A PROBABILISTIC ALGORITHM TO PREDICT MISSING FACTS FROM KNOWLEDGE GRAPHS

ANDRÉ LOPES GONZAGA

A PROBABILISTIC ALGORITHM TO PREDICT MISSING FACTS FROM KNOWLEDGE GRAPHS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Mirella Moura Moro
Coorientador: Mário Sérgio Alvim

Belo Horizonte

Janeiro de 2019

ANDRÉ LOPES GONZAGA

A PROBABILISTIC ALGORITHM TO PREDICT MISSING FACTS FROM KNOWLEDGE GRAPHS

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Mirella Moura Moro
Co-Advisor: Mário Sérgio Alvim

Belo Horizonte

January 2019

.

Gonzaga, André Lopes.

G642p A probabilistic algorithm to predict missing facts from knowledge graphs [manuscrito] / André Lopes Gonzaga. – 2019.
xx, 45 f. il.

Orientadora: Mirella Moura Moro.
Coorientador: Mário Sérgio Alvim.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.41-45
.
1. Computação – Teses. 2. Web semântica – Teses. 3. Aprendizado do computador – Teses. 4.Base de conhecimento – Teses. 5. Predição de falhas – Teses. I. Moro, Mirella Moura.II. Alvim,Mário Sérgio III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV.Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A Probabilistic Algorithm To Predict Missing Facts From Knowledge Graphs

ANDRÉ LOPES GONZAGA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

PROFA. MIRELLA MOURA MORO - Orientadora
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. MARIO SÉRGIO FERREIRA ALVIM JÚNIOR - Coorientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. DENILSON BARBOSA
Department of Computing Science - University of Alberta

Belo Horizonte, 31 de janeiro de 2019.

Dedico este trabalho a Fernanda, Leia e Letícia.

Acknowledgments

Quero agradecer primeiramente a minha esposa Fernanda, que esteve comigo durante toda a minha caminhada acadêmica e profissional. Desde 2008 ela embarcou nesta aventura comigo e por mais que tenhamos tido momentos duros nesta caminhada, acredito que estes momentos só vieram para nos fortalecer. Obrigado por compartilhar comigo todos estes momentos e por me dar o bem mais precioso do mundo, que é a Letícia.

Agradeço também aos meus pais, Rosângela e Luiz Alberto, por todo o apoio nos momentos de dificuldade. Vocês são os responsáveis por toda a minha educação e por tudo que conquistei até aqui. Seria impossível chegar aonde cheguei sem o apoio financeiro, psicológico e emocional de vocês.

Também incluo minhas 3 irmãs, Lílian, Isabella e Luísa, por todos os momentos que passei com vocês. Apesar do maior contato com a Lílian, ambas as 3 são igualmente especiais para mim. Obrigado por sempre estarem me apoiando e admirando, mesmo de longe. Lílian, nunca me esquecerei dos seus conselhos e broncas, pois foram eles que me impulsionaram para frente.

Essa caminhada estaria longe de ser cumprida se não fosse pela minha orientadora Mirella Moro, que desde a graduação me empurra para frente, cobrando ao nível extremo trabalhos de excelência. Não tenho palavras para agradecer o que sempre fez por mim, e por ter acreditado no meu pontecial desde o começo. Além de orientadora, foi minha amiga em todos estes momentos, sempre tendo uma palavra de cobrança mas também de esperança e conforto. Muito obrigado por tudo.

Igualmente digo sobre o meu co-orientador Mário Alvim que sem dúvida foi o melhor professor que tive na pós-graduação. Foi ele o responsável por tantas discussões e por me elevar a um outro patamar teórico e técnico. Essa dissertação não sairia se não fosse por nossa longas conversas e discussões. Tive o privilégio de te conhecer e espero levar nosso relacionamento para o resto da vida.

Agradeço também ao laboratório CS+X que, apesar do pouco contato diário (devido ao meu emprego integral), tive pessoas sensacionais que pude contar, citando especialmente a Laís Rocha, Natércia Batista e Michele Brandão. Vocês todos são especiais pra mim e agradeço a amizade e compreensão durante todo este período.

Por fim, a todos os meus familiares, tios, primos, incluindo as famílias Lourenço e Alves. Durante todo este tempo vocês foram muito importantes para o meu desenvolvimento. Também aos meus amigos de Avenue Code, RPG, NGNG e resenha. Divido igualmente todo este trabalho com vocês.

"Do or do not. There is no try."
(Master Yoda)

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Formalizing and passing the current knowledge to future generations has always been a concern of humankind. Thousands of years ago, the first *Homo sapiens* made cave paintings to preserve information. In addition to the paintings, and associated with the progress of speech, knowledge was passed by word of mouth for generations with stories and myths. However, an existing problem of passing knowledge through stories is that they are often lost, as different people have their way of interpreting and passing them on to the new generations.

With the advent of writing, human knowledge began to be saved on papyri and stones, and stories began to be recorded. However, it did not solve the problem of properly knowledge dissemination, because the materials commonly used were depreciated by the action of nature (e.g., stones were corroded and papyrus were ripped). Then, with the invention of paper, knowledge could be stored in a solid material that resists more to the actions of nature, when subject to proper care. Until the end of 20^{th} century, the most common way to keep knowledge was through books (paper). Numerous libraries around the world kept information and knowledge in billions of books of all kinds, from the arts to novels and poetry. Still in the 20^{th} century, another great revolution started and transformed information processing: the computer. We say that the digital age started from there and, with this huge advance, knowledge began to be digitally saved.

Then, the initiative of having a network of computers connected [5] transformed the world. Having hypertext links representing knowledge that is easily understandable by readers contributed to spread and democratize information. Indeed, the World Wide Web (WWW) represents one of the greatest technological and social advances of the last century. A plethora of data being exchanged between computer clients and servers as well as data retrieval systems becoming more robust represent just two benefits of the Web development. However, the increasing number of hyperlinks has also created different processing issues, which in turn required improving indexing and searching algorithms as well as enhancing results quality. Yet, one problem persisted: all these hypertext links are only readable by humans who can *read* the texts and *navigate* through the links.

In such a context, the Semantic Web [6] was proposed to create Web pages that not only humans can understand, but machines can get knowledge from. The Semantic Web is often considered as an extension (of the current WWW) that brings a common structure to the content of Web pages. Therefore, it provides such content with meaning that allows external software agents to carry out sophisticated tasks on behalf of the reader and promote a greater degree of cooperation between humans and computers. In other words, a new age of computing was ushered in which machines are able to "process and *understand* the data that they merely display at present" [17].

Technically, the Semantic Web has promoted a graph-based representation of knowledge in which entities are graph nodes (e.g., *Cecília Meireles* and *Espectros*) and there is a labelled edge between two nodes if they are related (e.g., Cecília *has written* Espectros). Entities may also have types denoted by *is a* relations (e.g., Cecília *is a* writer, Espectros *is a* book). These relations are known as *facts*. In many cases, the sets of possible types and relations are organized in a *schema* or an *ontology*, which defines their interrelations and their usage restrictions.

The Semantic Web also provides a common framework to share and reuse data across many applications such as servers that expose data system using Resource Description Framework (RDF), common metadata vocabularies and web-based services. Adding communicating multi-agent systems to the Web usually requires ontologies (an ontology encompasses a formal representation and definition of categories) to setup common vocabulary and rules. The integration of agent technology and ontologies may significantly affect the use of Web services and the ability to extend programs to perform tasks for users more efficiently and with less human intervention [17].

A graph representation of knowledge is known as *Knowledge Graph* (KG), as coined by Google in 2012[1]. In a broader perspective, any graph-based representation of knowledge could be considered a Knowledge Graph, which includes any kind of RDF for example. Note that RDF organizes knowledge in triples of the form *subject–predicate–object*, which are easily equivalent to node-edge-node in a KG. Well known knowledge graphs include: Freebase [9], DBpedia [2], Wikidata [45] and YAGO [42]. Also, search engines and e-commerce websites, such as Amazon and Walmart, are

[1]https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

trying to improve user experience by using KGs, by attempting to better understand how clients interact with the system and building a huge network connecting users and products. Such graphs are often constructed from semi-structured knowledge, such as Wikipedia, or harvested from the Web. Indeed, Knowledge Graphs on the Web form a backbone of many information systems that require access to structured knowledge. The idea of feeding intelligent systems and agents with formalized knowledge of the world dates back to classic Artificial Intelligence research in the 1980s [23, 29].

Since its theoretical definition in 2001, the Semantic Web has evolved, built upon existing technology and even boosted many advances for data processing over the Web. Specially, the advent of Linked Open Data (LOD) has spurred interest over representations of general world knowledge as graphs from completely fresh perspectives, for example [34]. Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, explicitly defined, linked to other external data sets, and can be linked to/from external data sets [8]. For example, Figure 1.1 shows a recent view of linked data. In practice, LOD is being used to predict facts in Knowledge Graphs since it uses the information presented in one KG to predict the same fact in other [26].

Nonetheless, constructing a knowledge graph presents several issues. Specifically, no matter its building procedure, the resultant graph is not expected to be perfect. The reason is simple: as a model of the real world, formalized knowledge cannot reach full coverage (i.e. cover all information about every entity in the universe). Also, achieving 100% of both completion and correctness is unlikely. Then, there is a consequent trade-off between coverage and correctness that is addressed differently in each knowledge graph, for example [10] and [44].

The solution for such a problem may borrow from other known issues. Specifically, *link prediction* is the task of predicting the existence (or probability of correctness) of edges in graphs [24]. In our context, link prediction is critical for KG not achieving 100% of completion and correctness. Even if KGs could reach a perfect representation of the world, such world changes over time, and then link prediction still remains important. Furthermore, link prediction can be easily mapped to not only adding missing data to KGs [30, 36], but also discovering links in the context of linked data [15], predicting new relationships over professional networks [11], and even applied to predicting drug-target and protein-protein interactions in biomedical graphs [13].

Contributions. Following such a link prediction reasoning, this work proposes a probabilistic algorithm (called ProA) that is based on actual existing paths in the graph to infer possible missing edges from an entity source to an entity target. Since

