusion in Section 7.

2 Related Work

In this section, we touch upon two categories of related work: KB sources and link prediction techniques.

A number of freely available and open source KBs, created in different ways, are described in [1]. The two most notable that cover general knowledge are Wikidata and DBpedia. Both represent the knowledge found in the online encyclopaedia Wikipedia, but they do it in two distinct ways. Wikidata is the underlying provider of knowledge to Wikipedia with data curated by the community [3]. On the other hand, DBpedia extracts knowledge directly from the web pages of Wikipedia based on their text content [4].

Yet another great ontology (YAGO) [5] is a KB similar to DBpedia in the sense that most of its knowledge is extracted from Wikipedia. However, the semantic lexicon WordNet is used to resolve ambiguity problems and gives YAGO better correctness. Since YAGO is based on data from multiple sources, it naturally contains more knowledge compared to DBpedia. Moreover, Wikidata, DBpedia, and YAGO rely on

data that are already structured. Other approaches to constructing KBs are more automatic and based on unstructured data, for instance NELL [6] and Knowledge Vault [7].

All of the mentioned KBs can be accessed in a document format, typically RDF. Furthermore, Wikidata, DBpedia, and YAGO provide a SPARQL service for remote querying. In this project, we will use Wikidata to construct a domain-specific dataset for experiments since it contains the most up-to-date data and provides a stable SPARQL interface.

Approaches to the link prediction problem come in a variety of flavours. Some of the simplest techniques are based on similarity between nodes in a graph representation of a KB [8]. Here, the heuristic is that similar nodes are more likely to be linked. The major challenge with these approaches is that similarity can be measured in many different ways. In [8] alone, more than 20 measures are described and the authors add that the performance of the different measures may vary from case to case. As such, deciding on a good similarity measure is not a trivial task. Other techniques approach link prediction as a matrix factorisation problem. As an example, the RESCAL [9] method factorises a three-dimensional matrix representation of a KB and produces a model that captures features of the data which can be used to predict missing links.

Many recent link prediction techniques are based on vector representations of KBs. This representation is called a graph embedding and in the context of link prediction, it can be used in a number of ways. Currently, one of the most popular ways to use graph embeddings for link prediction is in translational models. In a translational model, knowledge about entities is represented as translations in the graph embedding. One of the first link prediction methods based on this approach is TransE [10]. TransE have since been extended by the RotatE method [11] which is based on embeddings in a complex vector space. Using this approach, RotatE has the ability to model complicated KB relations which is not possible with TransE. Recently, the RotatE method inspired the HAKE technique [2] which is based on embeddings in a polar coordinate system. The purpose of the method is to exploit the nature of polar coordinates to persist the hierarchical nature of knowledge. The HAKE technique currently produces state-of-the-art results in the link prediction task.

In this paper, we will take a closer look at the HAKE technique by examining its underlying theory and replicating the experiments from the original paper. We also test the technique further by applying it to a domain-specific KB extracted from Wikidata.

3 Preliminaries

In the following, we cover a number of topics that lay the foundation for the HAKE technique and how it is used for link prediction. In Section 3.1, we introduce KBs at the conceptual level and describe a way of representing them as graphs. Section 3.2 focuses on graph embedding and how they can be used for link prediction

3.1 Knowledge Bases

A knowledge base (KB) is a collection of facts that explain how different concepts in a certain domain are related. Throughout this project, we will use the medical domain for exemplification, however, KBs can also span multiple domains. The constituents of a KB are divided into two categories: entities and relations. An entity can be a concrete object or an abstract concept. In our case study, an example of an entity is a concrete drug such as aspirin, but it can also something more abstract such as the class of infectious diseases. A relation describes a connection between two entities. For instance, a relation can describe a medical treatment by linking a drug and a disease. Together, entities and relations form facts. Formally, we define a KB in the following way.

Definition 1 (Knowledge base). Given a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$, a knowledge base (KB) is a set of triples $\mathcal{K} = \{(h, r, t) \mid h, t \in \mathcal{E} \wedge r \in \mathcal{R}\}$ where each triple is a fact describing a relation r between a head entity h and a tail entity t.

Example 1 (A simple knowledge base). Suppose we want to represent the fact that pain can be treated with aspirin. For this we need two entities, PAIN and ASPIRIN, and one relation, TREATED WITH. Together, these make up one fact in the KB $\mathcal{K}$ which, based on Definition 1, can be formulated in the following way.

$$\mathcal{E} = \{\text{PAIN}, \text{ASPIRIN}\}$$
$$\mathcal{R} = \{\text{TREATED WITH}\}$$
$$\mathcal{K} = \{(\text{PAIN}, \text{TREATED WITH}, \text{ASPIRIN})\}$$

In order to defined the link prediction problem introduced in Section 1, we first define a more general problem. Inference in KBs is concerned with the completeness aspect of KB refinement and is defined in the following way.

Definition 2 (Knowledge base inference). Let $\mathcal{K}^*$ be a KB of all true knowledge and $\mathcal{K}$ be a KB of observed knowledge such that $\mathcal{K} \subset \mathcal{K}^*$. The problem of KB inference is to compute the missing knowledge $\mathcal{K}'$ in $\mathcal{K}$ such that $\mathcal{K} \cup \mathcal{K}' = \mathcal{K}^*$.

The term "missing knowledge" in Definition 2 can be refer to both relations and entities. In the context of the medical domain, finding a missing relation between a disease and a drug can hint at a new treatment, while discovering missing entities can lead to new drug proposals.

