Comparison of Symbolic & Sub-Symbolic Approaches for Knowledge Graph Completion

Master Thesis

presented by
Lars Joormann
Matriculation Number 1721931

submitted to the
Data and Web Science Group
Prof. Dr. Stuckenschmidt
University of Mannheim

August 2022

Abstract

Add a brief summary!

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

1.2 Research Question

Chapter 2

Knowledge Graphs

Knowledge graphs are graph-structured knowledge bases. They store information in the form of relationships between entities. A single information in the graph is referred to as fact. It consist of two entities and a relation between those two. This can be expressed as a triple: $(head, relation, tail)$. Knowledge Graphs are referred to as graphs since their entities can be interpreted as nodes and the relations as labelled and directed edges in a graph. The label here indicates which kind of relation the entities share and the direction indicates which entity is the head entity and which the tail entity i.e. an edge points from the head to the tail. An example for a knowledge graph can be seen in figure 2.1. [8]

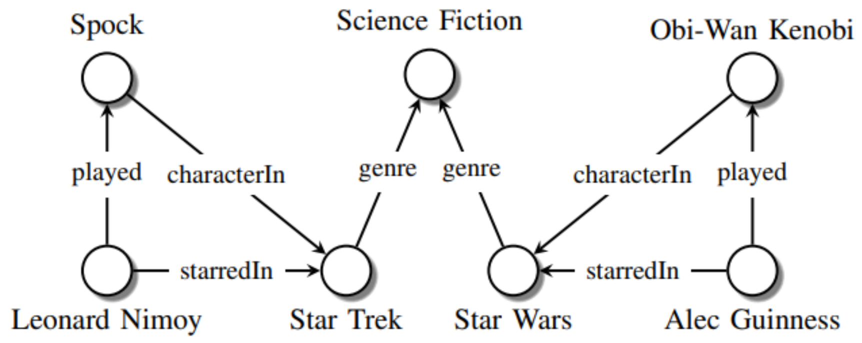
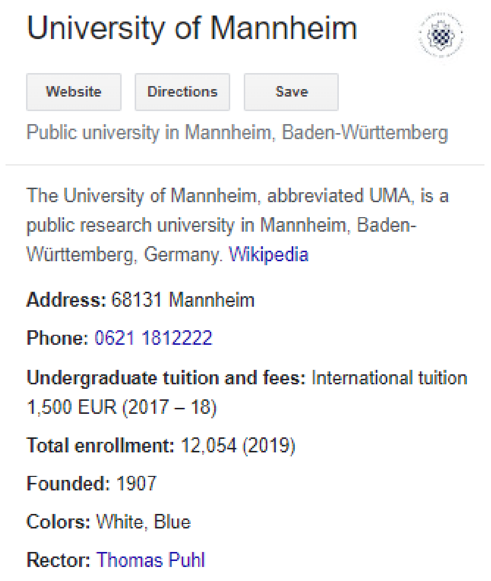


Figure 2.1: Example of a Knowledge Graph

The term 'knowledge graph' was first introduced by Google in 2012 [13]. In their blog they spoke about how they use their knowledge graph to enrich search engine results. The most noticeable part of how they use knowledge graphs are the side windows when searching for an entity with their search engine. An example of such can be seen in figure 2.2. Here we can see what kind of knowledge Google has about the University of Mannheim. For example it seems to be that *(University of Mannheim, founded_in, 1907)* is one of the facts in their Knowledge Graph.


University of Mannheim

Public university in Mannheim, Baden-Württemberg

The University of Mannheim, abbreviated UMA, is a public research university in Mannheim, Baden-Württemberg, Germany. Wikipedia

Address: 68131 Mannheim

Phone: 0621 1812222

Undergraduate tuition and fees: International tuition 1,500 EUR (2017 – 18)

Total enrollment: 12,054 (2019)

Founded: 1907

Colors: White, Blue

Rector: Thomas Puhl

Figure 2.2: Example of a Google Side Window

In the recent years knowledge graphs have become more and more popular and have found their way into further applications than search engines. An overview of applications can be seen in figure 2.3. Question Answering systems use knowledge graphs to enhance their results. Examples here include social chatbots and digital assistance like Siri. Recommender Systems leverage knowledge graphs as side information to improve and diversify their recommendations. Moreover, knowledge graphs are also used in information retrieval, domain-specific applications and more. [17]