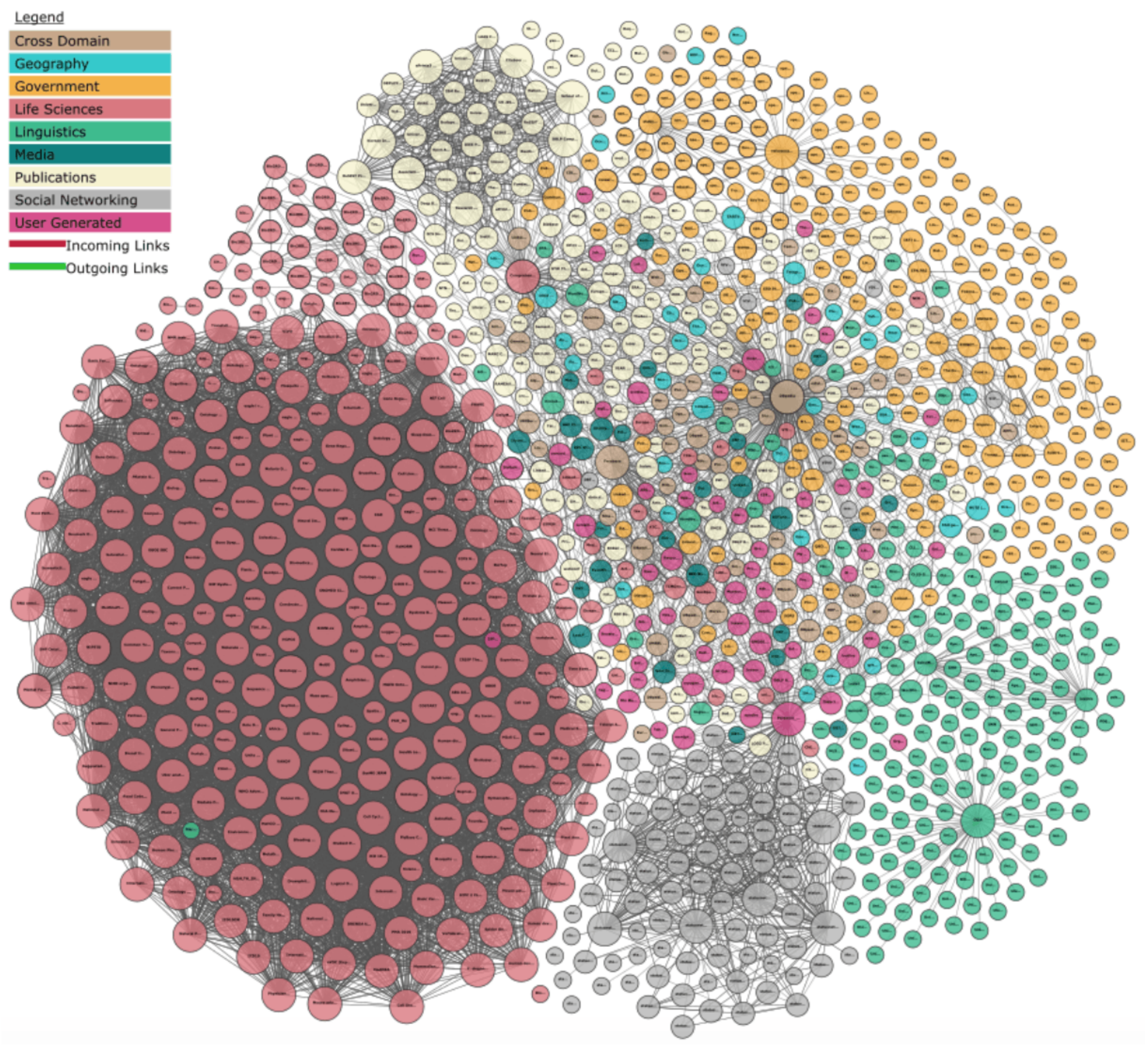


Figure 1.1. Linked Datasets in 2018 (source: https://lod-cloud.net/). Every node in this graph is a Knowledge Graph; edges are connections between them; and each color is a different domain (Knowledge Graphs could be focused on a specific domain such as Publications, Government and Life Sciences). Some KGs are clearly hubs and are essential to this shared network since they have a lot of information and connections.

this probabilistic approach uses only the distribution over paths between nodes in the graph, there are no entities properties or any other external dependencies to predict facts. Our evaluation also shows that using the probability distribution provides good performance when compared to other algorithms. Our advantages over the state-of-the-art are simplicity in terms of operation and computational complexity. As ProA takes advantage of the paths distribution of the KG, the actual KG structure is irrelevant and our solution could be applied to any KG.

Text Organization. This thesis is organized as follows. Chapter 2 presents an overview of related work, most of which is used as baseline. Chapter 3 presents our new probabilistic algorithm. Then, Chapter 4 presents our evaluation and results, whereas Chapter 5 concludes this work and provides future work.

Chapter 2

Related Work

This chapter starts with an overview of Machine Learning and Knowledge Graphs, and follows with related work on link prediction.

2.1 Probabilistic Reasoning

On trying to draw timeline on machine learning, the earliest events (most probably) happened around 1763 with the publication of the work that underlies the Bayes Theorem. Thomas Bayes's work *An Essay Towards Solving a Problem in the Doctrine Of Chances* [4] was published two years after his death, having been amended and edited by a friend of Bayes, Richard Price. Later, in 1810, Pierre-Simon Laplace published *Théorie Analytique des Probabilités* [22], in which he expands upon the work of Bayes and defines what is now known as Bayes Theorem.

Bayesian inference is one of the many applications of Bayes Theorem, a particular approach to statistical inference. The probabilities involved in Bayes Theorem may have different interpretations. With the Bayesian probability interpretation, the Theorem expresses how a subjective degree of belief should rationally change to account for availability of related evidence. Then, Bayesian inference is fundamental to Bayesian statistics.

Formally, Bayes Theorem is stated as Equation 2.1.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{2.1}$$

where A and B are events and $P(B) \neq 0$. Then, $P(A)$ and $P(B)$ are the probabilities of observing A and B independently of each other. Also, $P(B|A)$ and $P(A|B)$ are conditionals probability: the likelihood of an event occur given the other is true.

Later, in 1913, Andrey Markov first described a set of techniques that later become known as Markov chains. A Markov process, named after the Russian mathematician, is a stochastic process that satisfies the Markov property. Roughly speaking, a process satisfies the Markov property if it can predict its future based only on its present state; hence, it does so independently from its history.

The Markov process is important because it defines the learning of an event by considering only the current state of the problem, i.e., disregarding its past. It is formally presented in Equation 2.2.

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n), \tag{2.2}$$

which means if the probability of an event X_{n+1} is equal x given all past events, it is equal only with the present event X_n. Directed graphs often describe a Markov chain, where the edges of graph n are labeled by the probabilities of going from one state at time n to the other states at time $n+1$, $P(X_{n+1} = x | X_n = x_n)$.

Although machine learning algorithms have existed for a long time, the ability to automatically apply complex mathematical calculations to big data is a recent development. Renewed interest in machine learning is due to the same factors that have made data mining and Bayesian analysis even more popular. For example, the usage of latent features[1] is increasing each year (as discussed in Section 2.4), but this technique should be studied carefully.

Quinn et al. [40] report a strong association between myopia in children and their exposure to night-time lighting during their first two years. This work was published on Nature and associated an occurrence of light during night and myopia in children. In other words, the variables *myopia* and *children* are most likely to be correlated if *night-time lighting* is present. Figure 2.1(a) shows how the variables are correlated in such a study.

Then, Gwiazda et al. [16] were unable to confirm such a surprising result, but they found that myopic parents are more likely to employ night-time lighting aids for their children. Moreover, there is an association between myopia in parents and their children. Hence, the hidden variable *parents-myopia* was not used in the first study leading a wrong inference and a consequent wrong result. Figure 2.1(b) shows the new hidden variable impacting the variable *myopia* in children. This later work is a successfully example of how to use latent features. We further discuss how it is being used on Knowledge Graphs in Section 2.4.

Machine Learning algorithms can generally be divided in three main categories.

[1]Not directly observable.

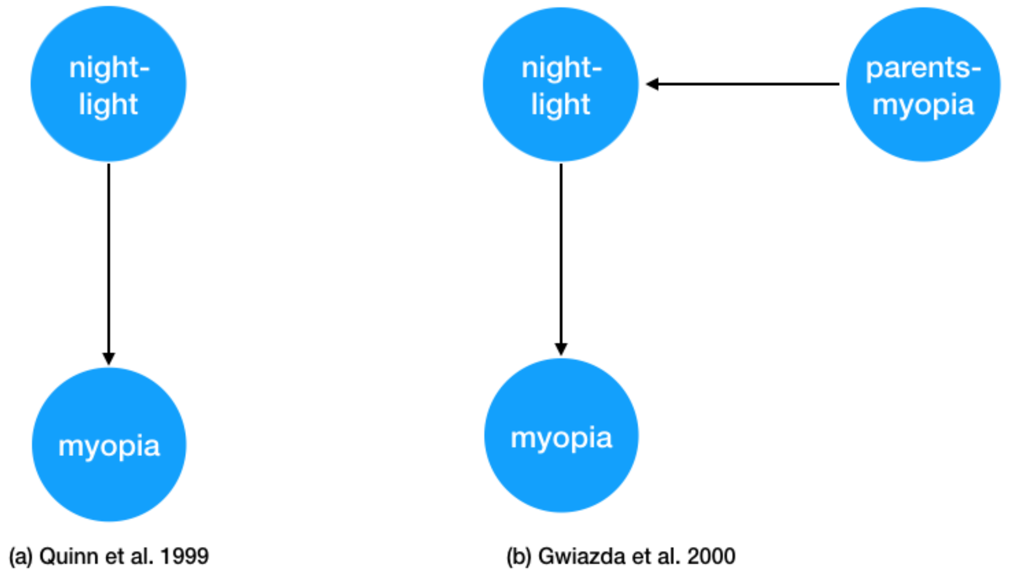


Figure 2.1. Example of inappropriate usage of latent features. In (a), night-lighting and myopia are being correlated since children exposed by night-light are more likely to have myopia. However in (b) a new hidden variable is added about the parents myopia. Parents that have myopia uses night-light to take care of their children so the real cause of children myopia is not the night-light but the parents myopia themselves.

The first one is the *supervised learning*, which takes an input variable x and and returns and output variables Y with the goal of approximating the mapping function from x to Y, such that Y can be well predicted from x. The *supervised* part is about the process of an algorithm to learn from the training dataset. As the correct answers from the training dataset are previously known, the algorithm can make predictions, check their correctness, and the learning process stops when the algorithm achieves an acceptable level of performance. For example, regression and classification problems are classical supervised learning problems.

The second one is *unsupervised learning*, which takes just the input data x and no corresponding output variables. Its goal is to model a structure or distribution in the data in order to learn more about it. The *unsupervised* part is because it does not have a correct answer and there is no way to check its output. In other words, such algorithms are responsible for discovering and presenting the results. For example, clustering and association problems are common uses of unsupervised learning methods. A mix of the first and second methods is the *semi-supervised learning*, which takes a large amount of input data x and only some of is labeled Y. These problems have both supervised and unsupervised learning features, and many real work machine learning problems fall into such category.

A third category is *reinforcement learning* which differs from supervised learning in that correct input/output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. The focus is not on performance but finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

2.2 Knowledge Graphs

The data science research community has been working hard to improve the information quality in Knowledge Graphs. There are two main goals of such KGs refinement: increase completeness [21] [33] [37] [38]; and improve correctness [14] [25] [39]. A recent survey [36] discusses approaches and evaluation methods related to such two goals. It distinguishes refinement methods along different dimensions, mainly focused in completion versus correction. In addition, both completion and correction approaches can be distinguished by the targeted kind of information. Some approaches target completing/correcting information about entities, while others target on relations information between entities.

There is also a third way to distinguish refinement methods in Knowledge Graphs that relates to the data: using the KG data itself, called *internal approach*; and using external data such as text *corpora* or other KGs data. Using an internal approach is often more difficult, because to predict or discover an incorrect/missing fact, it must perform complex mathematical calculations over the current state of Knowledge Graph.

2.2.1 Representation

A graph $G = (V, E)$ is a set of vertices V and edges E. Knowledge graphs model information in the form of entities as vertices and relationships between them as edges. Recently, KGs have appeared in the Semantic Web community with the purpose of creating a "web of data" that is readable by machines. This vision of the Semantic Web remains to be fully achieved, as some parts are still under construction.

Here, we use the RDF standard to represent facts in the form of binary relationships: $[subject, predicate, object]$ triples (or simply SPO triples), in which $subject$ and $object$ are entities, and $predicate$ is the relation between them. The existence of a particular SPO triple indicates an existing fact, i.e., the respective entities are in a relationship of the given type, as the examples in Figure 2.2. The figure shows known facts such as "Barack Obama is married to Michelle Obama", which is translated to the triple $[BarackObama, isMarriedTo, MichelleObama]$ and to two nodes

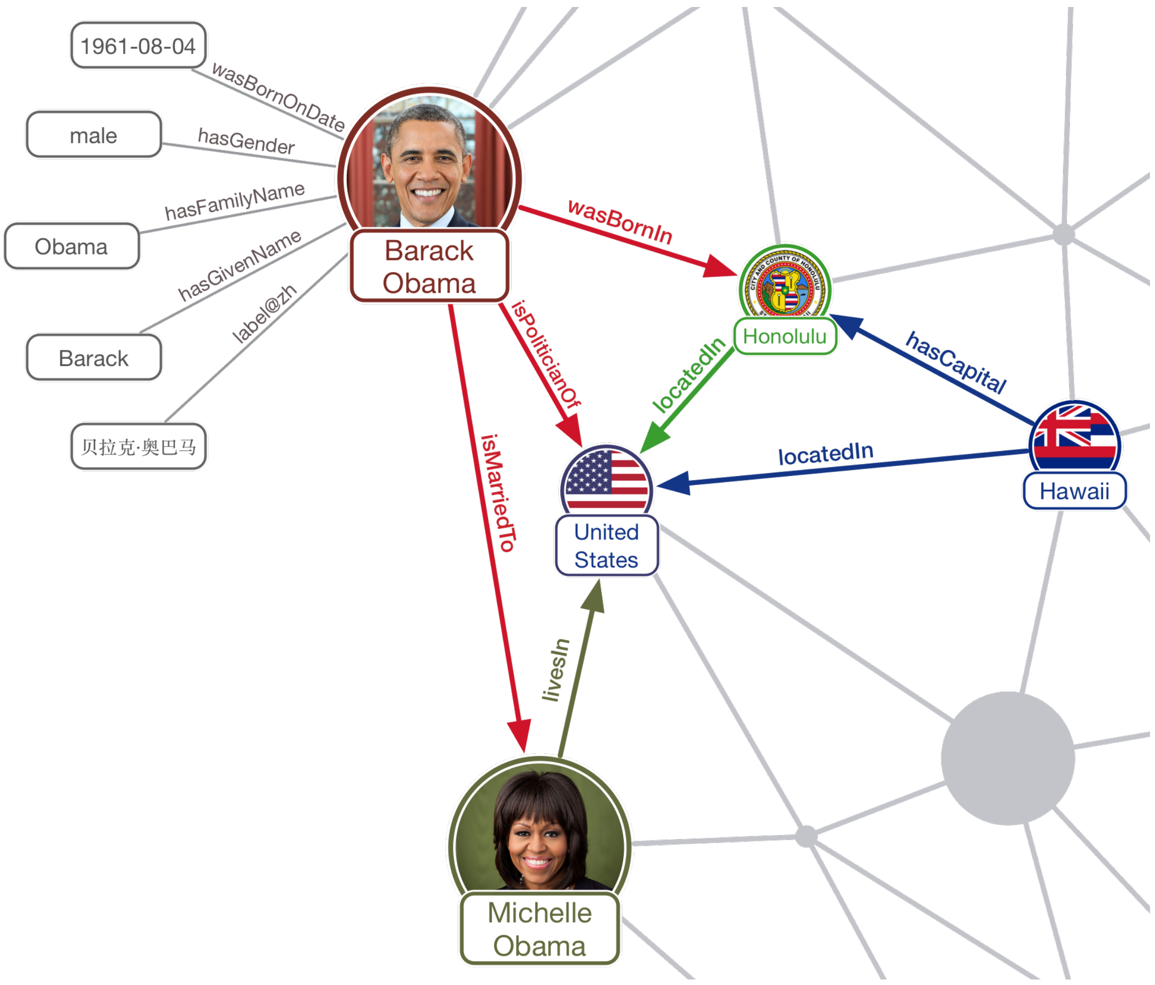


Figure 2.2. Sample of Knowledge Graph where entities are nodes and the relationships between them are the edges.

($BarackObama$, $MichelleObama$) and an edge ($isMarriedTo$) into the graph. The same applies to "Barack Obama was born in Honolulu", "Honolulu is capital of Hawaii", and so on.

Having all facts translated, their respective SPO triples form a multigraph[2], where nodes represent entities (all subjects and objects), and directed edges represent relationships. Different relations are represented via different types of edges (also called *edge labels*). The direction of an edge indicates whether entities occur as subjects or objects, i.e., an edge points from the subject to the object.

[2]A graph that may have multiple edges; that is, edges that have the same end nodes.

2.2.2 World Assumption

While existing triples always encode known true relationships (facts), there are different paradigms for the interpretation of non-existing triples:

- CLOSED WORLD ASSUMPTION: a non-existing triple indicates false relationship. For example, in Figure 2.2, there is no *wasBornIn* edge from *Michelle Obama* to *Honolulu*, which is interpreted as she was definitely not born there.

- OPEN WORLD ASSUMPTION: a non-existing triple is interpreted as unknown, and the corresponding relationship can be either true or false. Still in Figure 2.2, the missing edge $[MichelleObama, wasBornIn, Honolulu]$ is not interpreted as she was not born there; but rather, as the fact is unknown. This more cautious approach is justified, since KGs are known to be very incomplete.

Usually RDF and the Semantic Web use the open-world assumption, and we follow such trend and use the open-world assumption as well.

2.2.3 Construction

Completeness, accuracy, and data quality are important parameters that determine the usefulness of Knowledge Graphs and are influenced by the way they are constructed. KG construction methods are usually classified into four main groups:

- CURATED: triples are manually created by a closed group of experts.

- COLLABORATIVE: triples are manually created by an open group of volunteers.

- AUTOMATED SEMI-STRUCTURED: triples are automatically extracted from semi-structured text[3] via hand-crafted rules, learned by Artificial Intelligence rules, or regular expressions.

- AUTOMATED UNSTRUCTURED: triples are automatically extracted from unstructured text via machine learning and natural language processing techniques.

Construction of curated Knowledge Graphs typically leads to highly accurate results, but this technique does not scale well due to its dependence on human experts. Collaborative KG construction, which was used to build Wikipedia and Freebase, scales better but still has some limitations. For instance, the attribute *place of birth* is missing for 71% of all people included in Freebase, even though this is a mandatory property

[3]Texts in table format such as infoboxes in Wikipedia.

of the schema [46]. Also, a study [43] found that the growth of Wikipedia has been slowing down. Consequently, automatic Knowledge Graph construction methods have been gaining more attention.

Such methods can be grouped into two main approaches. The first approach exploits semi-structured data, such as Wikipedia infoboxes, which has led to large, highly accurate knowledge graphs such as YAGO [42] and DBpedia [2]. The accuracy[4] of facts in such automatically created KGs is often still very high. For instance, the accuracy of YAGO was estimated to be over 95% through manual inspection of sample facts [7], and the accuracy of Freebase [9] was estimated to be 99%. However, semi-structured text still covers only a small fraction of the information stored on the Web, and completeness (or coverage)[5] is another important aspect of KGs. Hence the second approach tries to "read the Web" by extracting facts from the natural language texts of Web pages. Examples include projects as NELL [50] and the Knowledge Vault [14].

2.3 Link Prediction

The link prediction task is: given an incomplete graph, find existing edges to increase the completeness In the context of regular graphs and complex networks, *Link prediction* is an important task because graphs are often incomplete or need improvement. For example, a social network developer may want to predict relationships between users or study the possible evolution of a professional network [11]. Indeed, link prediction is critical due to KGs not achieving 100% of completion and correctness. Even if KGs could reach a perfect representation of the world, such world changes over time, and then link prediction still remains important. Now, there are many works improving link prediction over Knowledge Graphs. This section follows with a comprehensive overview of relevant approaches that are used as our baselines (in our experimental evaluations).

Angeli et al [1] propose a system to complete KGs with a degree of confidence for each unseen fact. As facts in these bases are often devoid of context, it defines the notion of *truth* to reflect whether it would assume a fact to be true without evidence to the contrary. Then, it can further refine the task by determining whether an arbitrary fact is plausible (true in the absence of contradictory evidence). The approach generalizes word similarity metrics to a notion of *fact similarity* and judges unseen facts based on the aggregate similarity between them and existing members of the database. For

[4]The trustworthiness of the present facts.

[5]Completeness is about facts that should be in Knowledge Graph but they are not. Correctness is about facts that are in Knowledge Graphs but they are there wrongly and then should be removed.

instance, if *philosophers are mortal* but it is known that Greeks are philosophers, and that philosophers and Greeks are similar, it is reasonable to infer that *philosophers are mortal* is nonetheless plausible.

Databases were originally developed to support deterministic applications such as banking, payroll, accounting, inventory; all of which require a *precise* semantics of the data. However, there is also imprecise and uncertain data that contain an explicit representation of uncertainty. *Degrees of belief* in knowledge bases (KB) was described for the first time in 1996 [3], which also included important aspects of how to deal with uncertainty. Its authors propose the *random-worlds* method that relies on the principle of indifference: it treats all of the worlds the agent considers possible as being equally likely. The approach is able to integrate qualitative default reasoning with qualitative probabilistic reasoning then providing a language in which both types of information can be easily expressed. It also describes the uncertainty in knowledge bases in terms of *degrees of belief*. The most widely used framework for assigning degrees of belief to each relation in a knowledge base, which are essentially subjective probabilities, is the Bayesian paradigm. It assumes a space of possibilities and a probability distribution over this space, called the *prior distribution*, and calculates posterior probabilities by conditioning on what is already known.

Using such an approach requires to specify the space of possibilities and the distribution over it. Indeed, the usual practice is considering such decisions as subjective. Then, the approach in [3] assumes that the KB contains all the knowledge the agent has, and it allows a very expressive language so as to make this assumption reasonable. In other words, any knowledge the agent has (that could influence the prior distribution) is already included in the KB. As a consequence, it gives a single uniform construction of a space of possibilities and a distribution over it. Once it has this probability space, it can use the Bayesian approach to compute the probability of an assertion φ given a KB, and then compute the probability of φ using the resulting posterior distribution.

Still in the relational scenario, the Path Ranking Algorithm (PRA) [21] extends the idea of using random walks of bounded lengths for predicting links in multi-relational Knowledge Graphs. In particular, let $\pi_L(i, j, k, t)$ denote a path of length L, where t represents the sequence of edge types $t = (r_1, r_2, \ldots, r_L)$. PRA also requires a direct arc representing the existence of a relationship of type k from e_i to e_j. Then, PRA can compute the probability of following such a path by assuming that at each step, it follows an outgoing link uniformly at random. A probability of this particular path can be computed recursively by a sampling procedure, similar to PageRank [35]. The key idea in PRA is to use these path probabilities as features for predicting the probability of missing edges.

In a different perspective, Chekol et al [12] describe how to combine uncertainty and time in Knowledge Graphs. Since facts are usually accompanied by a confidence score that witnesses how likely it is for them to hold, the authors show a solution for the management of uncertain and temporal data in KGs. The work relies on one main step: using Markov Logic Network (MLN) that provides the necessary underpinning to formalize the syntax and semantics of uncertain temporal KGs. Then, they explore the usage of Probabilistic Soft Logics (PSL) and compare the results against the MLN approach.

Xu and Barbosa [47] report an effectiveness evaluation of the existing knowledge base embedding models for link prediction. It describes a new benchmark, which is much larger and complex than previous ones and helps validate the effectiveness. The results show knowledge base embedding models are generally effective for link prediction but unable to give improvements for the state-of-art neural relation extraction model with the existing strategies, while pointing limitations of existing methods.

Information theory has been taken as a prospective tool for quantifying the complexity of complex networks, including Knowledge Graphs. Xu et al [48] started to study the information entropy or uncertainty of a path using information theory. It applies the path entropy to the link prediction problem in real-world networks. Specifically, the authors propose a new similarity index, namely Path Entropy (PE) index, which considers the information entropy of shortest paths between node pairs with penalization to long paths. Empirical experiments demonstrate that PE index outperforms the mainstream link predictors.

2.4 Specific Techniques

As already discussed, the presence or absence of certain triples in relational data is correlated with (i.e., predictive of) the presence or absence of certain other triples. We model each possible triple x_{ijk} over the set of entities and relations as a binary random variable $Y(ijk) \in 0, 1$ that indicates its existence, and each random variable is correlated with each other. There are three main ways to model these correlations, as follows.

- Latent feature models: Assume all $Y(ijk)$ are conditionally independent; given latent features associated with subject, object and relation type and additional parameters.

- Graph feature models: Assume all $Y(ijk)$ are conditionally independent; given observed graph features and additional parameters.

- MARKOV RANDOM FIELDS: Assume all $Y(ijk)$ have local interactions.

The Latent and Graph feature models predict the existence of a triple x_{ijk} via a score function $f(x_{ijk};\theta)$, which represents the model's confidence that a triple exists given the parameters θ. Assuming N_e and N_r as the total number of entities and relations respectively, O as a set of observed triples, the conditional independence assumptions of those models allow the probability model to be written as Equation 2.3.

$$P(Y|O,\theta) = \prod_{i=1}^{N_e}\prod_{j=1}^{N_e}\prod_{k=1}^{N_r} Ber(Y(ijk)|\sigma(f(x_{ijk};\theta))), \tag{2.3}$$

where $\sigma(u) = 1(1+e^{-u})$ is the sigmoid (logistic) function, and

$$Ber(y|p) = \begin{cases} p, & \text{if y=1} \\ 1-p, & \text{if y=0.} \end{cases} \tag{2.4}$$

We discuss various possible forms for the score function $f(x_{ijk};\theta)$ ahead.