To represent a KB, we can use an edge-labelled directed graph where nodes represent entities and edges represent relations. This graph-based representation is defined in the following.

Definition 3 (Knowledge graph). A knowledge graph (KG) is a 3-tuple (V, E, ϕ) where V is the set of entities, $E \subseteq V \times V$ is the set of relations, and the function $\phi\colon E \to \mathcal{T}$ assigns a relation $\mathcal{T}_i \in \mathcal{T}$ to each $e \in E$.

The direction of a relation in a KG is implied by the direction of its corresponding edge. For instance, the arrow in Fig. 1, which represents the KB from Example 1, indicates that pain is treated with aspirin and not the other way around.

Any KB $\mathcal{K}$ over entities $\mathcal{E}$ and relations $\mathcal{R}$ can be represented as a KG by letting $V = \mathcal{E}$ and $E = \{(h, t) \mid (h, r, t) \in \mathcal{K}\}$. The edge labelling function is defined such that $\phi(h, t) = r$ for each $(h, r, t) \in \mathcal{K}$.

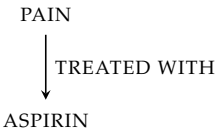

Fig. 1. The KB $\mathcal{K} = \{(\text{PAIN}, \text{TREATED WITH}, \text{ASPIRIN})\}$ represented as a knowledge graph.

Over recent years, KGs have become synonymous with KBs and is likely a more frequently used term. Throughout this project, we use the term KG as an interchange for KB. For example, we can talk about triples (h, r, t), as defined in Definition 1, in the context of a graph.

Now that we have a concrete representation of knowledge in the form of a KG, we can put the inference problem from Definition 2 into context and formulate the precise problem that we address in this project.

Problem formulation Let $\mathcal{K} = (V, E, \phi)$ be a KG where E is the observed relations, or links, between the entities V. Let E^* denote the set of all possible true links. Then the set $E' = E^* \setminus E$ represents the missing links in $\mathcal{K}$. The task of link prediction is to predict whether a given link $e \in V \times V$ such that $e \notin E$ is true, i.e. if $e \in E^*$ [8], [12].

The link prediction problem concerns the completeness aspect of KB refinement, introduced in Section 1, with focus on identifying missing relations. A solution to the problem typically consists of assigning a probability of existence to a potential edge in $V \times V$. In the following section, we will dive deeper into graph embedding which is the foundation for many recent link prediction techniques.

3.2 Graph Embedding

To make graphs more efficient to work with computationally, it can be beneficial to embed them into a vector space [13]. This technique is called graph embedding. The goal of graph embedding is to keep the dimensionality of the target vector space, called the embedding space, low while preserving properties of the graph. Depending on the properties of interest, the embedding can happen on different levels. For instance, in weighted graphs, each edge e between two nodes u and v is associated with a weight $w(e)$ that can be used to represent, for instance, the importance or capacity of e. This weight can be represented in the embedding by letting the distance between u and v in the embedding space approximate $w(e)$. In the context of KGs, however, the properties of interest lie in both the nodes and the edges.

Generally, embedding a graph can be seen as a problem of learning a function g that maps nodes and edges to vectors in an $\mathbb{R}^d$ embedding space. When embedding KGs, this is typically done by defining a score function $f(h, r, t)$ that describes how likely it is for the triple (h, r, t) to exist [11]. This function should be constructed in a way that gives high scores to true triples $(h, r, t) \in \mathcal{K}$ and low scores to false triples $(h', r', t') \notin \mathcal{K}$. If the score function has this property, learning the embedding can be seen as an optimisation problem with respect to f. Specifically, we wish to maximise the value of f for every $(h, r, t) \in \mathcal{K}$. We use bold letters to denote embeddings of nodes and edges, for instance $\boldsymbol{e} = g(e)$ where $\boldsymbol{e}$ is the embedded version of the entity e.

Some KG embedding techniques are referred to as translational models where, given a triple (h, r, t), the relation $\boldsymbol{r}$ acts as a translation from $\boldsymbol{h}$ to $\boldsymbol{t}$ in the embedding space. One of the simplest translational models is TransE [10]. The aim of this technique is to satisfy $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ for each triple $(h, r, t) \in \mathcal{K}$. The intuition behind this translation technique is shown in Fig. 2 where the triple (h, r, t) is embedded as $\boldsymbol{h} = (0.5, 0.5)$, $\boldsymbol{t} = (2.5, 1.5)$, and $\boldsymbol{r} = (2, 1)$. This clearly satisfies the objective since $(0.5, 0.5) + (2, 1) = (2.5, 1.5)$.

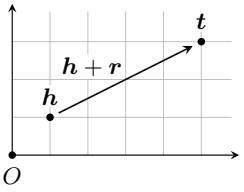

Fig. 2. The TransE model [10] aims to satisfy $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$ for every triple (h, r, t). In this case $\boldsymbol{h} = (0.5, 0.5)$, $\boldsymbol{t} = (2.5, 1.5)$, and $\boldsymbol{r} = (2, 1)$.

It should be noted that the embedding in Fig. 2 inhibits the $\mathbb{R}^2$ space since it makes for an intuitive illustration. However, a graph embedding usually occupies a space of higher dimensions since it allows for more details to be captured.

It is clear that if a TransE embedding satisfies $\boldsymbol{h} + \boldsymbol{r} = \boldsymbol{t}$, we can use $\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}$ as a measure of correctness; the closer the value of f is to 0, the more correct the embedding is. This leads to the TransE score function which is defined as $f(h, r, t) = -\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_p^2$. Here, $\| \cdot \|_p$ denotes the p-norm which has the general form

$$\|\boldsymbol{e}\|_p = \sqrt[p]{\boldsymbol{e}_1^p + \boldsymbol{e}_2^p + \cdots + \boldsymbol{e}_i^p}.$$

If the TransE embedding is perfect, true triples will have a score of 0 while false triples will have negative scores.