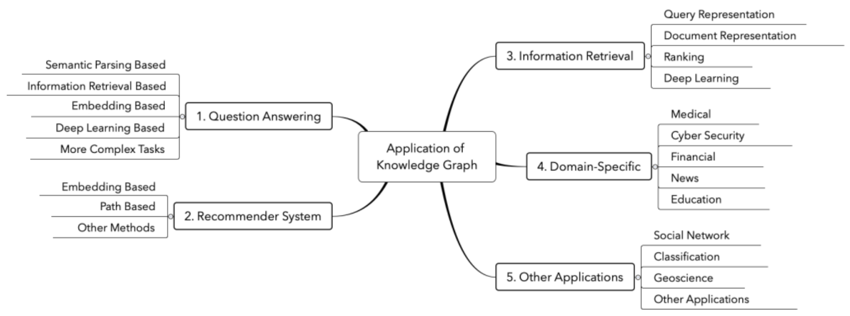


Figure 2.3: Applications of Knowledge Graphs

According to Paulheim [12] a knowledge graph is defined by the following four characteristics:

1. ”mainly describes real world entities and their interrelations, organized in a graph”
2. ”defines possible classes and relations of entities in a schema”
3. ”allows for potentially interrelating arbitrary entities with each other”
4. ”covers various topical domains”

The first characteristic defines that knowledge graphs consist of two kind of instances, entities and relations. An entity can be almost everything from an individual person to any kind of object. These entities are then linked through different kind of relations, which forms our graph.

The schema of the graph plays only a minor role. In most cases the instance-level statements (entities and triples) far outnumber the schema-level statements (entity classes and relations).

With the third characteristic Paulheim opens up the possibility that there are arbitrary relations between entities which are not included in the knowledge graph. The chapter 3 will discuss this further.

Lastly, another characteristic of knowledge graphs is that they do not focus on a single domain but interlink multiple topical domains.

2.1 Datasets

Chapter 3

Knowledge Graph Completion

As discussed in section 2 knowledge graphs are not complete. They contain noisy and incomplete data. It is practically impossible to cover every possible entity and relation existing in the real-world or even in their domain. There might be missing entities and relations or a Knowledge Graph can include two entities/ relations representing the same real-world entity. Knowledge graph completion tries to tackle these and other problems. It can be seen as a way of data cleaning for knowledge graphs. The solutions to the problems are defined into clear tasks, these include: entity resolution and entity and link prediction approaches.

Entity Resolution is according to Talburt "the process of determining whether two [entities] are referring to the same object or to different objects". [14]

Entity Prediction is the task of integrating new entities into the knowledge graphs. These entities are are discovered from other external sources and the knowledge graph includes no information about them. The goal is to find all possible relations this new entity has to the entities already existing in the graph. [1, p. 1]

Link Prediction is quiet similar to entity prediction. Instead of finding links for a new entity the goal here is to find all missing relations between already existing entities. [4, p. 125] Link prediction can be approached in two different ways: entity classification and triple classification. "Entity classification tries to predict the type or the class of an entity [...]" [6] For a triple with a missing tail $(h, r, ?)$ the goal would be to list all entities which fit into the tail along with their confidence. Triple classification on the other hand is a binary tasks. Here the input is a compete triple (h, r, t) and the goal is to predict whether this triple is true or not. [6]

In the following we are going to focus on the task of link prediction. There are various models tackling the problem. They can be categorized into one of the following two categories: symbolic and sub-symbolic approaches. Symbolic approaches are based on a high-level symbolic representation of the problem and try to reason about the problem with logic, this makes them comprehensible for humans. Sub-symbolic approaches on the other hand, solve the problem through less explainable mathematical equations. [6]

3.1 Symbolic Approaches

Symbolic approaches, or also sometimes referred to as ”Good Old Fashioned Artificial Intelligence”, focus on learning hypothesises in a symbolic (logical) language. [**?**] The strength of models from this approach lies in their explainability. Since the learned hypothesis are expressed in a logical language, either as axioms or rules, they are interpretable for a human and we can therefore reason about their predictions. In the following we will focus on rule-based models because they are more present for the link prediction tasks. [5] Rules are basically if-then clauses. Each rule has one or more conditions which have to be true and if they hold true the rule ”fires” and its conclusion can be derived. They are often written down as Horn clauses, an example for a Horn clause can be seen in its disjunctive and implicative form in equation 3.1 and 3.2 respectively. [**?**]

$$A \land \neg B_1 \lor \neg B_2 \lor ... \lor \neg B_n \tag{3.1}$$

$$A \leftarrow B_1, B_2, ..., B_n \tag{3.2}$$

Both equations express the same clause which states that if all conditions B, also called body atoms, are true so is the conclusion A, which is also referred to as the head.