What all models have in common is that they explain triples via latent features of entities (which is justified via various theoretical arguments). For instance, a possible explanation for the fact "Morgan Freeman received an Academy Award" is that he is a good actor. This explanation uses latent features of entities (being a good actor) to explain observable facts (Freeman receiving an Academy Award). We call these features *latent* because they are not directly observed in the data. One task of all latent feature models is therefore to infer these features automatically from the data.

Another way of modeling relations is using *tensors*. Tensors are mathematical objects that can be used to describe physical properties, just like scalars and vectors. In other words, tensors are a generalisation of scalars and vectors; a scalar is a zero rank tensor, and a vector is a first rank tensor. The rank (or order) of a tensor is defined by the number of directions (and hence the dimensionality of the array) required to describe it. For example, properties that require one direction (first rank) can be fully described by a 3×1 column vector, and properties that require two directions (second rank tensors) can be described by nine numbers as a 3×3 matrix. As such, in general an n^{th} rank tensor can be described by 3^n coefficients.

To represent relational data, we use the semantic web's RDF formalism where relations are modeled as triples of the form $[subject, predicate, object]$, and a predicate either models the relationship between two entities or between an entity and an attribute value. In order to model dyadic relational data as a tensor, some algorithms

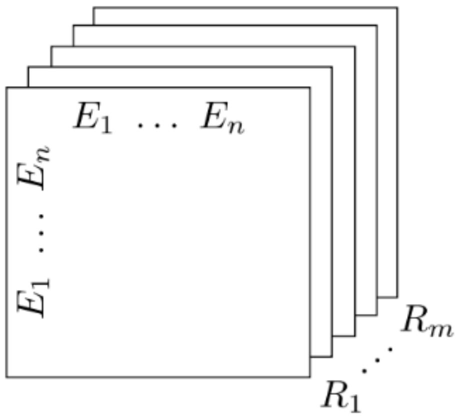


Figure 2.3. Tensor model for relational data. $E_1 \ldots E_n$ denote the entities, while $R_1 \ldots R_m$ denotes the relations in the domain.

employ a three-way tensor X, where two modes are identically formed by the concatenated entities of the domain and the third mode holds the relations. Figure 2.3 provides an illustration of this modelling method. A tensor entry $X_{ijk} = 1$ denotes the existence of a relation $[i^{th}entity, k^{th}predicate, j^{th}entity]$. Otherwise, for non-existing and unknown relations, the entry is set to zero.

RESCAL [32] is a relational latent-feature model that explains triples via pairwise interactions of latent features. In particular, it models the score of a triple x_{ijk} and a weight matrix $\mathbf{W}$ whose entries w_{abk} specify how much the latent features a and b interact in the k-th relation. This is a bilinear model, since it captures the interactions between two entity vectors using multiplicative terms. In general, it models block structure patterns via the magnitude of entries in matrix $\mathbf{W}$, while it uses homophily patterns via the magnitude of its diagonal entries.

Specifically, it employs a rank-r factorization, where each slice X_k is factorized by Equation 2.5.

$$X_k = AR_kA^T, for\ k = 1, \ldots, m. \tag{2.5}$$

For a short illustration of this mechanism, consider the example in Figure 2.4. The latent-component representations of Al and Lyndon will be similar to each other in this example, as both representations reflect that their corresponding entities are related to the object Party X. Because of this, Bill and John will also have similar latent-component representations. Consequently, the product $a^T_{Bill}R_{party}a_{PartyX}$ will yield a similar value to $a^T_{John}R_{party}a_{PartyX}$ and, as such, the missing relation can be predicted correctly. Please note that this information propagation mechanism through

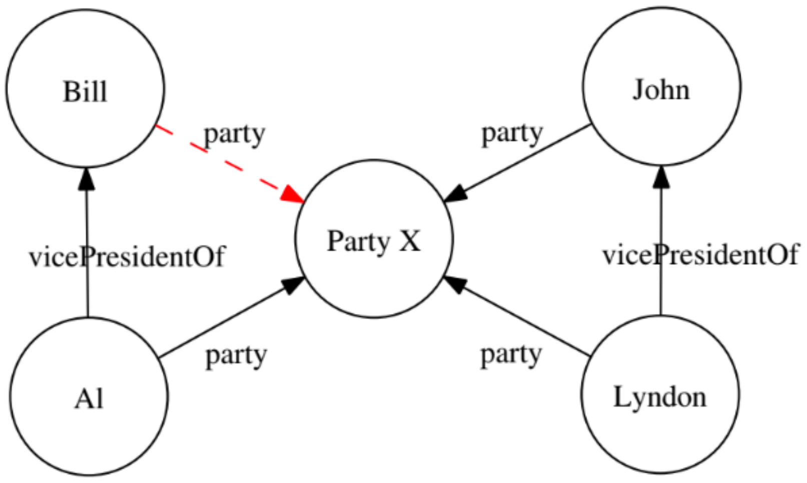


Figure 2.4. Visualization of a subgraph of the relational graph for the US presidents example. The relation represented by a dashed-red arrow is unknown.

the latent components would break if Bill and John had different representations as subjects and objects.

DistMult [49] is a special form of a bilinear model like RESCAL, where the non-diagonal entries in the relation matrices are assumed to be zero. In this model, each entity e_i is assigned a latent feature vector (embedding) n_i of dimensionality K; and each relation type is assigned an embedding r of the same dimensionality. The score of a candidate triple (e_s, r, e_o) is defined as: $f(x_s, r, o) = r^T(n_s x n_o)$.

Some models are based on multi-layer perceptrons (MLPs), also known as feed-forward neural networks. In the context of multidimensional data, they can be referred to multiway neural networks. This approach allows to consider alternative ways to create composite triple representations and to use nonlinear functions to predict their existence. For example, ER-MLP [27] uses a global weight vector for all relations based on multi-layer perceptrons (MLPs). Figure 2.5 shows its comparison against RESCAL as a Neural Network.

Another class of models are latent distance models (also known as latent space models in social network analysis), which derive the probability of relationships from the distance between latent representations of entities: entities are likely to be in a relationship if their latent representations are close according to some distance measure. TransE [10] translates the latent feature representations via a relation-specific offset instead of transforming them via matrix multiplications. In particular, the score of a triple x_{ijk} is defined by Equation 2.6.

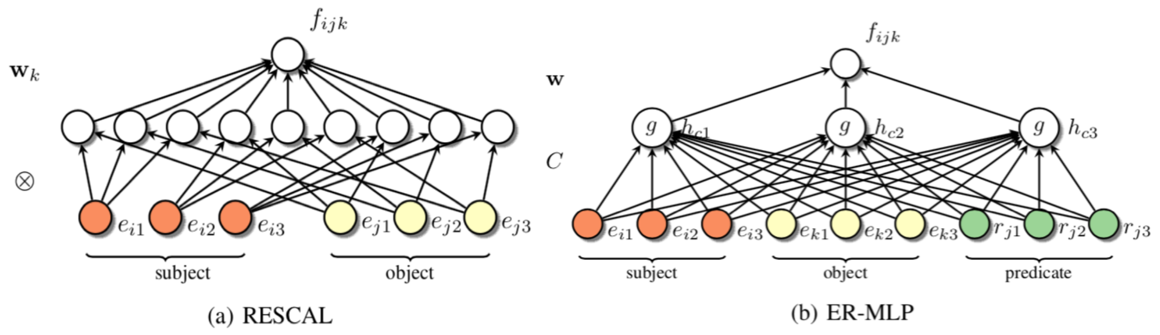


Figure 2.5. Visualization of RESCAL and ER-MLP model as Neural Network. Here, $H_e = H_r = 3$ and $H_a = 3$. Note, that the inputs are latent features. The symbol g denotes the application of the function $g(\cdot)$.

$$f_{ijk}^{TransE} = -d(e_i + r_k, e_j), \tag{2.6}$$

where $d(\cdot,\cdot)$ refers to an arbitrary distance measure such as the Euclidean distance.

HolE [31] learns compositional vector space representations of an entire Knowledge Graph. This method is related to holographic models of associative memory, as it employs circular correlation to create compositional representations. Compositional vector space models provide an elegant way to learn the characteristic functions of the relations in a Knowledge Graph, as they allow to cast the learning task as a problem of supervised representation learning. HolE introduces models of the form given by Equation 2.7.

$$Pr(\phi_p(s,o) = 1|\Theta) = \sigma(\eta_{spo}), \tag{2.7}$$

where $r_p \in R^{d_r}$, $e_i \in R^{d_e}$ are vector representations of relations and entities; $\sigma(x) = 1/(1 + exp(-x))$ denotes the logistic function; $\Theta = \{e_i\}_{i=1}^{n_e} \cup \{r_k\}_{k=1}^{n_r}$ denotes the set of all embeddings. Figure 2.6 shows its comparison against RESCAL as Neural Networks.

Complex [44] performs sparse tensor factorization of KG in the complex domain. Specifically, nodes and relations are modeled by d dimensional vectors with a real and an imaginary part ($Re(x)$, $Im(x)$). This allows to model anti-symmetric relations since the three way dot product (inner product) in the complex domain is not symmetric.

Proposed in 1927, Canonical Polyadic (CP) [18] decomposition is among the first tensor factorization approaches. CP generally performs poorly for link prediction as it learns two independent embedding vectors for each entity, whereas they are really tied. SimplE is a simple enhancement of CP that allows the two embeddings of each entity

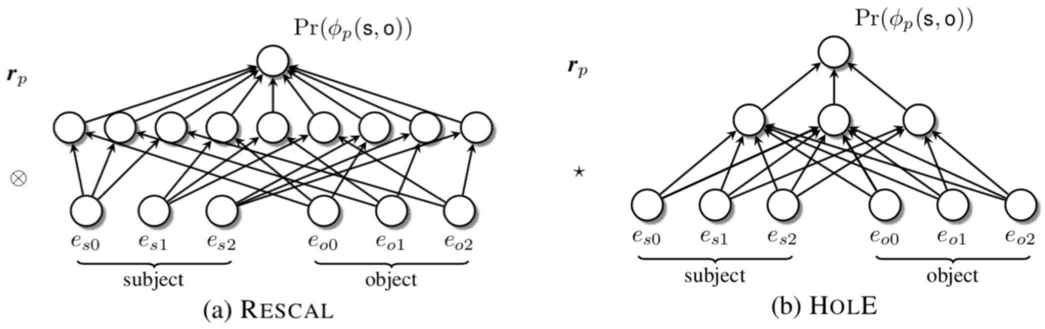

$\Pr(\phi_p(\mathsf{s},\mathsf{o}))$
$\boldsymbol{r}_p$
$\otimes$
e_{s0} e_{s1} e_{s2} e_{o0} e_{o1} e_{o2}
subject
object
(a) RESCAL
$\Pr(\phi_p(\mathsf{s},\mathsf{o}))$
$\boldsymbol{r}_p$
$\star$
e_{s0} e_{s1} e_{s2} e_{o0} e_{o1} e_{o2}
subject
object
(b) HOLE

Figure 2.6. RESCAL and HOLE as neural networks. RESCAL represents pairs of entities via d^2 components (middle layer). In contrast, HOLE requires only d components.

to be learned dependently. The complexity of SimplE grows linearly with the size of embeddings. The embeddings learned through SimplE are interpretable, and certain types of background knowledge can be incorporated into these embeddings through weight tying [19].

For example, let $likes(p, m)$ represent if a person p likes a movie m and $acted(m, a)$ represent who acted in which movie. Then, which actors play in a movie is expected to affect who likes the movie. In CP, observations about *likes* only update the t vector of *movies*, and observations about *acted* only update the h vector. Therefore, what is being learned about movies through observations about acted does not affect the predictions about likes, and vice versa. SimplE takes advantage of the inverse of relations to address the independence of the two vectors for each entity in CP. While inverse of relations has been used for different purposes, using them to address the independence of the entity vectors in CP is a novel contribution.

2.5 Final Considerations

Here, we propose a new way to predict edges in Knowledge Graphs based on path information from the topological structure of the graph. Given the probability distribution over all paths between two entities and the probability distribution of those paths in the whole graph, the probabilistic algorithm learns about it and generates a new probability distribution to infer if there is an edge to be predicted or not. The main difference between our approach to existing ones is the input. ProA only takes the path information over entities to predict facts, without any complex mathematical calculation. Moreover, the results show competitive performance and accuracy against current state-of-the-art, specially for dense graphs.

Chapter 3

Predict Missing Facts

We now introduce concepts necessary to understand our proposed solution, followed by our algorithm called *ProA*.

3.1 Basic Concepts

Knowledge Graphs typically have statistical patterns or regularities, which are not universally true but nevertheless allow for useful predictive power. They also have some deterministic rules, such as type constraints and transitivity (e.g., if Barack Obama was born in Honolulu, and Honolulu is located in the USA, then we can infer that Barack Obama was born in the USA). One example of such statistical pattern is homophily, that is, the tendency of entities to be related to other entities with similar characteristics. This has been widely observed in various social networks. For example, Brazil born actors are more likely to star in Brazil made movies.

Graphs can also exhibit global and long-range statistical dependencies, i.e., dependencies that can span over chains of triples and involve different types of relations. For example, the citizenship of Barack Obama (USA) depends statistically on the city where he was born (Honolulu), and such dependency involves a path over multiple entities (`Barack Obama`, `Honolulu`, `USA`) and relations (`bornIn`, `locatedIn`, `citizenOf`). A distinctive feature of relational learning is its ability to exploit such patterns for creating richer and more accurate models of relational domains.

Formally, let N_e be the number of entities, N_r be the number of relations, $\xi = \{e_1, e_2, ..., e_{N_e}\}$ be the set of all entities, and $\Gamma = \{r_1, r_2, ..., r_{N_r}\}$ be the set of all relation types in a Knowledge Graph. We model each possible triple $x_{ijk} = (e_i, r_k, e_j)$, indicating that entity e_i is related to entity e_j by relation type r_k, over these sets of entities and relations as a binary random variable $Y(ijk) \in \{0, 1\}$, which indicates its existence.

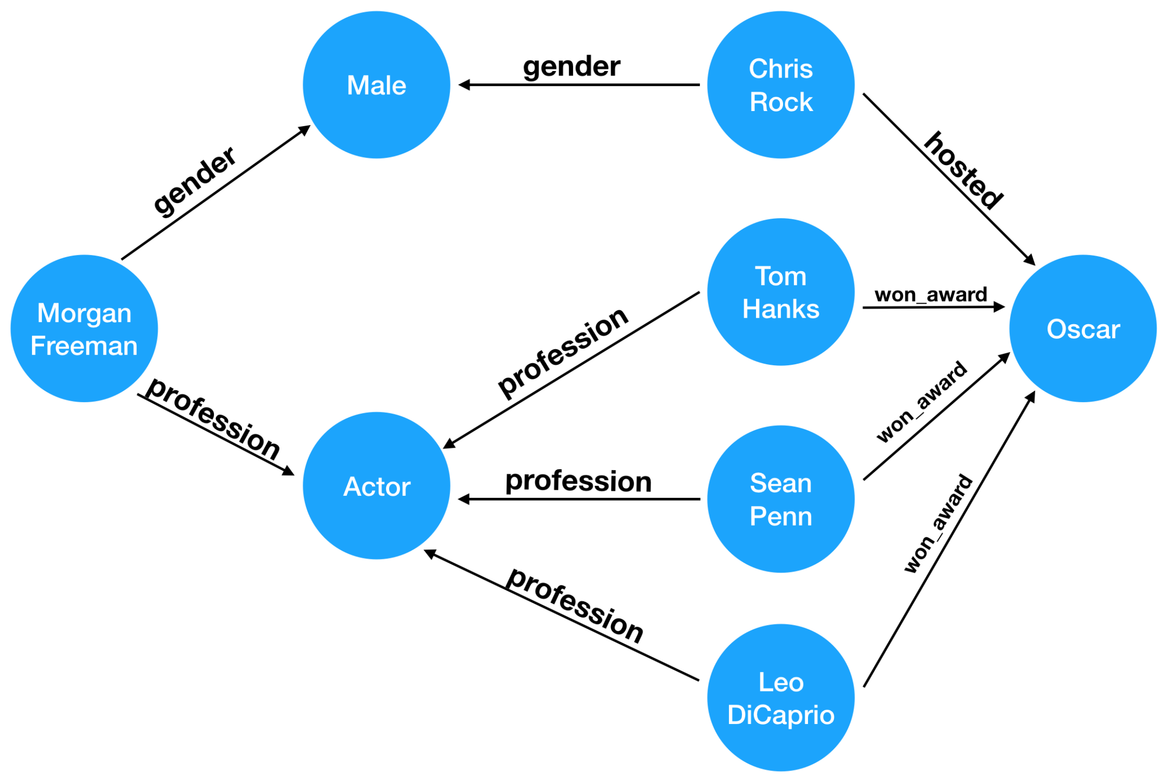


Figure 3.1. Probability dependencies between entities. There are two ways to reach `Oscar` from `Morgan Freeman`: through [`gender`, `gender`$^{-1}$, `hosted`], and through [`profession`, `profession`$^{-1}$, `won_award`].

While $Y(ijk) = 1$ indicates the existence of a triple, the interpretation of $Y(ijk) = 0$ depends on whether the open world or closed world assumption is considered. In this work, we assume it is open world, which means if a random variable is $Y(ijk) = 0$ then this fact is unknown.

A path D is a sequence of relations, in which the last entity of a relation is the first entity of the following one, with length greater than 0, connecting two or more entities. For example, a valid path could be: $[(BeloHorizonte, MinasGerais, capitalOf)$ $(MinasGerais, Brazil, stateOf)]$, which means the city `Belo Horizonte` is the capital of `Minas Gerais` and `Minas Gerais` is a state of `Brazil`. Then, there is a valid path between `Belo Horizonte` and `Brazil`. Moreover, random variables $Y(ijk)$ are conditionally dependent on each other because we consider paths and their dependencies. In other words, if the probability to reach a node x is $P(x)$ and the node y depends on x, then the probability of reaching y is $P(y|x)$.

Figure 3.1 illustrates an example of a Knowledge Graph. The graph represents real facts regarding `profession`, `gender`, `hosted_oscar` and `won_oscar` relations. Then, the probability of reaching the target node `Oscar` from source node `Morgan`

Table 3.1. Content of set D (set of paths at length at most 3) starting at Morgan Freeman from Figure 3.1.

-	$\ell = 1$	$\ell = 2$	$\ell = 3$
Morgan Freeman	Male	-	-
Morgan Freeman	Actor	-	-
Morgan Freeman	Male	Chris Rock	-
Morgan Freeman	Actor	Tom Hanks	-
Morgan Freeman	Actor	Sean Penn	-
Morgan Freeman	Actor	Leo DiCaprio	-
Morgan Freeman	Male	Chris Rock	Oscar
Morgan Freeman	Actor	Tom Hanks	Oscar
Morgan Freeman	Actor	Sean Penn	Oscar
Morgan Freeman	Actor	Leo DiCaprio	Oscar

`Freeman` depends on the probability of each relation in the path. Since this graph has four paths connecting `Morgan Freeman` to `Oscar`, the question here is: "Can we infer the existence of any direct relation between `Morgan Freeman` and `Oscar`"?

Let D be the set of paths of length at most ℓ between any two nodes. Table 3.1 shows the content of D considering the node `Morgan Freeman` with ℓ from 1 to 3. We only consider paths in which the random variable Y[1] is 1, i.e., there is a path between two nodes. There is also a probability distribution over D, because some paths can be more likely than others. There is a probability distribution for every valid path in D in the whole topological structure in the graph which means how often the path is in the graph structure. Note it is a valid probability distribution since the sum of values is 1.

In the current example, there are two possible paths from node `Morgan Freeman` to `Oscar`: $[(gender, gender^{-1}, hosted)]$ and $[(profession, profession^{-1}, won\_award)]$, and they occur one and three times, respectively. Then, since the probability of each one is 25%, we can infer that if there is an edge connecting these entities, the probability to be `won_award` is 75% and `hosted` is 25%, given the distribution of the paths.

Also, in such example, there are no paths of size other than three; however, in real examples (discussed in the next section), there are normally paths of different sizes, and the distribution table contains more values as well. It is also important to note that each path in the graph contains the same weight; that is, the probability of each of them is uniform. Some papers [48] put different weights for different path

[1] $Y(ijk) = 1$ when the nodes i and j are connected through relation k.

sizes in a graph, which is not the case in this work since we consider every path in set D with the same probability. Usually, the weights are defined according to the length of paths. For example, the larger the path size, the smaller the weight assigned to it. For Knowledge Graphs, the weight is the same, since the edges have semantics, and a longer path between two entities should not necessarily be penalized.

In summary, the intuition here is: popular paths can predict relations between two entities. If a certain path occurs many times between a set of similar nodes, it can denote that there is a great probability to have the same edge between those similar nodes. For example, considering again Figure 3.1, the nodes `Tom Hanks`, `Sean Penn`, `Leo DiCaprio` and `Chris Rock` share the same relation types as `Morgan Freeman`. Nonetheless, its more likely that `Morgan Freeman` has won an Oscar than hosted one, because there are more similar nodes that won the award than hosted it.

Regarding the distribution math, a distribution over all edges in the graph can be defined by counting the number of each type of relation divided by the total of edges in the KG. For instance, consider the Freebase train dataset with 483,142 facts. One of its popular edge is `award_nominee`, as it appears almost 16,000 representing 0.03311 of total. Then, the probability of an edge being `award_nominee` in KG is about 3.3%.

Formally, the distribution to predict a fact is defined by Equation 3.1.

$$P(Y(ijk) = 1) = \sum_{m_k}^{|D|} \frac{d_{m_k}}{|D|}, \tag{3.1}$$

where $|D|$ is the number of paths in D, and each d_{m_k} is a valid path in D ending with a relation k. Then, the final distribution after this calculation can be sorted and ranked. Intuitively, the most probable relation after this calculation should be the correct fact between those entities.

Applying Equation 3.1 for `Morgan Freeman` and `Oscar` example results in: $i = MorganFreeman$, $j = Oscar$, $|D| = 4$, $|R| = 4$. For k (the relation to be predicted), there are three possible values: `hosted`, `won_award` and `None`[2]. The final distributions are: $hosted = 1/4$, for $None = 0$ and for $won\_award = 1/4 + 1/4 + 1/4 = 3/4$. Then, with 75% of probability, the relation to be predicted is `won_award`. For each relation k, there is a value in the final distribution table.

In summary, the problem tackled in this work is defined as: "Given a source and a target entities, what steps are necessary to predict a fact between them?"

[2]A non-relation between the source and target node.

Algorithm 1 ProA's algorithm to generate probability distributions of missing facts in a KG

```
Require: KG, source entity, target entity, length $\ell$
Ensure: Ranked predicted edges between source and target
 1: $D$ = all paths from source to target at most length $\ell$
 2: $D_{dist} = \emptyset$                    ▷ initialize distribution paths set
 3: $Gen_{dist} = \emptyset$                  ▷ initialize general distribution
 4: $Final_{dist} = \emptyset$                ▷ initialize final distribution
 5: for each $d \in D$ do
 6:     if $d \notin D_{dist}$ then
 7:         $D_{dist}$.add($d$)               ▷ add $d$ to set $D$
 8:     else
 9:         $D_{dist}[d]$.increment(1)        ▷ increment $d$ counter
10: for each $path \in KG$ do
11:     for each $d \in D$ do
12:         if $path = d$ then
13:             if $d \notin Gen_{dist}$ then
14:                 $Gen_{dist}$.add($d$)     ▷ add $d$ to set $Gen_{dist}$
15:             else
16:                 $Gen_{dist}$[d].increment(1)
17: for each $d \in D_{dist}$ and $d \in Gen_{dist}$ do
18:     $Final_{dist} = P(d \in D_{dist}) \times P(d \in Gen_{dist})$
19: Return: $Final_{dist}$
```

3.2 ProA Algorithm

Our solution, the ProA algorithm, gets all possible paths from a source node to a target node to infer which relation is most likely to connect between those two nodes. It satisfies the Markov property of the probability conditional distribution, since it depends upon the present state of the graph, according to Equation 2.2. In other words, the algorithm has the memoryless property of a stochastic process.

Also, ProA takes advantage of the distribution over two sets, D and Gen, to create a fact ranking. A relation ranking is generated from ProA's output and the predicting task can be applied properly since the best relation ranked is usually the right relation to be predicted.

ProA gets every path between the source and the target entities, and generates an initial probability D_{dist} from D. Then, ProA calculates the weight of each of those relations of D in the whole graph to get how often they are. This new set is called Gen_{dist}. Finally, the final distribution is given by Equation 3.2, in which each path d_i in D and Gen has its proper probability distribution.

Algorithm 1 shows ProA processing. First, lines 1-4 initialize the variables. Then,

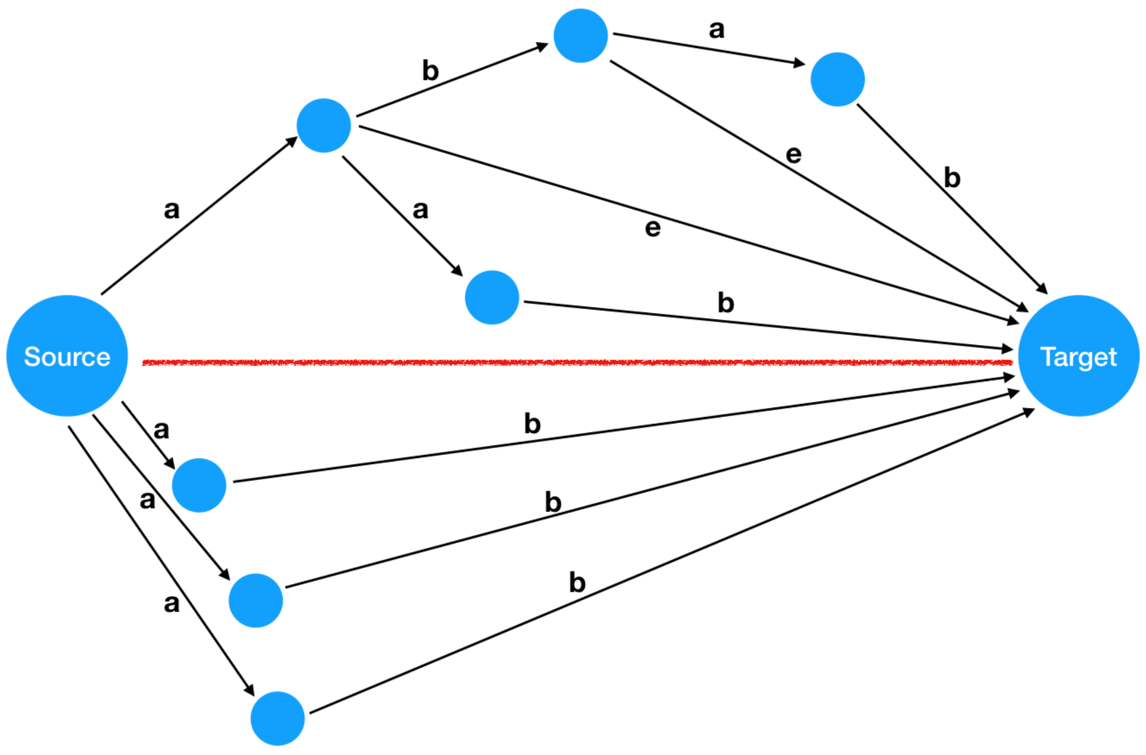


Figure 3.2. Link prediction based on path probabilistic distribution. To predict the red fact between source and target, ProA finds all possible paths between two nodes at most length ℓ (in this scenario $\ell = 3$). The set of possible paths is $\{(a, b), (a, e), (a, a, b), (a, b, e)\}$, and the most common path is (a, b). Usually, the fact $\{e\}$ exists when (a, b) is a path between two nodes; hence, ProA infers with a certain probability that $\{e\}$ could be the missing red fact.

the loop in lines 5-9 creates the probability distribution of paths between source and target at most length ℓ (or how many times they occur). Next, the loop in lines 10-16 gets the distribution over those paths in the whole graph structure. The final distribution is calculated in lines 17-18.

Again using Figure 3.1 with a piece of real Knowledge Graph that focuses on Morgan Freeman and the Oscars. Entity `Morgan Freeman` is connected to `actor` and `male` entities through relations `profession` and `gender`, respectively. Applying ProA algorithm for entities `Morgan Freeman` and `Oscar` finds all drawn paths to generate D. Then, for each path in D, a final probability distribution of facts that connect entities is calculated following Algorithm 1. In summary, ProA learns that `won_award` is more probable to occur for an actor that `hosted` from gender relation. The final distribution is about all relations found between source and target, and ProA can properly infer each missing relation.

Table 3.2. Example of probability distribution

Relation	**Probability**
(a,b)	0.5
(a,e)	0.1666666667
(a,a,b)	0.1666666667
(a,b,e)	0.1666666667

A higher level example of the whole link prediction process is illustrated in Figure 3.2. Given the paths distribution between two entities in the graph, the probabilistic algorithm learns about relations that usually connect the query entities (source and target). For example, between nodes `source` and `target` there are four kinds of paths $\{(a, b), (a, e), (a, a, b), (a, b, e)\}$, and the probability distribution is in Table 3.2.

ProA does not only infer missing relations between entities, but also predicts whether there is no relation between them. Considering the Knowledge Graphs Freebase and Wordnet, an initial empirical evaluation (performed while testing ProA) shows that `None` is the most probable relation to occur between entities connected by a path at most length three. Therefore, ProA learns about this special relation and infers whether there is a relation or not.

The calculation of time complexity for ProA is similar to well known algorithms in graphs such as Breadth-first search and Depth-first search. In the worst case, every node and relation will be explored, then ProA is at least $O(|V| + |E|)$. However ProA takes ℓ as argument, so the final time complexity is $O((|V|+|E|)^{\ell})$, which means if ProA considers the number of paths at most three, it has a time complexity of $O((|V|+|E|)^3)$ and so on.

3.3 Real Examples

In this section, we explain the algorithm based on two real examples from Freebase. We use the Freebase training file that has 483,142 entities, and we construct the graph according to all the relationships established in this file (details about this dataset are in Section 4.1). Once the graph is created, we select a fact at random, remove its relation between nodes, and run the algorithm to see if it can predict it or not. Then we run the algorithm with two entities that have no relation and we verify if it does not predict any relation, as expected.

Table 3.3. Path distribution between a movie and a genre.

1st Relation	2nd Relation	3rd relation	Times
/film_release_region	/tv/tv_program/country_of_origin	/film/film/genre	3
/films_in_this_genre	/films_in_this_genre	/film/film/genre	127
/film/film/genre	/tv/tv_genre/programs	/film/film/genre	2
/film_release_region	/film_release_region	/film/film/genre	150
/film/film_genre/films_in_this_genre	/film/film/genre	/media_common/netflix_title/netflix_genres	1
/film/film/production_companies	/film/production_company/films	/film/film/genre	9
//film_release_distribution_medium	/media_common/netflix_genre/titles	/tv/tv_program/genre	3
/film_release_region	/film/film/country	/media_common/netflix_title/netflix_genres	22
/film/film/genre	/film/film_format/film_format	/film/film/genre	1
/film/film_genre/films_in_this_genre	/film/film_genre/films_in_this_genre	/film/film_genre/films_in_this_genre	33
/film/film/subjects	/film/film_subject/films	/film/film/genre	3
/film_release_region	/film_release_region	/film/film_genre/films_in_this_genre	33
/film/film/genre	/films_in_this_genre	/films_in_this_genre	13
/film/film/production_companies	/film_film_distributor_relationship/film	/media_common/netflix_title/netflix_genres	1
/film/film/production_companies	/film_film_distributor_relationship/film	/film/film_genre/films_in_this_genre	7
/film_release_distribution_medium	/film_release_distribution_medium	/film/film/genre	11
/film_release_region	/film/film/country	/film/film/genre	182
/film/film/production_companies	/film/production_company/films	/film/film_genre/films_in_this_genre	1
/film_release_region	/film_release_region	/film/film/genre	10
/film/film/other_crew./film/film_crew_gig/film_crew_role	/film/film_crew_gig/film	/film/film/genre	63
/films_in_this_genre	/tv/tv_genre/programs	/tv/tv_program/genre	4
/film/film/starring./film/performance/actor	/film/actor/film./film/performance/film	/film/film/genre	3
/film/film_genre/films_in_this_genre	/film/film/genre	/film/film/genre	35
/film/film/genre	/media_common/netflix_genre/titles	/media_common/netflix_title/netflix_genres	2
/film/film_genre/films_in_this_genre	/film/film_genre/films_in_this_genre	/media_common/netflix_title/netflix_genres	13
/film/film/genre	/tv/tv_genre/programs	/tv/tv_program/genre	7
/film/film_crew_gig/film_crew_role	/film/film_crew_gig/film	/media_common/netflix_title/netflix_genres	3
/film/film/production_companies	/film/production_company/films	/media_common/netflix_title/netflix_genres	2
/film_release_region	/film/film/gross_revenue/currency	/film/film/genre	1
/film_release_region	/film_release_region	/film/film_genre/films_in_this_genre	3
/film/film/other_crew./film/film_crew_gig/film_crew_role	/film/film_crew_gig/film	/film/film_genre/films_in_this_genre	12
/film/film/subjects	/film/film/genre	/film/film_genre/films_in_this_genre	1
/film/film/genre	/film/film/genre	/film/film/genre	8
/film_release_region	/film_release_region	/media_common/netflix_title/netflix_genres	3
/film/film/production_companies	/film_film_distributor_relationship/film	/film/film/genre	29
/film/film/production_companies	/film_film_distributor_relationship/distributor	/film/film/genre	3
/film/film_crew_gig/film_crew_role	/film_crew_gig/film_crew_role	/film/film/genre	12
/film/film_genre/films_in_this_genre	/film/film/genre	/film/film_genre/films_in_this_genre	6
/film/film/genre	/film/film/genre	/film/film_genre/films_in_this_genre	4
/film_release_distribution_medium	/media_common/netflix_title/netflix_genres	/film/film/genre	1
/film/film/genre	/media_common/netflix_genre/titles	/film/film/genre	15
/film/film/other_crew./film/film_crew_gig/film_crew_role	/film/film/other_crew./film/film_crew_gig/film_crew_role	/film/film_genre/films_in_this_genre	2
/film_release_distribution_medium	/film/film/subjects	/film/film/genre	1
/film/film_genre/films_in_this_genre	/tv/tv_genre/programs	/film/film/genre	1
/film/film/genre	/film/film_genre/films_in_this_genre	/film/film/genre	51
/film_release_distribution_medium	/film/film_subject/films	/film/film/genre	3
/film/film/production_companies	/award/award_nomination/nominated_for	/film/film_genre/films_in_this_genre	3
/film_release_region	/tv/tv_program/country_of_origin	/tv/tv_program/genre	4
/film/film/subjects	/film/film_genre/films_in_this_genre	/film/film/genre	1
/film/film/genre	/media_common/netflix_genre/titles	/film/film_genre/films_in_this_genre	3
/film/film_genre/films_in_this_genre	/tv/tv_program/genre	/tv/tv_program/genre	1
/film/film/other_crew./film/film_crew_gig/film_crew_role	/film/film/other_crew./film/film_crew_gig/film_crew_role	/media_common/netflix_title/netflix_genres	1
/film_release_region	/film/film/country	/film/film_genre/films_in_this_genre	34
//film_release_distribution_medium	/film_release_distribution_medium	/film/film_genre/films_in_this_genre	1

Table 3.4. Final relation probability between *The Jetsons Meet the Flintstones* (movie) and *Romance film* (genre). ProA predicts that such two entities could be connected through `/film/film/genre` with 16% of chance.

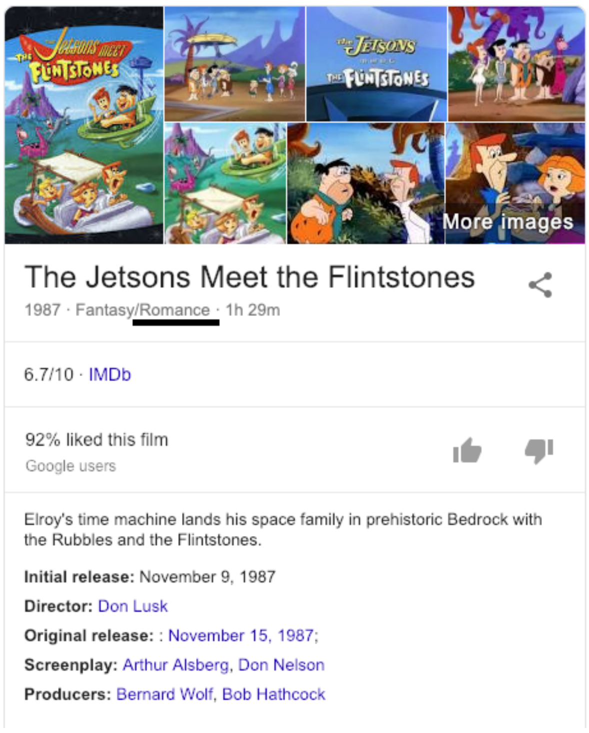
Relation	**Probability**
None	0.7396
/film/film/genre	0.1620
/film/film_genre/films_in_this_genre	0.0714
/media_common/netflix_title/netflix_genres	0.0217
/media_common/netflix_genre/titles	0.0030
/film/film_regional_release_date/film_release_region	0.0008
/film/film_regional_release_date/film_release_distribution_medium	0.0006
/film/film/country	0.0003

3.3.1 The Jetsons Meet the Flinstones and Romance

Regular Test. For the first test, the two entities selected at random were `/m/063zky` and `/m/02l7c8` which are *The Jetsons Meet the Flintstones* (movie) and *Romance film* (genre), respectively. The relationship between them is `/film/film/genre`, meaning the film is from the genre of romance. Then, the first step is to remove this relation connecting those two entities and check if after the algorithm ends the relation is predicted properly.

The algorithm receives as input the graph itself, the source node, the target node, and a size ℓ to check the maximum length of paths between the nodes. Then, it checks if there is any relation between both nodes; and if so, identifies such a relation. Next, the second step is to generate the probability distribution between all paths of size at most ℓ between the input nodes. Table 3.3 shows how many times each path occur between those nodes. The total number of paths is 948, and the most probable path for the input nodes is: [`/film_release_region`, `/film/film/country`, `/film/film/genre`], which occurs 182 times, or 19.19%. Note that `None` is a valid relation for our algorithm. ProA deals with it as explained in the next section.

Given the path distribution, the next step is to generate a dictionary with all occurred paths and how many times each one exists. Since it already knows how many relations the graph has, our solution calculates the final distribution for each path and the final probability of each relation to connect the source node to target node (i.e., `The Jetsons Meet the Flintstones` and `Romance film`, in this example). Finally, our solution ranks those relations in terms of how likely each connects them, as Table 3.4 shows.



More images

The Jetsons Meet the Flintstones

1987 · Fantasy/Romance · 1h 29m

6.7/10 · IMDb

92% liked this film

Google users

Elroy's time machine lands his space family in prehistoric Bedrock with the Rubbles and the Flintstones.

Initial release: November 9, 1987

Director: Don Lusk

Original release: : November 15, 1987;

Screenplay: Arthur Alsberg, Don Nelson

Producers: Bernard Wolf, Bob Hathcock

Figure 3.3. Knowledge Graph from Google showing information about *The Jetsons meet the Flinstones* movie. The genre of this movie is labeled as *Fantasy/Romance*.

In summary, given the distribution of paths between two nodes and the global path distribution over all the graph, ProA infers which relation should be the valid one between the input nodes. In this particular case, ProA predicts well that the two nodes be connected with a relation `/film/film/genre` with 16.19% of chance. To better illustrate, Google Knowledge Graph shows the properties for this movie in Figure 3.3.

Two Additional Tests. There are two more scenarios that we can test to see if ProA predicts correctly the right relation between the movie and the right genre. The first one is selecting as input a movie and a completely different entity – with which movie would not be connected because it does not make sense, such as Food Network[3]. The second test considers another genre that could be a relation for the selected movie.

Starting with Food Network, there is not many paths between those two entities. Then, the `None` relation has 99.5% probability to be the right relation between them. In other words, the algorithm predicts correctly since Food Network and the movie do not seem to be related at all.

In the second testing scenario, we try to infer a relation between the movie *The*

[3]It is a television channel

Table 3.5. Relation distribution between *The Jetsons Meet the Flintstones* (movie) and *Biography* (genre). ProA predicts they could be connected through `/film/film/genre` with 13.37% of chance, which is correct. However, it is more likely to be a Romance than a Biography given the distribution.

Relation	**Probability**
None	0.7767
/film/film/genre	0.1337
/media_common/netflix_title/netflix_genres	0.0381
/film/film_genre/films_in_this_genre	0.0284
/film/film_regional_release_date/film_release_region	0.0114
/film/film/country	0.0031

Jetsons Meet the Flintstones and the genre *Biography.* We know that the right genre is romance with a probability of 16.19%, as already discussed in the regular testing. Then, the algorithm gets the results from Table 3.5. The relation found makes sense since a movie could be connected to a genre through `/film/film/genre`, but it is most likely to be connect to Romance than Biography.

3.3.2 Anarcho Punk and Norfolk

Now, we select two nodes at random and make sure they do not have any relation. The entities are *Anarcho Punk* and *Norfolk*, which are a music genre and a United States city in the Virginia state, respectively. Intuitively, there is no relation between those since such a music genre is not related with Virginia city. Then, the expected result for ProA is to predict that there is no relation between them; i.e., the `None` relation should be predicted.

Creating the set D of paths between the entities gives the notion that there are not many relations between them. Then, `None` clearly appears as the most likely to be the relation between them, as Table 3.6 shows. If we ignore the `None` relation, the second most probable relation is `/music/genre/artists` with 0.01%, which is very small.

In fact, such results show that even not considering the `None` relation, the probability of the two entities being connected is very low. In the next section, we explain the `None` relation as it is important for predicting facts; however, in this test scenario, we clearly realize that the two vertices are not connected.

Table 3.6. Final relation probability between Anarcho Punk (music genre) and Norfolk (city). ProA predict that those two entities could not be connected at all.

Relation	**Probability**
None	0.9884
/music/genre/artists	0.0100
/music/artist/genre	0.0017
/media_common/netflix_genre/titles	0.000005
/broadcast/genre/content	0.0000005

3.4 The None Relation

The `None` relation, which is the absence of relation, is always the most common relation in the final probability table. As presented in Tables 3.4, 3.5 and 3.6, `None` has respectively 0.74%, 0.77% and 0.98% of chance to be predicted; however, as it is the most likely to be predicted, why is it not a final result?

ProA generates a distribution table of all possible relations from nodes to the target entity. Then, there are many entities not connected to others. The Knowledge Graph is usually dense, but not dense enough to have facts between every node. At the end, ProA identifies such properties on the KGs and learns from them.

For example, in Section 3.3.2, Anarcho Punk and Norfolk are not connected. However, if ProA generates a table distribution from Virginia (state) and Norfolk (city), and there is a relation from Anarcho Punk in the middle of paths (between the city and state), the `None` relation will be counted, because the music genre is not connected to the city as we demonstrated.

Based on empirical experiments, if `None` has probability smaller than 92%, the first relation found is the most likely relation that ProA calculated to be correct. If `None` probability is equal to or greater than 92%, then ProA infers that there is no relation between the vertices and, hence, there is no edge to be added in the graph between the two entities.

Chapter 4

Experiments and Results

In order to evaluate ProA performance, we first detail the dataset over which we build a knowledge graph. Then, we present two of the most common metrics for evaluating prediction on KG. Last, we present our evaluation and results.

4.1 Datasets

In order to test and evaluate ProA, we use two datasets: Freebase and Wordnet. Freebase[1] was a large collaborative knowledge base that contains data mainly added by its community members. It was an online collection of structured data harvested from many sources, including individual, user-submitted wiki contributions. Freebase aimed to create a global resource to allow people (and machines) to access common information more effectively. On 16 December 2014, Knowledge Graph announced that it would shut down Freebase over the succeeding six months and help with the move of the data from Freebase to Wikidata, which is another Knowledge Base.

Wordnet[2] [28] is a large lexical database for English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser.

Following [10][3], this work evaluates the probabilistic approach sampling on the Freebase dataset (FB15k) and WordNet dataset (WN18). They are very different in coverage: FB15k contains mostly named entities connected through strongly typed

[1] http://www.freebase.com/

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

[3] A very well cited paper with over 680 citations in June, 2018.

Table 4.1. Dataset details

Dataset	Entities	Relations	Train	Test	Valid
FB15K	14,951	1,345	483,142	59,071	50,000
WN18	40,943	18	141,442	5,000	5,000

relations, whereas WN18 contains mostly common nouns connected through lexical and semantic relations. Table 4.1 presents the dataset statistics.

The data contain relations with high variation in the number of instances – 39% of the relations have at most 10 instances, while the most frequent relation[4] has almost 16,000. This disparity is also reflected in the distribution of node degrees: 12% of the entities have degree equal or less than 10 (appear in at most 10 instances). The average degree of a node in FB15k is approximately 13.2 overall, and 32.4 on the training data [20].

4.2 Evaluation Metrics and Process

As evaluation metrics, we consider Mean Reciprocal Rank (MRR) and Hits@K, which are commonly used for link prediction [10]. The MRR is the average of the reciprocal ranks of results for a sample of queries N. The reciprocal rank of a query response is the inverse of the rank of the first correct answer. For example, if the correct result is returned in the first place, its MRR is 1. If such expected result appears in second place, its MRR is 1/2; for third place it is 1/3, and so on.

Formally, the MRR is defined by Equation 4.1.

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}, \tag{4.1}$$

where $rank_i$ is the rank of the positive instance i predicted by the model with respect to the negative samples. Table 4.2 shows an example of how MRR is calculated.

In a different perspective, Hits@K is how far the predictions are from the positive sample. It is calculated by Equation 4.2.

$$Hits@K = \frac{|\{i|rank_i < K\}|}{N}, \tag{4.2}$$

The evaluation process is: select N random facts from the graph; remove the edges connecting those entities; execute ProA between them to verify how effective the

[4]/award/award nominee/award nominations./award/award nomination/award nominee

Table 4.2. MRR and Hits example. The final MRR is calculated by Equation 4.1: $1/3 \times (1/2 + 1 + 1/3) = 11/18$ (0.61). About Hits, the correct response was predicted at the first place once (`was_born_in`) and three times at least in third place; so Hits@1 and Hits@3 are 1/3 and 1 respectively.

Query	**Proposed Results**	**Correct Response**	**Rank**	**Reciprocal rank**
won_award	hosted, **won_award**	won_award	2	1/2
was_born_in	**was_born_in**	was_born_in	1	1
married_with	was_born_in, hosted, **married_with**	married_with	3	1/3

Table 4.3. MRR, Hits@1, Hits@3 and Hits@10 on FB15K and WN18 datasets. The algorithms are sorted by best results in decreasing order.

	MRR		**Hits@1**		**Hits@3**		**Hits@10**	
Algorithm	**FB15K**	**WN18**	**FB15K**	**WN18**	**FB15K**	**WN18**	**FB15K**	**WN18**
[27] ER-MLP	0.288	0.712	0.317	0.775	-	-	-	-
[10] TRANSE	0.380	0.454	0.231	0.089	0.472	0.823	0.641	0.934
[32] RESCAL	0.354	0.890	0.409	0.904	-	-	-	-
[31] HOLE	0.524	0.938	0.402	0.93	0.613	**0.945**	0.739	**0.949**
[49] DISTMULT	0.654	0.938	0.546	0.728	0.733	0.914	0.824	0.936
[44] COMPLEX	0.692	0.941	0.599	0.936	0.759	**0.945**	0.840	0.947
[19] SIMPLE	**0.727**	**0.942**	**0.660**	**0.939**	0.773	0.944	0.838	0.947
PROA	0.706	0.483	0.563	0.304	**0.861**	0.701	**0.962**	0.703

prediction is. Then, given the results and the correct response, we evaluate how many times an algorithm gives the correct response at K place. Table 4.2 exemplifies it as well.

Since the probabilistic algorithm generates a list of possible facts (including the `None` relation), the evaluation is based on how accurate the prediction is in terms of *MRR* and *Hits* using Equations 4.1 and 4.2.

4.3 Results

We now proceed by presenting the results of the evaluation. The evaluation considers ProA and each of the algorithms from the state of the art presented in Section 2.4. We tested ProA against 2,000 facts in FB15K and WN18 with $\ell = 3$, and the results from other algorithms were taken from their papers.

Table 4.3 presents the results. Specifically, the columns are grouped by the metrics (previous section) calculated over each of the two datasets. *ER-MLP* and *RESCAL* do not provide information about Hits@3 and Hits@10.

The resuls show that our probabilistic approach (ProA) has competitive results

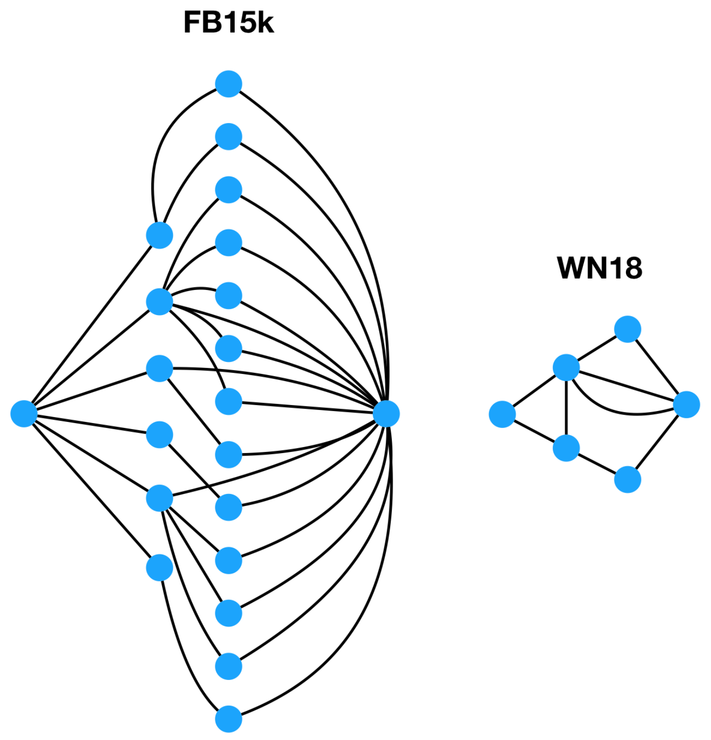


Figure 4.1. Differences in the number of paths between FB15k and WN18: there are four paths from entity source to entity target in WN18 and 36 paths in FK18k.

in terms of accuracy over baselines for FB15K. The main reason is dense graphs have multiple paths between entities, and ProA learns from those and properly infers the missing facts. Then, the Freebase dataset has an average degree of 32.4 that indicates a high level of relation between entities, which could classify it as a dense graph.

SimplE, which is a very recent work, has the best results in terms of MRR for FB15K and WN18 datasets. The *MRR* results from ProA show a small improvement over other works but SimplE, without significant statistical difference over *Complex* and *DistMult*. However, there is a significant improvement for Hits@3 and Hits@10 indicating that ProA often has a good precision under ranking of possible facts.

On the other hand, Wordnet has average degree of three. Then, Table 4.3 shows that ProA does not fully learn the paths. In our complementary evaluations of this dataset, the number of paths between two entities is often one, and then ProA cannot properly predict based solely in one relation. Moreover, ProA has a considerable hits rate in sparse graphs indicating that in 70% of cases, the number of facts predicted is in the top three and top 10 results.

To better clarify such results, consider two small, actual samples from the datasets

Table 4.4. Relation ranking generated from Figure 4.1

Ranking generated for FB15K	
Relation	**Probability**
None	0.739
/music/genre/artists	0.211
/music/artist/genre	0.049

Ranking generated for WN18	
Relation	**Probability**
None	0.682
_member_of_domain_topic	0.298
_hypernym	0.008
_hyponym	0.007
_synset_domain_topic_of	0.002
_derivationally_related_form	0.0002
_member_holonym	0.0001

with structure illustrated in Figure 4.1 and relations with probabilities in Table 4.4. Clearly, FB15k has more paths than WN18 (an overall feature that is true for the whole dataset). In this specific example, FB18k has 32 more paths between source and target entities than WN18. As already mentioned, ProA takes advantage of paths in the graph; hence, for sparse graphs as WN18 in this example, the task becomes much more difficult.

Specifically, for WN18, the right relation to be predicted is `_member_holonym`. However, according to Table 4.4, for ProA `_member_holonym` is at seventh place with a very low probability. For FB15k, ProA correctly predicts `/music/genre/artists` relation with high probability. Moreover, once again, even having `None` with more probability than other relations, ProA learns that such a relation is only considered when it has probability larger than 0.92.

In a different perspective over the same example, in terms of MRR and Hits@K, ProA gets 1.0 in MRR for FB15k, because the first relation in the ranking is the desired one. Likewise, Hits@1, Hits@3 and Hits@10 also get 1.0. However, in WN18, the MRR is 0.166, Hits@1 is 0, Hits@3 is 0, and Hits@10 is 1; because the desired relation is at seventh place.

Overall, ProA works well with dense Knowledge Graphs (such as Freebase) because it requires a certain amount of paths to generate the paths probability distribution. For KGs with less paths between entities, the probabilistic algorithm could not learn properly and then the prediction is reasonable. Wordnet is an example of sparse Knowledge Graph, which means there is a few number of paths between nodes, making it difficult for ProA.

Chapter 5

Conclusion

This work presents ProA, a probabilistic algorithm to predict missing facts in Knowledge Graphs. The approach generates probability distributions over paths between entities and predicts missing facts. In ProA, the paths are exploited to create a probabilistic model that can capture rich interactions in relational data. Since other predicting algorithms present complex mathematical calculations and generally a considerable space/time complexity, ProA takes advantage of the current topological state of the graph to predict missing relations with a good accuracy.

Our experimental evaluation shows ProA provides a good performance against the state-of-the-art, specially in dense graphs, then solving a complex problem as link prediction. Moreover, ProA is available on GitHub[1] and it is easy to use. Intuitively, user can setup a Knowledge Graph dataset as input and define parameters such as the most allowed length of paths, and then ProA predicts every possible fact.

As future work, we plan to further exploit the probability distribution over sparse graphs and apply ProA to complex networks, as the topological structure is important. A related idea is to explore the presence of relations `None` and correlate it with the concept of *dull nodes* as proposed in [41].

Also, there are runtime improvements to be done in ProA such as applying dynamic programming to avoid recalculating the probability already calculated. It is also necessary to investigate how to improve results in sparse graphs. Moreover, an interesting topic is applying ProA not only for link prediction but for other important task over Knowledge Graphs such as entity recognition.

[1]https://github.com/andrehigher/proa

Bibliography

[1] Angeli, G. and Manning, C. D. (2013). Philosophers are mortal: Inferring the truth of unseen facts. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 133--142, Sofia, Bulgaria.

[2] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, pages 722--735, Boston, MA, USA.

[3] Bacchus, F., Grove, A. J., Halpern, J. Y., and Koller, D. (1996). From statistical knowledge bases to degrees of belief. *Artif. Intell.*, 87(1-2):75--143. ISSN 0004-3702.

[4] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370--418.

[5] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. (1994). The world-wide web. *Commun. ACM*, 37(8):76--82.

[6] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, pages 29–37.

[7] Biega, J., Kuzey, E., and Suchanek, F. M. (2013). Inside yago2s: A transparent information extraction architecture. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 325--328, New York, NY, USA.

[8] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1--22.

[9] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247--1250, Vancouver, Canada. ACM.

[10] Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 2787--2795, USA.

[11] Brandão, M. A. and Moro, M. M. (2017). Social professional networks: A survey and taxonomy. *Computer Communications*, 100:20--31.

[12] Chekol, M. W., Pirrò, G., Schoenfisch, J., and Stuckenschmidt, H. (2017). Marrying uncertainty and time in knowledge graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 88--94, San Francisco, CA, USA.

[13] Crichton, G., Guo, Y., Pyysalo, S., and Korhonen, A. (2018). Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches. *BMC Bioinformatics*, page 19:176.

[14] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601--610, New York, NY, USA. ACM.

[15] Georgala, K., Hoffmann, M., and Ngomo, A. N. (2017). An evaluation of models for runtime approximation in link discovery. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 57--64.

[16] Gwiazda, J., Ong, E., Held, R., and Thorn, F. (2000). Myopia and ambient night-time lighting. *Nature*, 404:144 EP –.

[17] Hendler, J. (2001). Agents and the semantic web. *IEEE Intelligent Systems*, 16(2):30--37. ISSN 1541-1672.

[18] Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys*, 6(1):164--189.

[19] Kazemi, S. M. and Poole, D. (2018). Simple embedding for link prediction in knowledge graphs. *CoRR*, abs/1802.04868.

[20] Kotnis, B. and Nastase, V. (2017). Analysis of the impact of negative sampling on link prediction in knowledge graphs. *CoRR*.

[21] Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529--539.

[22] Laplace, P. (1812). *Théorie analytique des probabilités.* Courcier, Paris.

[23] Lenat, D. B. and Guha, R. V. (1989). *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. ISBN 0201517523.

[24] Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031.

[25] Ma, Y., Gao, H., Wu, T., and Qi, G. (2014). Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In *CSWS*, volume 480 of *Communications in Computer and Information Science*, pages 29--41. Springer.

[26] Maki, T., Takahashi, K., Yamaguchi, A., Wakahara, T., Kobayashi, T., Kodate, A., and Sonehara, N. (2016). Link prediction of LOD by multiple label propagation algorithm considering semantic distance. In *IEEE 5th Global Conference on Consumer Electronics, GCCE 2016, Kyoto, Japan, October 11-14, 2016*, pages 1--4.

[27] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

[29] Moravec, H. (1988). *Mind Children: The Future of Robot and Human Intelligence.* Harvard University Press, Cambridge, MA, USA. ISBN 0-674-57616-0.

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

[31] Nickel, M., Rosasco, L., and Poggio, T. A. (2015). Holographic embeddings of knowledge graphs. *CoRR*, abs/1510.04935.

[32] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 809--816.

[33] Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271--280. ACM.

[34] Nishioka, C. and Scherp, A. (2017). Keeping linked open data caches up-to-date by predicting the life-time of RDF triples. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 73--80.

[35] Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161--172.

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

[37] Paulheim, H. and Bizer, C. (2013). Type inference on noisy RDF data. In *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 510--525. Springer.

[38] Paulheim, H. and Bizer, C. (2014). Improving the quality of linked data using statistical distributions. *Int. J. Semant. Web Inf. Syst.*, 10(2):63--86. ISSN 1552-6283.

[39] Pipino, L. L., Lee, Y. W., and Wang, R. Y. (2002). Data quality assessment. *Commun. ACM*, 45(4):211--218. ISSN 0001-0782.

[40] Quinn, G. E., Shin, C. H., Maguire, M. G., and Stone, R. A. (1999). Myopia and ambient lighting at night. *Nature*, 399:113 EP –.

[41] Sett, N., Chattopadhyay, S., Singh, S. R., and Nandi, S. (2016). A time aware method for predicting dull nodes and links in evolving networks for data cleaning. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 304--310.

[42] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of International Conference on World Wide Web*, pages 697--706, Banff, Canada.

[43] Suh, B., Convertino, G., Chi, E. H., and Pirolli, P. (2009). The singularity is not near: Slowing growth of wikipedia. In *Proceedings of the 5th International Symposium on Wikis and Open Collaboration*, pages 8:1--8:10, New York, NY, USA. ACM.

[44] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2071--2080.

[45] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78--85.

[46] West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., and Lin, D. (2014). Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515--526, New York, NY, USA. ACM.

[47] Xu, P. and Barbosa, D. (2018). Investigations on knowledge base embedding for relation prediction and extraction. *arXiv*, 1802.02114.

[48] Xu, Z., Pu, C., and Yang, J. (2016). Link prediction based on path entropy. *Physica A: Statistical Mechanics and its Applications*, 456:294–301.

[49] Yang, B., Yih, W., He, X., Gao, J., and Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575.

[50] Zimmermann, A., Gravier, C., Subercaze, J., and Cruzille, Q. (2013). Nell2rdf: Read the web, and turn it into RDF. In *Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data*, pages 2--8, Montpellier, France.