TransE is a simple model but it lacks the ability to model complicated relations often found in KGs. Examples of these are (a) symmetric relations where both (h, r, t) and (t, r, h) are present, (b) inverse relations where r is an inverse of r' if both (h, r, t) and (t, r', h) exist, and (c) composed relations where r is a composition of r' and r'' if (h, r, t), (h, r', t'), and (t', r'', t) are present. This is an issue addressed by the RotatE approach [11]. This technique embeds triples into a complex embedding space where relations represent rotational translations between entities. More specifically, given a triple (h, r, t), RotatE aims to satisfy $\boldsymbol{h} \circ \boldsymbol{r} = \boldsymbol{t}$. Here, $\circ$ denotes the element-wise product of two vectors, meaning that if $\boldsymbol{t} = \boldsymbol{h} \circ \boldsymbol{r}$ then $\boldsymbol{t}_i = \boldsymbol{h}_i \boldsymbol{r}_i$. The advantage of RotatE is that it is capable of modelling the different types of relations described above.

In addition to complicated relations, KGs often also exhibit semantic hierarchies. For instance, in the medical domain, drugs are grouped in different categories such as antipyretics and vaccines. The HAKE technique [2], which build upon RotatE, is developed specifically to handle such structures. The purpose of the technique is to persist semantic hierarchies in the graph embedding. This is done by dividing the embedding of entities into two parts: one that represents their hierarchical levels and one that distinguishes them on the same level. Then the relation embeddings act as translations between the entities.

A KG embedding can be used for link prediction by passing candidate triples through the score function. The triples that get assigned the highest scores are most likely to exist. The HAKE technique currently produces state-of-the-art results for this link prediction task, which is why we take a closer look at it in the next section.

4 The HAKE Technique

This section will cover the intuition behind the HAKE technique and how it is used for link prediction. We start, in Section 4.1, by giving an intuitive understanding of the technique by introducing its overall goal and concrete concepts. In Section 4.2, we take a look at how these concepts are deployed in a learning setting.

4.1 Representing Hierarchies in Embeddings

As stated earlier, the HAKE technique embeds entities and relations into a polar coordinate system. In such coordinate system, a point is described by its distance from the origin as well as its angle relative to a reference direction. These constituents are referred to as the radial coordinate, denoted ρ, and angular coordinate, denoted ϕ, respectively. Given an embedded entity $\boldsymbol{e} = (\boldsymbol{e}_\rho, \boldsymbol{e}_\phi)$, we use $\boldsymbol{e}_\rho$ to denote the radial part and $\boldsymbol{e}_\phi$ to denote the angular part of the corresponding entity. This is illustrated in Fig. 3 where the entity $\boldsymbol{e}$ has radial coordinate 2 and angular coordinate 0.8. As in TransE, embeddings usually live in a high dimensional space. For the sake of illustrative examples, however, we use the $\mathbb{R}^2$ space.

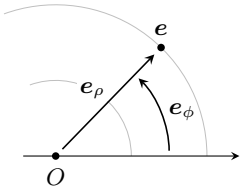

Fig. 3. The embedded entity $\boldsymbol{e} = (2, 0.8)$ in the polar coordinate system.

The motivation behind this two-fold embedding is, as mentioned earlier, that each component models a distinct aspect of the entity. Fig. 4 illustrates entities from a medical domain. Here, DRUG is the most general entity and thus lies at the top of the hierarchy. One step down the hierarchy and we get a certain class of drugs, ANTIPYRETICS. The entities ASPIRIN and IBUPROFEN are instances of ANTIPYRETICS and thus represent another step down the hierarchy. Entities on the same level of the hierarchy have the same radial coordinate and different angular coordinates.

Since the HAKE technique relies on a translational model like TransE and RotatE, it aims to achieve a similar goal. For each triple (h, r, t), the embedded relation $\boldsymbol{r}$ acts as a translation from $\boldsymbol{h}$ to $\boldsymbol{t}$. Since each entity consists of two parts, radial and angular, the objective of this translation is two-fold. For the radial coordinate, the translation is a scaling operation, i.e. for each triple (h, r, t), we want to satisfy

$$\boldsymbol{h}_\rho \circ \boldsymbol{r}_\rho = \boldsymbol{t}_\rho. \tag{1}$$

Once again, the $\circ$ operator is the element-wise product, i.e. $\boldsymbol{h}_{\rho_i}\boldsymbol{r}_{\rho_i} = \boldsymbol{t}_{\rho_i}$. This is what allows the translation to move

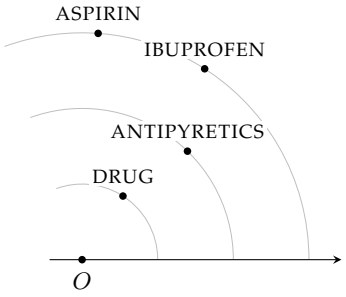

Fig. 4. The goal of the representation in the HAKE technique is that entities of the same hierarchical level, such as ASPIRIN and IBUPROFEN, have similar radial coordinates but different angular coordinates.

from one level of the semantic hierarchy, namely $\boldsymbol{h}_\rho$, to another level $\boldsymbol{t}_\rho$.