3.1.1 AnyBURL

AnyBURL [7], short for Anytime Bottom-Up Rule Learning, is a link prediction algorithm that generates a set of rules from an existing knowledge graphs and leverages the generated rules to make predictions. The core idea behind AnyBURL is that ”[...] sampled paths from a knowledge graph (random walks) are examples of very specific rules, which can be transformed into more general rules.” [11]

Learning Rules

Rules are learned bottom-up, meaning we start with a sampled random path in the knowledge graph which then gets generalized into a rule. The sampled random path is called the ground path or grounding. In [7] generalized rules are classified into three types AC_1, AC_2 and C.

C rules are generalizations of cyclic ground paths:

$$h(Y, X) \leftarrow b_1(X, A_2), ..., b_n(A_n, Y). \tag{3.3}$$

AC_2 rules are generalizations of acyclic ground paths:

$$h(c_0, X) \leftarrow b_1(X, A_1), ..., b_n(A_n, A_{n+1}). \tag{3.4}$$

AC_1 rules can be generalized from both cyclic (with $c_0 = c_{n+1}$) and acyclic ones ($c_0 \neq c_{n+1}$):

$$h(c_0, X) \leftarrow b_1(X, A_1), ..., b_n(A_n, c_{n+1}). \tag{3.5}$$

X and Y are for variables appearing in the head of the Horn clause, A_i is a variable appearing only in the body and c_i is a constant. Furthermore, $h(e_1, e_2)$ and $b_i(e_1, e_2)$ are another notation to express a triple (e_1, h, e_2) or (e_1, b_i, e_2).

3.2 Sub-Symbolic Approaches

Sub-symbolic approaches are based on statistics. They try to learn numerical models from the existing facts in the knowledge graph. These models can then be used to predict the existing of a triple. [8] The most prominent models for knowledge graph completion of this approach are embedding-based models. These models learn a vector representation for each entity and relation. This is also often referred to as an embedding. In this representation it is then assumed that similar entities and similar relations will have similar vectors. An example for these embeddings can be seen in figure 3.1. On the left side we see a knowledge graph with three entities and two relations. Every entity and relation are represented on the right side by an embedding. Our two entities *Washington D.C* and *New York City* are both cities and therefore we can assume that their semantic meaning are quiet similar. The embeddings of these two entities are also quiet similar which demonstrates that our previous assumption is correct. [2] Our embeddings are also called latent features because they can not be directly observed in the data. Instead our model has to infer these features from the data. [8]

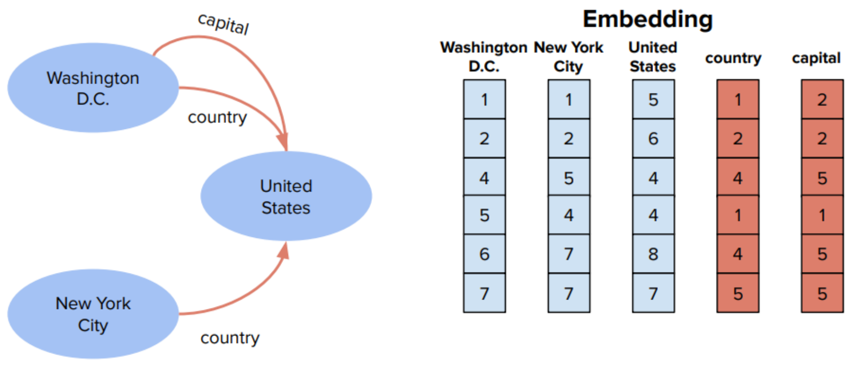


Figure 3.1: Example of an Embedding for a Knowledge Graph

An embedding-based model is defined by three characteristics [2]:

1. representations of entities and relationships
2. the scoring function
3. the loss function

As stated earlier our entities and relations are represented through vectors, their embeddings. Some models vary from this a bit and use complex numbers instead of real ones [15] or use matrices to represent relationships [9].
The score function $f(h, r, t)$ calculates the distance between the embeddings of two entities relative to their relation. If the triple holds true, its score should be in an optimal case equal to 1.
Lastly the loss function defines the objective which is going to be minimized during the training of our model where the embeddings for our entities and relations are learned.