For the angular coordinate, the translation is a rotation that, for each triple (h, r, t), should satisfy

$$\boldsymbol{h}_\phi + \boldsymbol{r}_\phi \bmod 2\pi = \boldsymbol{t}_\phi. \tag{2}$$

The angular translation of an entity $\boldsymbol{e} = (\boldsymbol{e}_\rho, \boldsymbol{e}_\phi)$ can be seen as moving along the circumference of a circle with radius $\boldsymbol{e}_\rho$ starting at angle $\boldsymbol{e}_\phi$ rad. This translation has a periodic nature in the interval $[0; 2\pi]$ which is the reason for the modulo operator in (2).

Example 2 (Relations as translations). To illustrate how a relation acts as a translation from one entity to another, we consider Fig. 5 that shows an embedding of the triple (h, r, t). Here, we assume $\boldsymbol{h} = (1, 0.3)$, $\boldsymbol{t} = (3, 0.8)$, and $\boldsymbol{r} = (3, 0.5)$. This embedding satisfies (1) and (2) since $1 \cdot 3 = 3$ and $0.3 + 0.5 \bmod 2\pi = 0.8$ respectively.

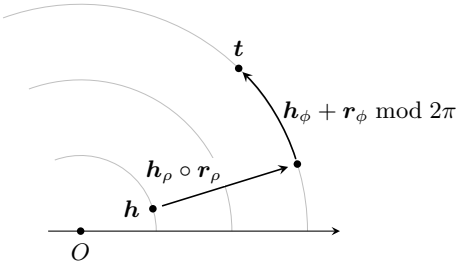

Fig. 5. When the triple (h, r, t) is embedded, the relation $\boldsymbol{r} = (3, 0.5)$ acts as a translation from $\boldsymbol{h} = (1, 0.3)$ to $\boldsymbol{t} = (3, 0.8)$.

When learning the embedding, we must define a measure of similarity or dissimilarity between our current embedding of a triple (h, r, t), and a true perfect embedding. We do this through a distance function which intuitively describes how good an embedding is at satisfying (1) and (2). For the radial and angular component, we define separate distance functions. For the radial component, we have the distance function

$$\mathrm{d}_\rho(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = \|\boldsymbol{h}_\rho \circ \boldsymbol{r}_\rho - \boldsymbol{t}_\rho\|_2 \tag{3}$$

and for the angular component, the distance function is defined as

$$\mathrm{d}_\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = \left\| \sin \frac{\boldsymbol{h}_\phi + \boldsymbol{r}_\phi - \boldsymbol{t}_\phi}{2} \right\|_1. \tag{4}$$

We define the final distance function as the sum of (3) and (4) as seen in (5).

$$\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = \mathrm{d}_\rho(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) + \mathrm{d}_\phi(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) \tag{5}$$

It is clear that if the embedding is perfect, we will have $\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = 0$ for every triple (h,r,t). Similarly, the more off the embedding is, the larger the distance value.

Example 3 (Distance functions). We reconsider Fig. 5 from Example 2. Applying (5) to the valid triple (h,r,t) gives us a distance of

$$\sqrt{(1\cdot 3-3)^2} + \sin\frac{0.3+0.5-0.8}{2} = 0$$

as expected. Now, assume we introduce a new triple (h,r',t) to the KG. Initially, we have $\boldsymbol{r'} = (2, 1.3)$ and by applying the distance function (5) we get

$$\sqrt{(1\cdot 2-3)^2} + \sin\frac{0.3+1.3-0.8}{2} = 1.389.$$

This will get penalised in the learning process and the embedding of r' will be adjusted.

4.2 Learning the Embedding

For the learning process, we must first define the score function. In the HAKE technique, the distance function (5) is at the core of the score function. However, to satisfy the property that the score function must assign high scores to correct embeddings and lower scores to wrong embeddings, we invert the sign as seen in (6). In this way, a perfect embedding will get the highest score 0.

$$f(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = -\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) \tag{6}$$

Furthermore, the weights λ and μ are added to each term in the distance function. These weights are treated as hyperparameters and will be learned by the model together with the entity and relation embeddings.

$$\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = \lambda\mathrm{d}_\rho(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) + \mu\mathrm{d}_\phi(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) \tag{7}$$

The HAKE method uses negative sampling [14] when learning the embedding. The purpose of this sampling technique is to cope with the softmax function typically used when learning classification problems. The softmax function maps a real-valued vector to a vector of numbers between 0 and 1 that can be treated as a probability distribution. Given a KG $\mathcal{K}$ and a score function f, we can compute the softmax value for a triple $(h,r,t) \in \mathcal{K}$ as

$$\frac{e^{f(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t})}}{\sum_{(h',r',t')\in\mathcal{K}} e^{f(\boldsymbol{h'},\boldsymbol{r'},\boldsymbol{t'})}}.$$

The problem here is the summation in the denominator which becomes increasingly expensive to compute as the KG grows. According to [1], many freely available KGs contain billions of facts making softmax computation infeasible.

Negative sampling provides a way around this by generating a number of negative triples $(h',r,t') \notin \mathcal{K}$ for every positive triple $(h,r,t) \in \mathcal{K}$. Based on a positive triple (h,r,t), a negative triple is either on the form (h',r,t) or (h,r,t') where h' and t' are sampled from the entities in the KG. For convenience, we describe negative triples given a positive triple (h,r,t) as the set

$$\begin{aligned}\mathcal{N}_{(h,r,t)} = &\{(h,r,t') \mid t' \in \mathcal{E} \wedge (h,r,t') \notin \mathcal{K}\} \\ &\cup \{(h',r,t) \mid h' \in \mathcal{E} \wedge (h',r,t) \notin \mathcal{K}\}.\end{aligned} \tag{8}$$

Furthermore, we use the notation $\mathcal{N}_{(h,r,t)_n}$ for the set of n uniformly sampled negative triples.

As in many other machine learning problems, we wish to minimise a certain loss function in the learning process. We use the distance function (5) and negative sampling to define the loss function as

$$\begin{aligned}&-\log\sigma(\gamma - \mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t})) \\ &-\sum_{(h',r,t')\in\mathcal{N}_{(h,r,t)_n}} P(h',r,t')\log\sigma(\mathrm{d}(\boldsymbol{h'},\boldsymbol{r},\boldsymbol{t'}) - \gamma),\end{aligned} \tag{9}$$

where γ is a positive margin parameter and σ is the sigmoid function,

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

To get an intuition of (9), we notice two things. First, from (5) we know that the value of the distance function cannot be negative. Second, we note that the value of $\log\sigma(\cdot)$ grows as its parameter increases. Therefore, the first term gets the lowest value when $\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = 0$. In the second term, the order of γ and $\mathrm{d}(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t})$ is flipped. This means that the value for the second term gets lower as the distances of the negative triples get higher. Immediately, this satisfies the objectives of the HAKE discussed in the previous section. Moreover, each negative triple in (9) is multiplied by a weight P defined as

$$P(h',r,t') = \frac{e^{\alpha f(\boldsymbol{h'},\boldsymbol{r},\boldsymbol{t'})}}{\sum_{(h'',r,t'')\in\mathcal{N}_{(h,r,t)_n}} e^{\alpha f(\boldsymbol{h''},\boldsymbol{r},\boldsymbol{t''})}},$$

where α is the sampling temperature.

Contrary to what we described earlier, this weight is actually a softmax distribution. In this case, the computation is not problematic since it is dependent on a fixed size of samples, namely the negative sample size n, and not the size of the entire dataset $|\mathcal{K}|$.

Now, with all necessary concepts defined, we formulate the learning process which is illustrated in Fig. 6. First, we initialise the entity and relation embeddings on Line 1 and 2. This is done by creating an embedding vector $\boldsymbol{e} \in \mathbb{R}^{d2}$ for each $e \in \mathcal{E}$ and setting its entries $\boldsymbol{e}_i$ to values drawn from the uniform distribution $\mathcal{U}$. The relation embeddings go through the same procedure. The main learning loop is based on mini-batch gradient decent using the Adam optimiser [15]. In each iteration, we sample a batch of b true triples $\mathcal{B}^+$ from the training dataset $\mathcal{K}$ on Line 5. For each true triple, we sample n negative triples in the batch $\mathcal{B}^-$ on Line 7. Together, these form the training batch $\mathcal{B}$ composed of pairs of a true triple and a set of negative triples as seen on Line 8. Based on $\mathcal{B}$, we compute the sum of losses using (9) and update the parameters of the model by taking a gradient step on Line 10. The parameters that get updated are the entity embedding, the relation embedding, the radial weight λ, and the angular weight μ.

Fig. 6. Pseudo code for the training procedure of the HAKE technique inspired by [10].

Require: Training dataset $\mathcal{K} = \{(h, r, t) \mid h, t \in \mathcal{E} \wedge r \in \mathcal{R}\}$, dimensionality of embedding space d, margin γ, batch size b, and negative sample size n.

$\boldsymbol{e}_i \leftarrow \mathcal{U}(-\frac{\gamma}{d}, \frac{\gamma}{d})$ for each $e \in \mathcal{E}$, where $\boldsymbol{e} \in \mathbb{R}^{2d}$
$\boldsymbol{r}_i \leftarrow \mathcal{U}(-\frac{\gamma}{d}, \frac{\gamma}{d})$ for each $r \in \mathcal{R}$, where $\boldsymbol{r} \in \mathbb{R}^{2d}$
loop
  $\mathcal{B} \leftarrow \emptyset$
  $\mathcal{B}^+ \leftarrow \text{sample}(\mathcal{K}, b)$
  for $(h, r, t) \in \mathcal{B}^+$ **do**
    $\mathcal{B}^- \leftarrow \text{sample}(\mathcal{N}_{(h,r,t)}, n)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\big((h, r, t), \mathcal{B}^-\big)\}$
  end for
  Update parameters w.r.t $\sum_{((h,r,t),\mathcal{B}^-)\in\mathcal{B}} \nabla\big(-\log\sigma(\gamma - \mathrm{d}(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})) - \sum_{(h',r,t')\in\mathcal{B}^-} P(h', r, t') \log\sigma(\mathrm{d}(\boldsymbol{h'}, \boldsymbol{r}, \boldsymbol{t'}) - \gamma)\big)$
end loop

5 Experiments

We conduct two different experiments in the following sections to evaluate the HAKE technique for link prediction. While both use the same setup and evaluation methods, they examine different aspects of the technique. In Section 5.1, we cover the setup of both experiments as well as the metrics used for evaluation. Section 5.2 and 5.3 explain the different experiments and present their results.