3.2.1 RESCAL

One such an embedding-based model is RESCAL [9][10]. The model explains triples via pairwise interactions of latent features. It calculates the score of a triple as

$$s(h, r, t) = e_h^T M_r e_t \tag{3.6}$$

where $e_h, e_t \in \mathbb{R}^D$ and $M_r \in \mathbb{R}^{D*D}$. In this formula the interactions between two entity vectors are captured using only multiplicative terms, which makes this a bilinear model.

Furthermore, we can see that entities are encoded into a D-dimensional vector representation and have the same representation regardless of whether they occur as head or tail entity in a triple. Moreover, they also share the same representation for entities independent of the relation in a triple. Meaning that for every relation the vector of an entity stays the same. In [8] the authors of the RESCAL model argue that this allows their model to "[...] propagate information between triples [...]" and "[...] capture global dependencies in the data.". The relation in equation 3.6 is represented via a matrix M_r. Important to note about this matrix is its asymmetry. This allows the model to capture asymmetric relations. E.g. while the model should predict the triple $(Darth\_Vader, father\_of, Luke\_Skywalker)$ as ***True***, the triple $(Luke\_Skywalker, father\_of, Darth\_Vader)$ should be predicted as *False*.

Training through RESCAL-ALS

The model is trained through finding estimates for the matrices E and M_r which can be achieved by solving following regularized minimization problem

$$\min_{E,M_r} f(E, M_r) + g(E, M_r) \tag{3.7}$$

where

$$f(E, M_r) = \frac{1}{2}(\sum_{h,r,t} \|X_{hrt} - s(h, r, t)\|_F^2) \tag{3.8}$$

and

$$g(E, M_r) = \frac{1}{2}\lambda(\|E\|_F^2 + \sum_k \|M_r\|_F^2). \tag{3.9}$$

The term g is included as regularization term to prevent the model from overfitting. The regularized minimization problem in equation 3.7 is solved with an for RESCAL adjusted version of the alternating least-squares approach called RESCAL-ALS. This approach consists of a sequence of very efficient, closed-form updates as described in [10]. This training method assumes the closed-world assumption.

Training through Pairwise Loss

In [8] an alternative training method under the open-world assumption is presented which is more similar to the way other embedding-based models are trained. They refer to it as pairwise loss. Here two sets of triples are used to train the model, positives and negatives denoted as D^+ and D^- respectively. The positive triples are every known existing triple in our knowledge graph and the negative triples are created by corrupting existing triples. For the triples in the negative set it is not guaranteed that they are negative since a randomly corrupted triple could actually be an unknown true fact, but they are seen as *assumed-to-be-negative*.

Using stochastic gradient descent or any of its variations following objective function is then minimized

$$\min_{E,M_r} \sum_{x^+ \in D^+} \sum_{x^- \in D^-} \mathcal{L}(s(x^+), s(x^-)) + g(E, M_r) \tag{3.10}$$

using the same regularization as in equation 3.9 and where $\mathcal{L}$ is a margin-based ranking loss function such as

$$\mathcal{L}(s, s') = max(1 + s' - s, 0). \tag{3.11}$$

The usage of the margin-based loss has the advantage that negative triples do not have to be necessarily negative, they just have to be ”more negative” than the positive ones.

3.2.2 DistMult

DistMult [16] is another embedding-based model. It also uses the same basic bilinear scoring function as RESCAL in equation 3.6. The difference here is that DistMult restricts the relation matrix M_r to be a diagonal matrix. This reduces the parameters for the learnable parameters for a single relation from $D * D$ to just D. Therefore DistMult can be seen as a simplified version of RESCAL with the score function

$$s(h, r, t) = e_h^T diag(r) e_t \tag{3.12}$$

where e_h, e_t and $r \in \mathbb{R}^D$. The reduction of the relation dimensionality makes the model more computational efficient, especially on large knowledge bases, but also less expressive than RESCAL. Chen [3] provides a good figure to compare the two models which can be seen in figure 3.2.