5.1 Setup and Evaluation Metrics

The testing procedure used for the HAKE technique originate from TransE [10] and has been used to evaluate a variety of related link prediction methods. The core of the test dataset is a set of true triples $\mathcal{S}$ that do not overlap with the training data $\mathcal{K}$. For each true triple $(h, r, t) \in \mathcal{S}$, we generate a set of false triples by replacing either h or t by entities sampled from the KG. The union of the true and false triples make up the final test dataset $\mathcal{S}'$. It should be noted that following this generating process may produce triples that appear in the training or validation datasets. Therefore, to avoid skewed results, we remove such triples from the test dataset. Generally, given a set of true test triples $\mathcal{S}$ over entities $\mathcal{E}$ and a set training triples $\mathcal{K}$, we can formulate the test dataset as

$$\begin{aligned}\mathcal{S}' = \mathcal{S} &\cup \{(h, r, t') \mid (h, r, t) \in \mathcal{S} \wedge t' \in \mathcal{E} \wedge (h, r, t') \notin \mathcal{K}\} \\ &\cup \{(h', r, t) \mid (h, r, t) \in \mathcal{S} \wedge h' \in \mathcal{E} \wedge (h', r, t) \notin \mathcal{K}\}.\end{aligned}$$

The triples in $\mathcal{S}'$ are now ranked by their score (6) such that the triple with the highest score is assigned rank 1 and the triple with the lowest score is assigned rank $|\mathcal{S}'|$.

For evaluation, we use the measures mean reciprocal rank (MRR) and hits-at-n (H@N) since they are widely adopted for link prediction techniques. The MRR is calculated as

$$\text{MRR} = \frac{1}{|\mathcal{S}|} \sum_{(h,r,t)\in\mathcal{S}} \frac{1}{\text{rank}(h, r, t)}. \tag{10}$$

Here, $\mathcal{S}$ is the set of true triples and the rank function returns the rank of a triple in entire test dataset $\mathcal{S}'$. Clearly, as more true triples are assigned low ranks, meaning that their reciprocal ranks are high, the MRR increases. Also worth noticing is that as we move down the ranked list of triples, the exact rank of a true triple has less impact on the overall MRR. That is, a change in rank from 200 to 199 has minimal impact compared to a change from 2 to 1.

The H@N measure is also based on the ranked set of triples. Given an n, the H@N value is the proportion of correct predictions in the first n elements of the ranked list as seen below.

$$\text{H@N} = \frac{|\text{correct predictions}|}{n} \tag{11}$$

Similarly MRR, a higher H@N value is better. If all of the top n ranked triples are true, the H@N will be 1.

For the sake of replication, we run both experiments using the Python implementation of the HAKE technique from [2].[1] This implementation is based on the neural network library PyTorch.

5.2 Replication

This experiment is carried out by running the exact experiments presented in [2]. Here, the HAKE technique is evaluated on the three standard datasets WN18RR, FB15k-237, and YAGO3-10 that are based on the KGs WordNet, Freebase, and YAGO respectively. In the original experiments, the authors perform hyperparameter optimisation to find the best parameter configurations for each dataset. We use the same parameter configurations in the replication experiments.

The results of the replication experiments are shown in Table 1. The bottom row indicates how many percent the replicated results deviate from the original results. This deviation is calculated by

$$\frac{\text{replication} - \text{original}}{\text{replication}} \cdot 100\,\%.$$

By looking at the deviation, it is clear that the different datasets give rise to different results. On the WordNet dataset, the replication experiment ranges from performing −1.77 % worse to 0.172 % better. We do not consider this a significant deviation and it could just be due to differences in the initialisation of the entity and relation embeddings. However,

1. The code for the original HAKE paper is available on Github: https://github.com/MIRALab-USTC/KGE-HAKE.

TABLE 1
Results for Replication Experiment

	WordNet				Freebase				YAGO			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Original [2]	0.497	0.452	0.516	0.582	0.346	0.250	0.381	0.542	0.545	0.462	0.596	0.694
Replication	0.490	0.444	0.508	0.583	0.326	0.224	0.369	0.528	0.491	0.408	0.541	0.637
Deviation [%]	−1.41	−1.77	−1.55	0.172	−5.78	−10.4	−3.15	−2.58	−9.91	−11.7	−9.23	−8.21

replication results on the Freebase and YAGO datasets are much different from the original results. Worth noticing is that the replication results are consistently worse than the original. This indicates that the difference in performance is not only due to differences in the initial embeddings as suggested for the WordNet dataset. It should be noted, however, that due to limitations on computational power, we have not been able to run the experiments more than a few times per dataset. This is in contrast to the results in [2] which are averages of multiple runs. Nonetheless, we think that a −3.58 % to −11.7 % deviation is significant.

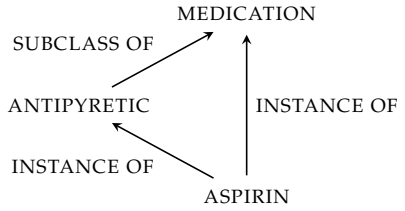

Fig. 7. The HAKE technique does not require a specific structure to the hierarchies in the KG. This means that we, for instance, find non-strict hierarchies in the domain-specific KG, where an entity is related to multiple entities at different hierarchical levels.

5.3 Domain-Specific Data

The goal of this experiment is to find out how the HAKE technique performs on domain-specific data which, compared to the data used in the replication experiment, is non-standard. We construct a KG specific for the medical domain by extracting data from Wikidata using its SPARQL endpoint.[2] In the construction, we explicitly include a class-subclass-instance hierarchy. In Wikidata, the hierarchy is based on the relations SUBCLASS OF[3] and INSTANCE OF[4]. As an example, we consider the entity MEDICATION[5], one of the most general entities in the KG, that occurs in the triple (ANTIPYRETIC, SUBCLASS OF, MEDICATION) which in turn occurs in (ASPIRIN, INSTANCE OF, ANTIPYRETIC).

We believe that this way of constructing the KG with a clear hierarchy makes it suitable for the HAKE technique. However, it should be noted that [2] does not mention any requirements to the hierarchy; for instance, if it must be balanced or strict. This means that, in the domain-specific KG, an entity can be related to entities at different levels of the hierarchy. An example of this seen in Fig. 7 where ASPIRIN is both an instance of ANTIPYRETIC and MEDICATION even though those entities occupy two different levels of the hierarchy.

The medical dataset has 20 422 entities, 5 types of relations, and a total of 34 216 facts. The results from the experiment with domain-specific data is shown in Table 2 where the replication experiment results from Table 1 is included for comparison. We clearly see a significant decrease in performance on the domain-specific dataset.

TABLE 2
Results for Domain-Specific Dataset Experiment

	MRR	H@1	H@3	H@10
WordNet	0.490	0.444	0.508	0.583
Freebase	0.326	0.224	0.369	0.528
YAGO	0.491	0.408	0.541	0.637
Domain-Specific	0.182	0.170	0.171	0.282

6 Discussion

From the experiments in the previous section, we found problems with result replication of the HAKE paper and that the technique does not perform well on domain-specific data. In this section, we provide possible explanations for those encountered problems, and discuss how future research may address the link prediction problem differently.

6.1 Evaluation Protocol

Most link prediction methods that stem from the TransE model [10], such as the HAKE technique, are evaluated on the same standard datasets that we introduced in Section 5.2. Immediately, this gives a good foundation for comparing the different techniques. However, if those datasets are not realistic, the measures do not say much about how the techniques perform in real-world applications. Therefore, it is worth looking at the structure and content of WordNet, Freebase, YAGO, and alike. This issue is addressed in [16]. It turns out that the standard datasets WN18RR, FB15k-237, and YAGO3-10 suffer from two major problems: data redundancy and Cartesian product relations.

Data redundancy is primarily caused by many pairs of triples being reverse of each other, that is both (h, r, t) and (t, r^{-1}, h) being present in the data. Here r^{-1} symbolises the reverse of the relation r. An example of this can be found in Wikidata where the relations MEDICAL CONDITION TREATED[6]

2. The code used for dataset construction can be found on Github: https://github.com/emilbaekdahl/p8-code. It is also attached to the hand-in of this project.
3. https://www.wikidata.org/wiki/Property:P279
4. https://www.wikidata.org/wiki/Property:P31
5. https://www.wikidata.org/wiki/Q12140
6. https://www.wikidata.org/wiki/Property:P2175

and DRUG USED FOR TREATMENT[7] can give rise to the triple (ASPIRIN, MEDICAL CONDITION TREATED, PAIN) which is a reverse of (PAIN, DRUG USED FOR TREATMENT, ASPIRIN). In [16], the authors find that, in datasets with many reverse triple pairs, a simple rule-based link prediction method performs on par with, or sometimes even better than, graph embedding-based techniques. Such rules for predicting reverse triples can be derived from statistical measures of the dataset [17]. Given two relations r and r', we compute the proportion of triples (h, r, t) where h and t are reversely related by r', that is (t, r', h). This measure can be described as

$$\frac{|\{(h,r,t) \mid (h,r,t) \in \mathcal{K} \wedge (t,r',h) \in \mathcal{K}\}|}{|\{(h,r,t) \mid (h,r,t) \in \mathcal{K}\}|}.$$

If this proportion exceeds a given threshold, say 0.8, we conclude the rule

$$(h,r,t) \implies (t,r',h).$$

Repeating this for each pair of relations $(r, r') \in \mathcal{R} \times \mathcal{R}$ will produce a rule-based classifier that yields an H@1 value around 0.7 on certain datasets [16].

Cartesian product relations occur when all elements in a set of head entities H are connected to all elements in a set of tail entities T with a given relation r, that is $\{(h,r,t) \mid (h,r,t) \in \mathcal{K} \wedge (h,t) \in H \times T\}$. The primary issue with Cartesian product relations is that they typically give rise to unrealistic link prediction tasks. According to [16], these types of relations often occur when mediator nodes are used to simplify complicated relations between entities. For instance, if we wanted to extend the KG in Fig. 1 such that the treatment of pain with aspirin is related to the side-effect nausea, we use a mediator node TREATMENT. An illustration of this can be seed in Fig. 8.

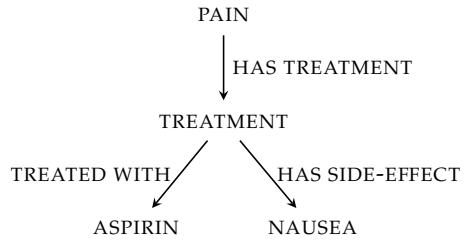

Fig. 8. The TREATMENT node acts as mediator to represent a multi-ary relationship between PAIN and a treatment.

The problem with this is that a part of the link prediction task is to predict (a) whether a specific medical condition has a treatment and (b) whether a specific drug is used for a treatment. These tasks are clearly trivial. Instead, a more interesting and realistic task is to predict (a) what drug is used for treating a specific medical condition and (b) what medical conditions can be treated using a specific drug.

We believe that the absence of reverse and Cartesian product relations is the main explanation of the results from the domain-specific experiments in Section 5.3. The dataset used in the experiment does not contain relations that form reverse or Cartesian product triples.