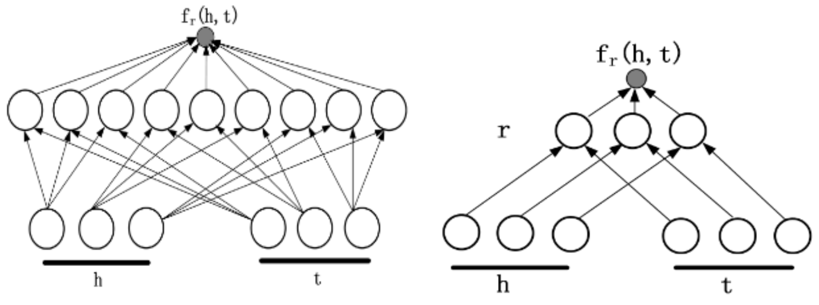


Figure 3.2: RESCAL vs. DistMult

Here our dimension size is $D = 3$. As we can see in RESCAL this leads to us having 3^2 latent features for the relation while DistMult requires only 3.

3.2.3 ComplEx

While DistMult is more effective in regard to time and space complexity, it can't express antisymmetric relations. RESCAL on the other hand can express this but the size of the relation matrix leads to an explosion in the number of parameters. To combine the advantages of both methods Trouillon proposes a model called ComplEx [15]. ComplEx uses a similar score function as DistMult but instead of calculating the dot product between the embeddings as real numbers, it uses complex vectors to represent the entities and relations. The dot product of complex vectors is referred to as the Hermitian dot product. Other than the dot product of real numbers it is not symmetrical. This enables the model to capture asymmetric relations while retaining the efficiency benefits of DistMult. The score function of ComplEx is

$$s(h, r, t) = Re(< w_r, e_h, \bar{e}_t >) \tag{3.13}$$

where e_h, e_t and $w_r \in \mathbb{C}$. To use this score function in our loss function, as defined in equation 3.11, the resulting scores need to be purely real. Therefore after calculating the Hermitian dot product only the real part is kept.

3.2.4 ConvE

3.3 Comparison of the Approaches

3.4 Model Evaluation

Chapter 4

Experimental Setting

Chapter 5

Comparison of Symbolic and Sub-Symbolic Performances

Chapter 6

Test Sets for Vulnerabilities

Chapter 7

Discussion

Chapter 8

Conclusion

Bibliography

[1] Matthias Baumgartner, Daniele Dell'Aglio, and Abraham Bernstein. Entity Prediction in Knowledge Graphs with Joint Embeddings. In *Proceedings of the Fifteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-15)*, pages 22–31, Mexico City, Mexico, June 2021. Association for Computational Linguistics.

[2] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge Graph Embeddings and Explainable AI. Technical report, April 2020. arXiv:2004.14843 [cs].

[3] Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. Knowledge Graph Completion: A Review. *IEEE Access*, 8:192435–192456, 2020. Conference Name: IEEE Access.

[4] Jennifer Golbeck. *Analyzing the Social Web*. Newnes, February 2013. Google-Books-ID: XP8jc2cDNrwC.

[5] Ioannis Hatzilygeroudis and Jim Prentzas. Neuro-Symbolic Approaches for Knowledge Representation in Expert Systems. *International Journal of Hybrid Intelligent Systems*, 1(3-4):111–126, January 2005.

[6] Eleni Ilkou and Maria Koutraki. *Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies?* November 2020.

[7] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3137–3143, Macao, China, August 2019. International Joint Conferences on Artificial Intelligence Organization.

[8] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015. arXiv:1503.00759 [cs, stat].

[9] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. page 8, 2011.

[10] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280, Lyon France, April 2012. ACM.

[11] Simon Ott, Christian Meilicke, and Matthias Samwald. SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models. *arXiv:2109.08002 [cs]*, September 2021. arXiv: 2109.08002.

[12] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, December 2016.

[13] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[14] John Talburt. *Entity Resolution and Information Quality*. Elsevier, January 2011. Google-Books-ID: tIB0IZYR8V8C.

[15] Theo Trouillon, Johannes Welbl, and Sebastian Riedel. Complex Embeddings for Simple Link Prediction. page 10, 2016.

[16] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. Technical Report arXiv:1412.6575, arXiv, August 2015. arXiv:1412.6575 [cs] type: article.

[17] Xiaohan Zou. A Survey on Application of Knowledge Graph. *Journal of Physics: Conference Series*, 1487(1):012016, March 2020.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 31.08.2022 Unterschrift