7. https://www.wikidata.org/wiki/Property:P2176

6.2 Future Work

In its nature, a KG is of course a graph and thus exhibits many interesting properties that can be taken into consideration for link prediction. However, graph embedding-based approaches for link prediction such as TransE, RotatE, and HAKE only consider the triples in isolation when learning the embedding, and thus only learn the structure of the graph. Immediately, it seems that properties and semantics of relations and entities in the KG are lost in this process. To address the link prediction problem effectively, future work should take into consideration not only the structure of the graph, but also the semantics of entities and their relations. We think that one of the first steps to consider such semantics is to cluster entities into groups that share similar properties. In [18] an efficient approach for graph clustering is introduced. Here, similarity between nodes is measured both by their structure and their attributes. This clustering step is applied to star-schema heterogeneous graphs which are defined in the following way.

Definition 4 (Star-schema heterogeneous graph). A heterogeneous graph is a 4-tuple (V, E, ϕ, ψ) where V is a set of nodes, $E \subseteq V \times V$ is a set of edges, $\phi\colon V \to \mathcal{T}_V$ maps each $v \in V$ to a node type $\mathcal{T}_{Vi} \in \mathcal{T}_V$, and $\psi\colon E \to \mathcal{T}_E$ maps each $e \in E$ to an edge type $\mathcal{T}_{Ei} \in \mathcal{T}_E$.

A heterogeneous graph is said to be on star-schema form if (a) node attributes are represented as nodes themselves, called attribute nodes, and (b) nodes that share attributes are connected to the same attribute nodes. Furthermore, non-attribute nodes are referred to as hub nodes, i.e. we have $\mathcal{T}_V = \{\text{hub}, \text{attribute}\}$. In context of the medical domain, for instance, diseases and drugs can be hub nodes while symptoms, effects, and causes are attribute nodes. Fig. 9 illustrates the difference between a "regular" heterogeneous graph and a star-schema heterogeneous graph. The d_i hub nodes represent diseases and are connected to the c_i and s_i attributes nodes representing causes and symptoms.

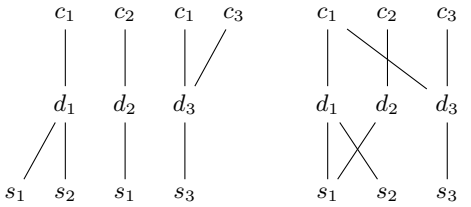

Fig. 9. The normalisation of the attributes in the left heterogeneous graph produces the star-schema graph on the right.

It is clear that we can represent a KG as a star-schema heterogeneous graph since it is a special case of a graph. Since the star-schema structure is capable of capturing similarity both in terms of the properties of entities and how they are related, we think that it may help improving link prediction techniques. Furthermore, most current KG link prediction techniques expect the graph to be homogeneous [13]. As such, using star-schema heterogeneous graphs for future research can also uncover advantages and disadvantages of using heterogeneous graphs in this problem setting.

Another issue that can be addressed by avoiding graph embeddings for link prediction is flexibility and scalability. A graph embedding is fixed to the KG it has been learned on. If we add new knowledge to a KG, either entities or relations, the embedding must be re-computed. This can become expensive and likely infeasible when the frequency of changes or the size of the KG increases. In connection with this, the embedding-based techniques also lack transparency when used for link prediction. If the model predicts a given link, it is not immediately obvious why that link might exist. Explainability is a not an issue that most KG link prediction techniques currently try to address but it may be worth considering in future research since it is an increasingly important topic [19]. Similarity measures on graphs are more clear for humans since the specific attributes or nodes that explain the similarity can be pointed out and inspected.

Therefore, we propose that future work in the field of link prediction, and KG refinement in general, should take similarity into account. This can be done with star-schema heterogeneous graphs as a foundation since they offer a unified similarity measure that exhibits promising results.

7 Conclusion

Representing KGs as low-dimensional vector embeddings has its computational advantages. Querying existing knowledge is fast since arithmetic operations on embedded entities and relations act as translation from head entity to tail entity and vice versa. The HAKE technique, furthermore, has the advantages of being able to model hierarchical knowledge and complex relations. However, the technique is not very precise and without a successful replication experiment, we doubt that the technique is flexible and scalable enough to be implemented in real-world use cases. This argument is substantiated by the fact that performance on domain-specific data is significantly worse than on general-domain knowledge found in standard datasets. We conclude that embedding technique might not be a suitable method for the link prediction problem in KGs. Therefore, a deeper investigation into correct methods for the problem is necessary. Any future solution to this problem should consider not only the structure of the graph, but also the semantics of entities and their relations, for it to work effectively.

References

[1] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, no. 3, pp. 489–508, Dec. 2016.

[2] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," Nov. 2019.

[3] D. Vrandecic and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014.

[4] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - A Large Scale, Multilingual Knowledge Base Extracted From Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[5] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web - WWW '07*, C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, Eds. ACM Press, 2007, pp. 697–706.

[6] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, and T. M. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*. ACM Press, 2010.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*. ACM Press, 2014.

[8] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, Mar. 2011.

[9] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." 01 2011, pp. 809–816.

[10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2787–2795.

[11] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," Feb. 2019.

[12] J. chao Li, D. ling Zhao, B.-F. Ge, K.-W. Yang, and Y.-W. Chen, "A link prediction method for heterogeneous networks based on bp neural network," *Physica A: Statistical Mechanics and its Applications*, vol. 495, pp. 1 – 17, 2018.

[13] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization."

[16] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, "Realistic re-evaluation of knowledge graph completion methods: An experimental study."

[17] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings."

[18] L. Chen, Y. Gao, Y. Zhang, C. S. Jensen, and B. Zheng, "Efficient and Incremental Clustering Algorithms on Star-Schema Heterogeneous Graphs," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, Apr. 2019.

[19] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning."