University of Zurich^UZH

Dependent Learning of Entity Vectors for Entity Alignment on Knowledge Graphs

Master's Thesis March 9, 2018

Leon Ruppen
of Lucerne LU, Switzerland

Student-ID: 11-606-035
leon.ruppen@uzh.ch

Advisor: **Matthias Baumgartner**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to thank my thesis advisor Matthias Baumgartner of DDIS departement at the University of Zurich. The door was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it.

Also, I would like to thank Bibek Paudel of DDIS for providing the ready to use implementation of the classification task used in the thesis.

Finally, I must express my very profound gratitude to the proofreaders. All the profound feedback was especially valuable to make the thesis more concise and comprehensible. Thank you.

Abstract

The linking of correspondent entities between multiple knowledge graphs (KGs) is known as entity alignment. This thesis introduces the embedding-based method *Dependent Learning of Entity Vectors (DELV)* for entity alignment. In an iterative fashion, the method learns a low-dimensional vector representation for the entities in a satellite model in dependence of a pretrained central model. *Word2vec* and *rdf2vec* constitute the basis for the embedding learning process. *DELV* is evaluated on real-world datasets, originating from the three knowledge graphs DBpedia, Wikidata and Freebase. *DELV* outperforms most of its baselines in terms of the *mean rank*, the *hits@1* and *hits@10*. While entity alignment is normally performed on two KGs, this thesis also demonstrates how *DELV* can be efficiently used for alignment of unlimited KGs.

Zusammenfassung

Die Verknüpfung von sich entsprechenden Entitäten zwischen mehreren Wissensgraphen wird als *entity alignment* bezeichnet. Diese Arbeit stellt die Methode *Dependent Learning of Entity Vectors (DELV)* für *entity alignment* vor. In einer iterativen Weise lernt das Verfahren eine niedrigdimensionale Vektordarstellung der Entitäten in einem Satellitenmodell, in Abhängigkeit von einem zentralen Modell. Word2vec und Rdf2vec bilden die Grundlage für das Lernen der Vektordarstellungen. *DELV* wird auf realen Datensätzen, basiernd auf den Wissensgraphen DBpedia, Wikidata und Freebase, ausgewertet. *DELV* übertrifft die meisten Methode für *entity alignment* in Bezug auf *hits@1, hits@10* und den *meanRank*. Während *entity alignment* normalerweise für zwei Wissensgraphen durchgeführt wird, demonstriert diese Arbeit, wie *DELV* auch für die Ausrichtung einer unbegrenzten Anzahl an Wissensgraphen verwendet werden kann.

Table of Contents

1

Introduction

With the standardization of the Resource Description Framework (RDF) [Hayes and Patel-Schneider, 2014], a large number of structured graph databases, also called knowledge graphs (KGs), were created by numerous data providers. Such graphs can be imagined as a set of triples, with each triple consisting of a subject, predicate and object. A predicate links a subject to an object, e.g. “Paris, isCapitalOf, France”. Because each provider models data to his specific needs and applies individual naming conventions, the databases are isolated in principle. If we consider the entity Q762 within Wikidata, which corresponds to /m/04lg6 in Freebase, it translates to the famous artist Leonardo da Vinci in DBpedia. Desirably, queries against those databases should be able to access information from the combination of all databases. Therefore, efforts were undertaken to link them together. The Linking Open Data (LOD) cloud is a collection of databases that are partially aligned. Databases in the LOD cloud can be clustered into three categories: Databases which are isolated (43.89%), peripheral datasets which are linked to some others (48.32%), and central hubs with many associated databases (7.79%). Only 50 links to any other dataset are required for a dataset to be included in the LOD cloud. Furthermore, about half of the linked datasets in the cloud are aligned with no more than two other databases [Schmachtenberg et al., 2014]. This situation is contrary to the idea of linked data, as most databases are only linked indirectly via a mediating database. Querying of linked datasets becomes sensitive to errors in the hubs and inefficient due to heavy load on a handful of databases. Changes in hubs are likely to lead to inconsistencies in dependent databases. The process of linking the entities has become known as entity alignment.

Recently, embeddings of graphs have been studied intensely [Ristoski and Paulheim, 2016, Yan et al., 2017, Ristoski et al., 2017, Zhu et al., 2017, Bordes et al., 2013]. A graph embedding refers to a low-dimensional vector representation of each node. Various applications and methods, where graph embeddings can be used, have been identified and proposed. While most methods use the transE model [Bordes et al., 2013] as their basis for the embedding learning process, others utilize the neural language model word2vec [Mikolov et al., 2013a] for RDF graphs (rdf2vec) [Ristoski and Paulheim, 2016]. TransE constrains the construction of the vector space based on interpreting a relation between entities as translations of vectors. Differently, word2vec and rdf2vec use co-occurrences of word pairs in the training corpus. Both methods lead to specific similarity properties among embedding vectors, which can be used for comparing entities and automated entity alignment.

This thesis answers the following question: How can the machine learning model word2vec be adapted for automated entity alignment on multiple KGs?

A joint graph-embedding mechanism, called Dependent Learning of Entity Vectors (DELV), has been developed. Using word2vec as a basis, DELV is able to perform entity alignment among two or more KGs. Unlike many existing methods [Zhu et al., 2017, Yan et al., 2017, Hao et al., 2016], DELV does not need to train the embeddings simultaneously or restrict them to a unified vector space. In an iterative manner, the embeddings for one KG, the central KG, are calculated first. Then, a second KG, the satellite KG, will be trained in dependence of the central KG. Many advantages follow from this iterative approach: Embeddings for unlimited satellite KGs can be learned against a central KG without the need to retrain the latter. This also allows for entity alignment between the satellites. Moreover, such a setup allows for computation in a distributed and local fashion. This could be used for datasets which cannot be published, e.g. due to licensing reasons, and avoids the need of having all datasets available in memory and at the same time.

The first step of the thesis consists of a summary of relevant publications in the area of entity alignment. Subsequently, the relevant concepts applied in this thesis will be explained in detail, building the theoretical basis for the experiments. In a second step, DELV is presented as a machine learning pipeline, illustrated with pseudo code and further insights into the utilized algorithms. Furthermore, the different test scenarios for DELV are defined. Then, the construction of the three real-world datasets, as well as the resulting graph and link quality is described and assessed. The performance of DELV is presented together with a comparison to selected baselines. Initially, DELV will be tested on two KGs. Afterwards, a third real world KG is included in a second experiment. The thesis concludes with a summary and an outlook.

1.1 Problem Description

The following notations, common to the field of KGs, will be used hereafter:

- **Triple:** A data triple consisting of an object, a relation and a subject, where both object and subject are entities of a knowledge graph. The relation describes how two entities are related to each other. An example is "Paris, isCapitalOf, France".

- **Knowledge Graph (KG):** A knowledge graph consists of the set of entities and relations, forming the triples. The sum of the triples completely describes a knowledge graph.

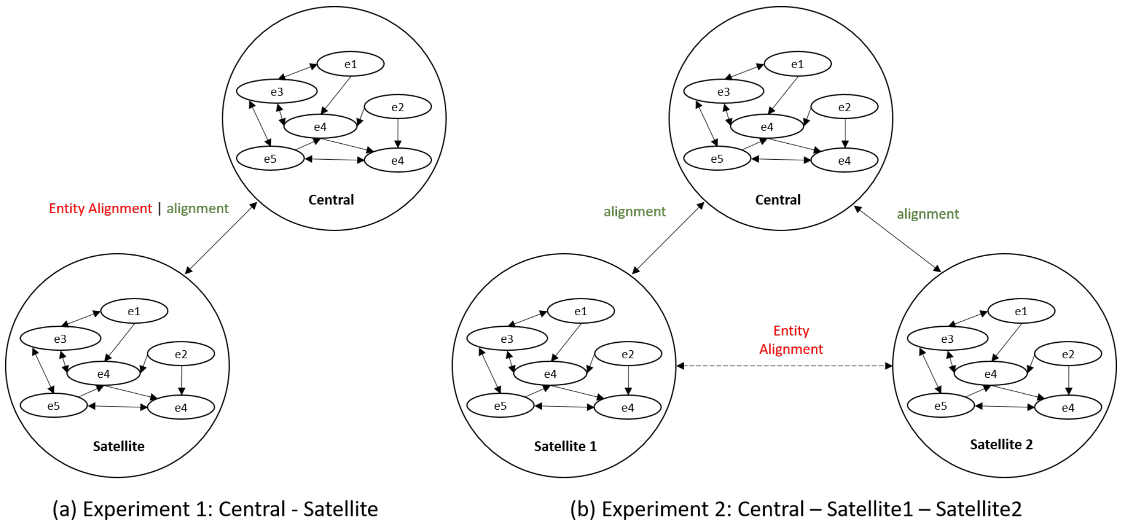

Entity Alignment | alignment
Central
Satellite
e1
e2
e3
e4
e5
(a) Experiment 1: Central - Satellite
alignment
alignment
Central
Entity Alignment
Satellite 1
Satellite 2
(b) Experiment 2: Central – Satellite1 – Satellite2

Figure 1.1: The Experiments

- **Satellite / Central:** The central model is a pretrained rdf2vec model with fixed embeddings for a given KG. A satellite is a single KG, the embeddings for which will be trained in dependency to the central KG, using an initial alignment.
- **Alignment:** A set of linked, corresponding entities e_n and e_m with $e_n \in satellite$ and $e_m \in central$. A portion of the alignments is used as input to the training procedure. The alignment ratio ranges between 0-100% and indicates how many links, relative to the satellite KG size, have been used for training.

The two problems showed in Figure 1.1 are the main focus of this thesis. In the first experiment, a satellite model is trained in dependence to a central model, using an initial alignment. More specifically, the main challenge is to train vector embeddings of a satellite model in dependence to already existing embeddings of the central model, such that the embeddings for correspondent entities in each KG are similar. Afterwards, the entity alignment task between the *satellite* and the *central* is evaluated.

In Experiment 2, a third KG is added as a *satellite*. Both *satellites* are trained in dependence to the central model. But in contrast to Experiment 1, the entity alignment is done between the two *satellites*. Experiment 2 can be seen as an extension of Experiment 1.

The links between all three of the KGs are known, providing the ground truth to check the predictions against. Clearly, a method knowing all the links between *satellite 1/central* and *satellite2/central* can determine the links between *satellite1* and *satellite2* in a transitive fashion. Therefore, the ambition is to use an alignment ratio as small as possible.

2

Literature

This chapter presents the relevant concepts as a basis for the introduction of DELV in Chapter 3. First, the neural language model word2vec is explained, followed by the concept of rdf2vec. Then, literature relating to entity alignment is summarized, and the baselines for this thesis are presented.

2.1 Word2vec

In the field of natural language processing (NLP), a word of a vocabulary was traditionally associated with a vector in a 1-of-N or one-hot encoding. A one-hot vector is a 1xN vector, which consists of 0s in all cells except for a single 1 that identifies the word. Treating words as atomic units is simple and leads to robust approaches. However, several drawbacks can be identified: High dimensionality of the vectors, data sparsity, computational complexity and lack of notion of similarity between the word vectors [Ristoski et al., 2017, Mikolov et al., 2013b]. In contrast, Vector Space Models (VSMs) map semantically similar words to nearby points in a continuous vector space. The basis for those models is the *Distributional Hypothesis*. It states that words in the same context share semantic meaning. Various approaches have been derived from this principle. They can be categorized as count-based methods (e.g., Latent Semantic Analysis (LSA)) or predictive methods (e.g., neural language models) [TensorFlow, 2017].

Word2vec, proposed in [Mikolov et al., 2013a], is a predictive neural language model designed to create continuous vector representations of words. In contrast to one-hot vectors, word2vec uses a distributed representation of words. The meaning of a word is represented with respective weights, distributed among its feature vector. This allows for low-dimensional, dense word vectors [TensorFlow, 2017,Mikolov et al., 2013b]. Although there exist various approaches utilizing continuous and distributed vector representations of words [Collobert and Weston, 2008,Hinton et al., 1986,Mikolov et al., 2009,Rumelhart et al., 1986], word2vec became widely used due to its superior computational efficiency. The next section explains the functionality of this approach.

2.1.1 The Algorithms

The main task of word2vec is to train high quality, distributed word vectors. To this end, two different algorithms, continuous bag of words (CBOW) and skip gram (SG), have been proposed [Mikolov et al., 2013a]. The CBOW architecture is explained first, followed by SG.

Continous Bag of Words (CBOW)

CBOW aims at predicting a target word w_t from c context words:

$$p(w_t \mid w_{t-c}...w_{t+c})$$

The input is a 1-out-of-V vector, with V being the size of the vocabulary. The hidden layer h (compare to Figure 2.1) results from the average of the multiplication of the input vectors x_i with the weight matrix W. As x_i is one-hot encoded, this multiplication corresponds to selecting a row in W. This leads to the following formula for the hidden layer [Rong, 2014]:

$$h = \frac{1}{C} \sum_{i=1}^{c} v_{w_i}$$

where v_{w_i} is $x_i\ W$. Note that a row v_{w_i} of W correspond to the word vector for word i. Intuitively, h is the average of all word vectors of the context words, as illustrated in Figure 2.1.

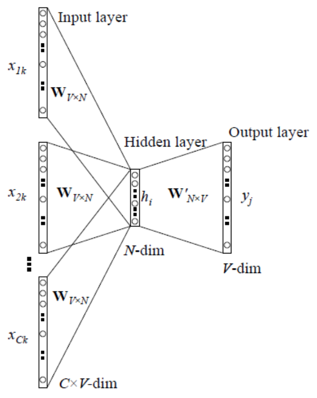


Figure 2.1: Continous Bag of Words (CBOW) [Mikolov et al., 2013a]

To get the output layer, h is multiplied by a different weight matrix W', resulting in a vector U that consists of a score u_j for each word in the vocabulary. Then, those scores are smoothed into a multinomial distribution using the softmax classifier:

$$y_j = p(w_t \mid w_{t-c}...w_{t+c}) = \frac{e^{u_i}}{\sum_{i'=1}^{V} e^{u_{i'}}} \tag{2.1}$$

Intuitively, applying softmax to the output layer results in a probability distribution, predicting for all words j in the vocabulary the probability of seeing word j given the respective context c.

Consequentially, the training objective of the network is to maximize $p(w_O \mid w_I)$, which is the likelihood of seeing the actual true output word w_O (having index j^* in the output layer) given context w_I:

$$max\ p(w_O \mid w_I) = max\ log\ y_{j^*} = u_{j^*} - log \sum_{j'=1}^{V} e^{u_{j'}} := -E \tag{2.2}$$

taking $E = -log\ p(w_O \mid w_I)$ as our loss function. Backpropagation and stochastic gradient descent are used to determine the respective update functions, which can be found in [Rong, 2014].

Skip Gram (SG)

SG can be interpreted as the inversion of CBOW. Instead of predicting a word given its context, the context given a word is estimated (Figure 2.2).

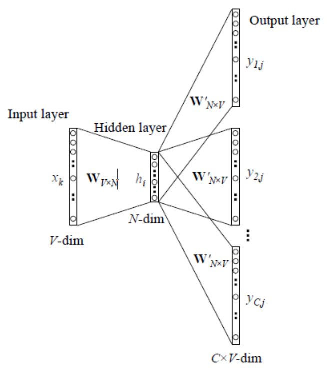


Figure 2.2: Skip Gram (SG) [Mikolov et al., 2013a]

SG takes only one input word and calculates c multinomial distribution, also using the softmax classifier [Rong, 2014]:

$$p(w_{c,j} = w_{O,c} \mid w_I) = y_{c,j} = \frac{e^{u_{c,j}}}{\sum_{j'=1}^{V} e^{u_{j'}}} \tag{2.3}$$

with $w_{c,j}$ being the c-th word of the context in the output, $w_{O,c}$ the c-th target word and w_I the only input to the SG algorithm. The resulting update equations are derived and explained in greater detail in [Rong, 2014].

Intuitively, the update equations for the word vectors can be understood as adding small parts of all output layer vectors, weighted by their prediction error $y_j - t_j$, to the respective word vector w_i. In case of overestimating the probability of word w_j

being the correct word for given input ($y_j > t_j$), the word vector moves away from the output vector w'_j. In case of underestimation, the same word vector moves closer to w'_j [Goldberg and Levy, 2014, Rong, 2014].

2.1.2 The Optimizations

Looking at Equation 2.1 and 2.3, it becomes clear that the softmax classifier requires computationally laborious update equations for the output layer vectors. For each training sample, the algorithm needs to iterate over the entire vocabulary to calculate the prediction error for each word. This quickly becomes infeasible for the massive datasets word2vec is trained on. Therefore, two independent modifications have been presented in [Mikolov et al., 2013a]. In order to design the algorithm in a more efficient way, the evident approach is to restrict the number of output vectors that need to be updated per training iteration. One can distinguish softmax- and sampling-based methods. While softmax-based approaches only modify the architecture of the softmax layer, sampling-based methods substitute the softmax loss function completely. The two solutions used in word2vec, the softmax-based Hierarchical Softmax (HS) and sampling-based Negative Sampling (NS) [Ruder, 2016], are presented hereinafter. The aim will be to illustrate these techniques, rather than explaining the mathematics. Minor optimizations used in word2vec, such as subsampling of frequent words or common word pairs and phrases detection, will not be discussed.

Hierarchical Softmax (HS)

The idea of hierarchical softmax is to substitute the flat softmax output layer with a hierarchical one. HS uses a binary tree having the words as leaves. Recalling that the main problem is to normalize over every word in the vocabulary, HS allows for a calculation of the already normalized probability by following the respective path to the leaf node. Note that the probabilities in the leaves are necessarily normalized in a binary tree [Ruder, 2016]. Therefore, to calculate the probability of a word given its context, the probabilities along the path to the word are multiplied. The following example illustrates the idea of HS: Suppose word2vec with CBOW with HS and the training sequence ['the', 'dog', 'is', 'gone']. 'gone' is the target word and 'the', 'dog' and 'is' form the context (Figure 2.3).

First, the input vectors for the three context words are averaged, raising h as the hidden layer vector. To calculate the probability of the target word given its context, one multiplies the probabilities along the path (marked in red in Figure 2.3) to the target word:

$$p(gone \mid context) = p(left\ at\ 1 \mid c)\ p(right\ at\ 2 \mid c) = (1 - sig(v'_1 h))\ sig(v'_2 h)$$

with sig being the sigmoid function $sig(x) = \frac{e^x}{e^x+1}$ and v'_n the vector representation v' of the inner unit at node n. As lookups in a balanced binary tree need at most $log\ V$

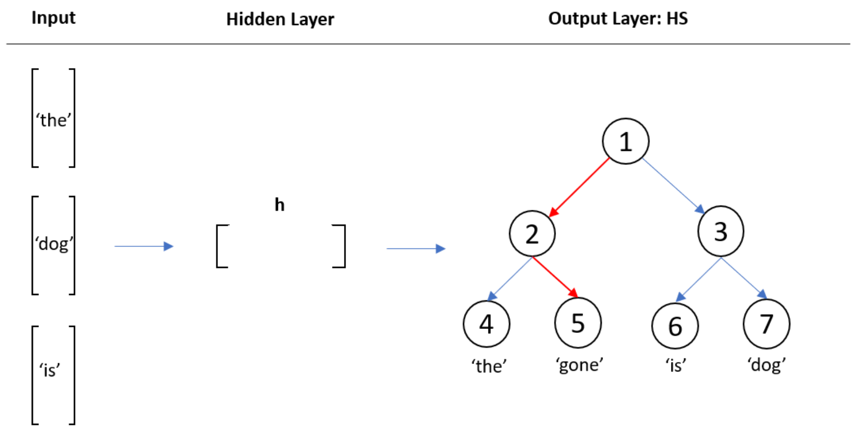


Figure 2.3: Hierarchical Softmax (HS)

evaluations, HS reduces computational complexity per training iteration from $O(V)$, in case of softmax, to $O(log\ V)$.

Negative Sampling (NS)

The basic idea of NS is to reduce the number of output vectors to update per training iteration. The goal is to update a sample only. While the correct output word, called 'positive sample', is always part of the sample, other words, called 'negative samples', need to be chosen from the vocabulary, according to a probability distribution. In [Mikolov et al., 2013b], the authors empirically determine a unigram distribution raised to the power of $\frac{3}{4}$ to be optimal. NS can be understood as an approximation to Noise Contrastive Estimation (NSE) [Gutmann and Hyvärinen, 2010]. While NSE can be shown to approximate the regular softmax as the number of negative samples k increases, NS does not support this guarantee. It only aims at producing high quality word vectors [Ruder, 2016]. NS takes NSE as a basis and simplifies it to produce a loss function, which only needs to be applied for $w_j \in W_{neg}$ and the positive sample, opposed to every word in the vocabulary. A detailed derivation of this loss function is found in [Ruder, 2016] or [Goldberg and Levy, 2014].

Word2Vec was successfully trained on a Google News data set with around 100 billion words and a vocabulary of 3 million words. The resulting vectors outperform word vectors produced by traditional methods, e.g. LSA, on several test sets [Mikolov et al., 2013a, Mikolov et al., 2013b].
Having laid out the two architectures and the main optimization techniques of word2vec, the next section focuses on the second important framework, called rdf2vec.

2.2 Rdf2vec

Rdf2vec is an approach “that uses language modelling approaches for unsupervised feature extraction from sequences of words, and adapts them to RDF graphs” [Ristoski and Paulheim, 2016]. While “language modelling approaches for unsupervised feature extraction [...]” refers to the described word2vec, the adaption to the RDF graphs is illustrated in Figure 2.4. It is noteworthy that rdf2vec is based on and closely related to [Yanardag and Vishwanathan, 2015, Perozzi et al., 2014].

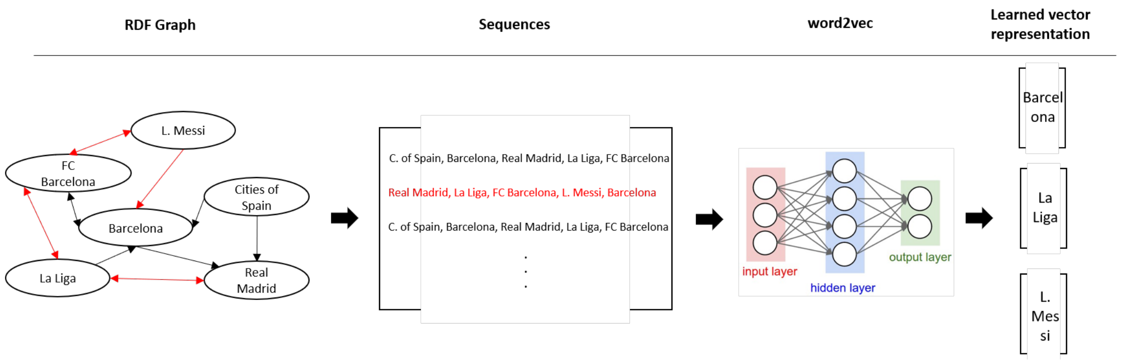

RDF Graph
Sequences
word2vec
Learned vector representation
L. Messi
FC Barcelona
Cities of Spain
Barcelona
La Liga
Real Madrid
C. of Spain, Barcelona, Real Madrid, La Liga, FC Barcelona
Real Madrid, La Liga, FC Barcelona, L. Messi, Barcelona
C. of Spain, Barcelona, Real Madrid, La Liga, FC Barcelona
input layer
hidden layer
output layer
Barcelona
La Liga
L. Messi

Figure 2.4: The Rdf2vec Approach

Given a RDF graph, an algorithm extracts sequences of connected nodes in the graph (see Figure 2.4). Then, instead of having sentences as input for word2vec, rdf2vec uses those sequences of node labels as input for word2vec. In turn, word2vec yields the word vector representations for every input node.
The question of how to extract those sequences is essential. Two different approaches, *Graph Walks* and *Weisfeiler-Lehman Subtree RDF Graph Kernels (WLGK)*, have been proposed for sequence extraction [Ristoski et al., 2017].

Graph Walks: Utilizing the breadth-first algorithm, *Graph Walks* produces sequences by exploring all connections for vertices v in a graph g. Given vertex v_i, *Graph Walks* recursively explores all edges e_i and connected vertices. As it is impossible to generate all walks on RDF graphs due to loops, breadth first is depth limited and becomes *Depth-Limited Search*, with depth d being a hyperparameter of the model. Furthermore, [Ristoski et al., 2017] remarks that with increasing size of the graph, “calculating [...] all graph walks with a given depth d for all the entities in the large RDF graph quickly becomes unmanageable”. This introduces a second hyperparameter *walksPerNode.*

Weisfeiler-Lehman Subtree RDF Graph Kernels (WLGK): This state-of-the-art kernel for graph comparison, introduced in [de Vries, 2013], “computes the number of sub-trees shared between two (or more) graphs by using Weisfeiler-Lehman test of graph isomorphism”. Furthermore, in [Ristoski et al., 2017] two modifications to enable the application of the algorithm on RDF graphs are added, yielding *WLGK*.

Although superior results for classification tasks (Naive Bayes, k-Nearest Neighbours) and regressions have been reported using *WLGK*, it is only feasible for smaller datasets due to its increased computational complexity. Rdf2vec with *Graph Walks* performs closely to the standard graph substructure feature generation strategies as presented in [Ristoski et al., 2017].

Nevertheless, rdf2vec models with *Graph Walks* trained on huge knowledge graphs, such as DBpedia or Wikidata, consistently outperform standard feature generation (e.g., TransE, TransH or TransR) in classification, document similarity and other common NLP tasks [Ristoski et al., 2017].

2.3 Related Work and Baselines

A broad range of approaches and applications regarding entity alignment have been proposed in the last decade. While earlier methods were content-based [Suchanek et al., 2011, Lacoste-Julien et al., 2013, Sun et al., 2017] and sometimes involved human feedback [Hassanzadeh and Consens, 2009], recent approaches are embedding-based and try to represent different KGs in a way that similar entities are similarly encoded. Most papers apply translation-based models for embedding generations, e.g. TransE [Bordes et al., 2013]. [Chen et al., 2017, Sun et al., 2017] apply a modified TransE framework to perform cross-lingual entity matching. [Guan et al., 2017] additionally introduces an iterative technique and differentiates between relations and attributes while training the embeddings. In connection with the Ontology Alignment Evaluation Initiative (OAEL) , many different ontology matching pipelines have been introduced, as described in [Achichi et al., 2016, Cheatham et al., 2015]. As the mentioned papers focus on different subproblems (e.g., cross-lingual, content-based approaches) within the area of entity alignment, it is not meaningful to compare the results of this thesis against their performance.

Conversely, the next three papers introduce comparable methods, and are therefore explained in more detail. All of these methods apply the translation-based method TransE as a basis for embedding learning. TransE interprets a relation between two entities as a constant translation between the corresponding entity vectors. Therefore, the main task is to find the best possible solutions for the equation $\vec{s} + \vec{r} = \vec{o}$ for all triples in the KG, with $\vec{s}$, $\vec{r}$ and $\vec{o}$ being the vector representation of the subject, the relation and the object of a triple.

In [Hao et al., 2016], a system for entity alignment, called *JEwP*, is proposed that is based on the structure of the KG. This makes it independent of the content. Given an alignment seed, the embeddings of two KGs are jointly learned in the same vector space. The initial alignment imposes a constraint on the TransE training process and enforces equalization of the correspondent entity embeddings over the training process. This serves as a bridge between two KGs, partly unifying them. As a consequence, embeddings can be trained in the same vector space and similar entities end up with similar embeddings. The authors claim that this is the first approach leveraging structural information only to perform entity alignment with a joint embedding model. The method is tested on two different data sets. The first dataset is based on the well-known benchmark dataset *FB15K* [Bordes et al., 2013]. As illustrated in Figure 2.5, the triples of *FB15K* are split randomly into two sets “[...] with a large amount of overlap”. Then, after splitting the intersecting entities into two parts, the first part can be used as an alignment seed. The second part is used in the evaluation. It is noteworthy that this approach greatly simplifies the entity alignment problem. The entities to link are exactly the same entities, generated according to the same design principles of Freebase and encode the same information. This is not the case if entity alignment is done between different KGs. The second dataset, called *DB_FB*, is based on the two KGs DBpedia (DB) and Freebase (FB). Based on a set of filtering rules, two graphs with 57’076 entities for DB and 19’166 entities for FB are created. While it is interesting to test the method on the simplified dataset, the second dataset *DB_FB* can be considered as a more realistic case.

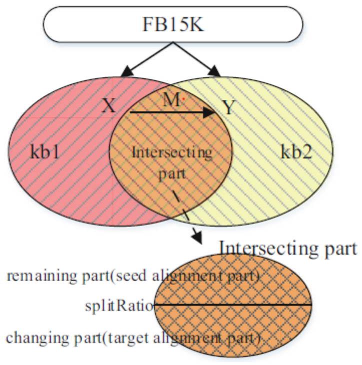

FB15K
X
M
Y
kb1
Intersecting part
kb2
Intersecting part
remaining part(seed alignment part)
splitRatio
changing part(target alignment part)

Figure 2.5: Construction of Datasets with an Overlap [Hao et al., 2016]

A similar approach, called *Iterative Entity Alignment via Knowledge Embeddings (IEAKE)*, is used in [Zhu et al., 2017]. Also using TransE as a basis, embeddings for entities and relations are jointly trained for two KGs with an alignment seed. Furthermore, an optimization strategy called soft alignment is introduced. After a fixed amount of training iterations, for all non-aligned entities in the first KG, the most similar entity in the second KG is calculated. If the difference between the entities is smaller than a certain confidence score, the entities will be considered aligned for the rest of the training. Therefore, the method can continuously update and find new entity alignments whilst training. *IEAKE* is tested on a dataset constructed according to the same principles as the first dataset of [Hao et al., 2016], described above. It was not tested on datasets that are based on two different KGs.

[Cai et al., 2017] states that the performance for existing methods drop when tested on sparse graphs, compared to dense graphs. Furthermore, the authors criticise that most methods are only tested on a single KG, using *FB15K* as basis. Therefore, they propose a cross-KG knowledge representation learning (KRL) method, *Cross-TransSparse*, which can be applied to sparse graphs across multiple KGs. The main difference to other approaches is a mechanism that improves the embeddings of the sparse graph by using the rich structural knowledge of the dense graph. The method has been tested on cross language datasets of DBpedia and on datasets that are based on FB and DB.

All three methods, i.e. *IEAKE, JEwP* and *Cross-TransSparse*, are comparable to the approach of this thesis. Although they calculate the embeddings for the two KGs simultaneously and in a unified vector space, they are still embedding-based approaches like the method presented in this thesis. They will be used as baselines when evaluating the performance of *DELV*.

3

Method

As suggested in the introduction, this thesis is about testing how entities and their counterparts among multiple KGs can be linked. Therefore, the method Dependent Learning of Entity Vectors (DELV), explained in the the next section, was developped.

3.1 Dependent Learning of Entity Vectors (DELV)

For the training of the desired word vectors, rdf2vec and a modified word2vec will be used. While the purpose of rdf2vec, transforming a KG into a set on sequences, is straightforward, the modification of word2vec needs further explanation.

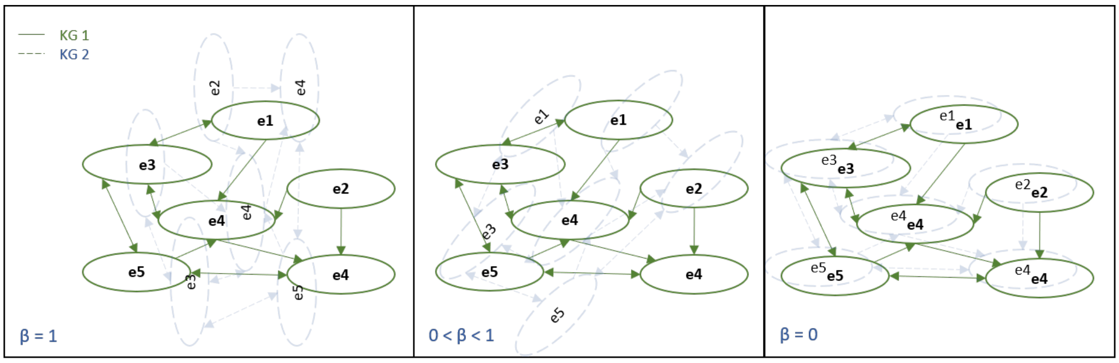


Figure 3.1: Visualization of β

What information word2vec encodes in its embeddings is diffuse and can be circumscribed in various ways: “Word vector representations capture many linguistic properties such as gender, tense, plurality and even semantic concepts like ‘capital city of’.” [Heuer, 2016] or “[...] these monolingual embeddings have been shown to encode syntactic and semantic relationships between language elements.” [Jansen, 2017]. For the present purpose, the important information, that word2vec needs to capture, is the semantic information of the input sentences. Semantic information refers to the arrangement of words and phrases in a sentence. Applying this idea to a KG, the semantic information equates to the structure of a KG and, therefore, the relationship among entities. Because the word vectors depend on the co-occurrence of word pairs, for two corresponding embeddings of different KGs to be similar, their surrounding entities and the structure between them must be similar. But due to random initialization, even for identical KGs, two runs of word2vec do not produce the same entity embeddings. Therefore, a number of entities of the satellite, the initial alignments, need to be trained against the fixed embeddings of their respective central entity embedding. Then, the rest of the entities can adjust around those fixed entity vectors, such that word2vec produces similar and comparable vectors for similar entities.

This argumentation implies that the success of the method is dependent on the similarity of the structure of the KG. And as different KGs often model the same, real world information, a similar structure for different KGs can be expected. But differences exist, as they are built according to different design principles, priorities, wealth of data and goals.

For the dependent training of the satellite, some modifications to word2vec have been made. The goal is to train those satellite vectors, which are part of the alignment, against their correspondent and fixed entity vector in the central model. In practice, this imposes a constraint on the original loss function of word2vec. The new loss function, formulated in the skip gram with negative sampling configuration (see Section 2.1), is:

$$\begin{aligned} min\ E = -log\ \sigma(v'_{w_O} h) - \sum_{w_j\ \in\ W_{neg}} log\ \sigma(-v'_{w_j} h) \\ s.t.\ \frac{dot(v^S_w, v^G_w)}{\|\ v^S_w\ \| * \|\ v^G_w\ \|} \geq (1 - adjustFactor)\ \forall\ (v^S_w, v^G_w) \in\ Alignment \end{aligned} \tag{3.1}$$

with v^S_w being the satellite vector trained against the correspondent goal vector v^G_w. The loss function stays the same as only the constraint is added. The cosine distance, defined as $cosdistance(x, y) = \frac{dot(x,y)}{||x||*||y||}$, is used as the similarity measure for comparing whether two vectors are similar. Therefore, two vectors are considered the same if their cosine distance is (close to) 1. But two corresponding entities in different knowledge bases can differ and should therefore not be treated as the same. Accordingly, an adjustment factor is included which allows the vectors to capture subtle differences while training. As it is practically impossible to determine the value of such a factor for each entity pair between different KGs, the adjustment factor is indirectly implemented by the parameters λ and β, as shown below. Given the new loss function, the new update equation for the hidden layer becomes a split function:

$$v^{new}_{W_I} = v^{old}_{W_I} + (\beta g_{W_I} + (1 - \beta)\ \lambda\ (v^{goal}_{W_I} - v^{old}_{W_I}))$$

with g_{W_I} being the gradient calculated through error back propagation. The new update equation introduces the two new parameters $\lambda\ \in\ [0, 1]$ and $\beta\ \in\ (0, 1]$. Depending on the values of the parameters, this update function allows the satellite vectors to deviate from their correspondent central vectors. While λ scales down the absolute value of the difference of the vectors, β allows the word vector to be a weighted sum of the standard gradient from the model error and the difference in the vectors. For $\beta = 1$ the satellite will be trained without adjustment to a central vector, while $\beta = 0$ is equal to directly initializing $v_{W_I}^{satellite} = v_{W_I}^{central}$ for $w_I \in Alignment$. λ should be used to scale the goal error to a similar magnitude as the gradient. As said gradient is scaled by the word2vec learning rate α, λ is set equal to the α.

Figure 3.1 provides an intuitive visualization of the meaning of β. Suppose two identical KGs which have the same structure and are completely linked. KG1 is the central and KG2 the satellite model. If no alignments ($\beta = 1$) are used, DELV produces completely independent word vectors (symbolized by perpendicular entities). $\beta = 0$ perfectly aligns the entities and $0 < \beta < 1$ produces similar vectors up to a certain degree. Notice that both parameters also affect how fast an aligned entity vector adjusts to its goal vector during the training process. How DELV is implemented in detail is shown in the next section.

3.2 Implementation and Pipeline

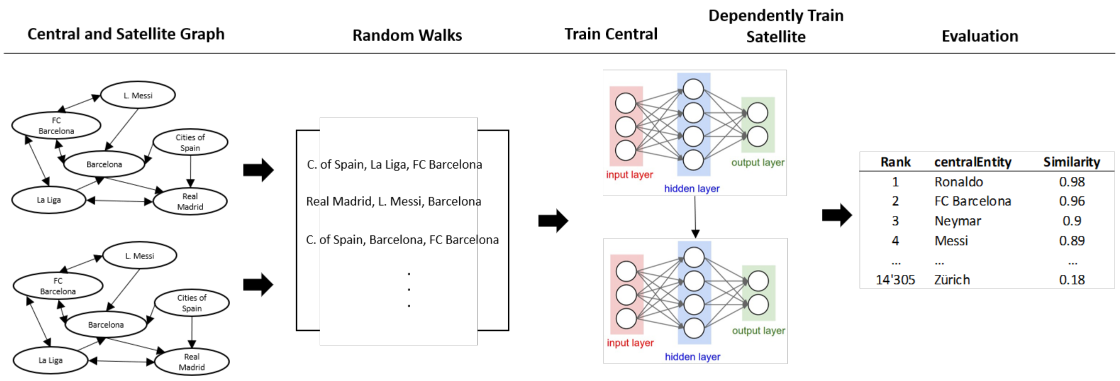
Figure 3.2 illustrates the machine learning pipeline:

1. Create random walks for the satellite and central graph separately: randomWalks(depth, walksPerNode)
2. Use the random walks for the central KG to train the central with word2vec: w2c(randWalks_Cent)
3. Use the satellite random walks and the alignments to dependently train the satellite embeddings with the modified word2vec: w2c_modified(randWalks_Sat, alignment ratio, embedings_central, β, λ)
4. Evaluate the entity alignment between the central and the satellite.

Below, the implementation and the parameters for each step are discussed and illustrated with pseudo code where applicable. While the random walk generation is implemented using Java, the other steps are written in Python.

3.2.1 Graph Walks

To create the random walks, the algorithm *Depth-Limited Search (DLS)*, similar to [Ristoski et al., 2017] and as stated in Figure 3.3, is used. *DLS* recursively creates *walksPerNode* many random walks with a depth d for a given entity for each entity in the KG. The depth does not include relations, e.g. the depth for the sequence "e_1, r_1, e_2, r_2, e_3" equals 3. The output is a set of random walks, which looks like $[e_1, e_2, ..., e_d]$. As there can be dead ends (entities which do not have an outgoing relation), a random walk is only included in the output if its depth is bigger than a *minimalDepth*, which is fixed at 3 for all experiments. The method *isValidVertex* allows for any kind of filtering, such



C. of Spain, La Liga, FC Barcelona

Real Madrid, L. Messi, Barcelona

C. of Spain, Barcelona, FC Barcelona

Rank	centralEntity	Similarity
1	Ronaldo	0.98
2	FC Barcelona	0.96
3	Neymar	0.9
4	Messi	0.89
...	...	...
14'305	Zürich	0.18

Figure 3.2: Machine Learning Pipeline

as ignoring string literals and data properties. The sequences produced are stored in a text file for further usage as input to word2vec.

Instead of *Graph Walks*, which randomly produces sequences, *WLGK*, introduced in [Ristoski and Paulheim, 2016] and described in Section 2.2, could have been used. Superior results are reported for *WLGK* compared to *Graph Walks*. The problem with *WLGK* is that it cannot be applied to big KGs. But the aim of this thesis is to produce a method, which is scalable and can be applied to entire KGs. Therefore, *Graph Walks* was preferred over *WLGK*.

3.2.2 Word2Vec

For word2vec, the well-known gensim implementation [Rehurek and Sojka, 2010] is used. As input, word2vec expects a set of sentences which will be the set of random walks. Multiple parameters need to be chosen when running word2vec. The most important ones are discussed below:

- **Size:** Refers to the size of the hidden layer. While bigger values for the size can lead to more accurate results, they also increase the training time. Values in the small hundreds are generally considered reasonable [Ristoski and Paulheim, 2016, Mikolov et al., 2013b].

- **Window:** Refers to the context size when processing word pairs in a sentence. E.g., using the sentence "a b c d e f g", a window size of 1 and "d" as the target word would yield the training pairs [dc,de], while a window size of 2 yields [db, dc, de, df]. Notice that the effective window is randomly chosen between 1 and *window*, which simulates the increased importance of words closer to the target. A window size of 5 for the English language is common [Mikolov et al., 2013a]. But compared to natural languages, the random walks encode information especially dense, as they do not include words like conjunctions or numbers. Therefore, a window size smaller than 5 should be used.

```
Data: G = (V,E): RDF Graph, d: walk depth, w: walksPerVertex, min: minDepth = 3
Result: randWalks: Set of random walks

for vertex v ele V:
    for counter = 0 To w:
        seq = walk Sequence
        seq = createWalk(v)
        if len(seq) >= minDepth:
            randWalks.add(seq)
        end

method createWalk(v):
    seq = walkSequence
    seq.add(v)

    if seq.hasReachedDepth or isDeadEnd(seq):
        return seq
    end

    nextVertex = None

    do:
        reachableVertices = getAllPossibleVertices(v)
        nextVertex = randVertex(reachableVertices)
    while(isValidVertex(nextVertex)

    seq.add(nextVertex)

    return createWalk(nextVertex)
```

Figure 3.3: Code Snippet: Graph Walks

```
# If currently trained entity is part of the alignment
if entity in alignment:
    goalVector = goalVectors[entity]

    entity_wordVector += beta * err_smax + (1 - beta) (lambda * (goalVector - entity_wordVector))

# If currently trained entity is NOT part of the alignment
else:
    entity_wordVector += err_smax
```

Figure 3.4: Code Snippet: Dependent Training of the Satellite Model

- **SG:** Indicates whether to use *SG* or *CBOW* as the training algorithm (see Section 2.1.1).
- **Negative:** Refers to the number of negative samples to use when using the negative sampling optimization technique. For large datasets a value of 2-5 is sufficient while small datasets need between 5 and 20 negative samples [Mikolov et al., 2013a].
- **Min Count**: Refers to the minimum number of times a word needs to appear in the corpus in order not to get sorted out. As multiple walks are generated for each entity, an entity appears at least *walksPerNode* many times in the corpus. Therefore, the value of min count is of no importance given that it is not bigger than *walksPerNode*.

The word2vec parameters are extensively documented in the literature and will not be discussed further.

3.2.3 Dependent Training of the Satellite Embeddings

The implementation of the dependent training process can be condensed to the following code snippet (Figure 3.4). It shows where the word vector for an entity gets adjusted. The 'if' statement implements the split function and the respective update equations introduced in Section 3.1.

3.2.4 Evaluation

The main purpose of the entity alignment task is to state how well the entities of the satellite model are linked to the central model. Only those entities will be considered, which were not trained against a goal entity embedding, i.e. $\forall\ entity\ e \notin Alignment$. The three metrics *mean rank, hits@1* and *hits@10* are measured. To calculate these metrics, the cosine distance to all central entities is calculated for each entity in the satellite, and sorted according to their distance. The first entry of the matrix is the corresponding central entity as predicted by the model. If this prediction is correct, it is a hit@1. Similarly, if the true correspondent entity is in the top ten of the sorted matrix, it is a hit@10. The rank denotes the general position of the true link within the matrix and lies between 1, the best possible rank, and the number of words in the central vocabulary. The mean rank is calculated over all the evaluated entities. Furthermore, additional statistics for the rank, such as standard deviation, median, min and max, are considered to gain insights about their distribution. Moreover, the mean rank is compared with a random estimator which is equal to $\frac{\#entities\_central}{2}$.

4

Experiments and Results

In this chapter, the findings of a parameter study will be discussed after explaining the creation of the datasets. Subsequently, the results[1] of the two experiments Central-Satellite and Central-Satellite-Satellite as well as the comparison with IEAKE are presented.

4.1 Datasets

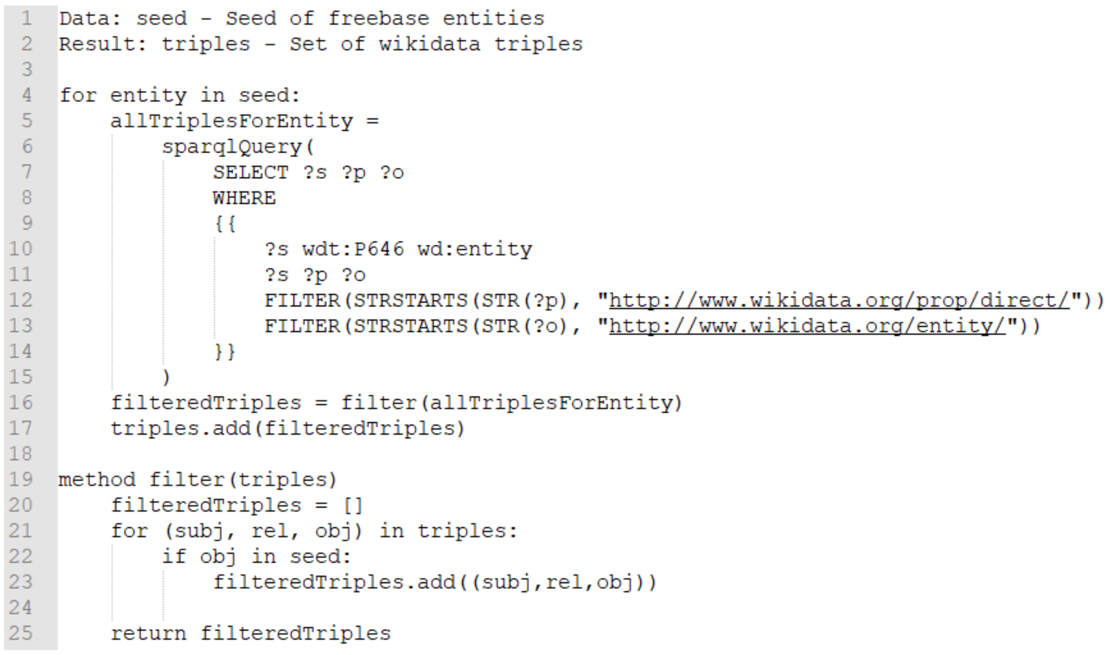
Three different KGs are needed to carry out the experiments. Furthermore, they all have to be interlinked in order to provide a ground truth to check the link predictions against. The three KGs DBedia (DB) [Auer et al., 2007], Wikidata (WD) [Vrandečić and Krötzsch, 2014] and Freebase (FB) [Bollacker et al., 2008] were selected. Performing tests on the whole KGs is not feasible at an early stage of the experiment due to the enormous size of the KGs. Therefore, the experiments are tested on a subset of the respective KGs. As a basis, the benchmark dataset *FB15K* [Bordes et al., 2013], containing 16296 entities, 1345 relations and 483142 triples, is selected. Using the SPARQL interface of WD (https://query.wikidata.org/) and DB (https://dbpedia.org/sparql), corresponding datasets were created, which meet the following criteria:

1. The three databases contain a similar amount of entities.
2. The entities need to be aligned.
3. The resulting graphs for each database must be connected, such that graph walks of at least depth five are possible.

Consequently, a list of all entities in *FB15K* was extracted. For each entity in the list, a SPARQL query was sent to the WD SPARQL interface, which returns the alignment between FB and WD, and all available triples containing the entity as a subject. The results are filtered such that only triples remain that have an object which also occurs in *FB15K*. Figure 4.1 illustrates this idea for WD. Note that P646 in Figure 4.1 denotes the

[1]All experiments in this chapter were run on a cluster of machines, that contain 24 CPUs and 64 GB memory each. Depending on the training data, a satellite model took between 4 and 22 hours to be trained.

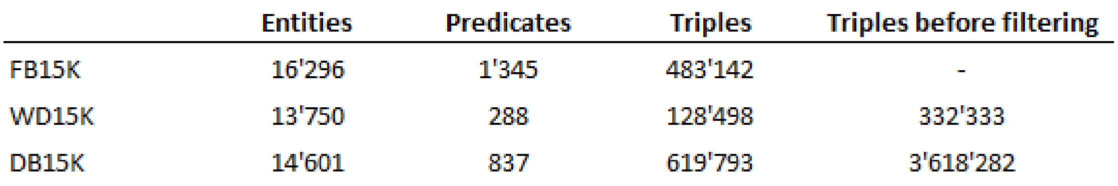
sameAs property between FB and WD. The two filter methods within the SQL query require the subject and the object of the triples to be an entity. Therefore, datatype properties and string literals are filtered.



```
Data: seed - Seed of freebase entities
Result: triples - Set of wikidata triples

for entity in seed:
    allTriplesForEntity =
        sparqlQuery(
            SELECT ?s ?p ?o
            WHERE
            {{
                ?s wdt:P646 wd:entity
                ?s ?p ?o
                FILTER(STRSTARTS(STR(?p), "http://www.wikidata.org/prop/direct/"))
                FILTER(STRSTARTS(STR(?o), "http://www.wikidata.org/entity/"))
            }}
        )
    filteredTriples = filter(allTriplesForEntity)
    triples.add(filteredTriples)

method filter(triples)
    filteredTriples = []
    for (subj, rel, obj) in triples:
        if obj in seed:
            filteredTriples.add((subj,rel,obj))

    return filteredTriples
```

Figure 4.1: Code Snippet: Creation of the Datasets

In the same fashion, the DB subset has been created. The resulting databases are called WD15K and DB15K. Figure 4.2 describes the statistics of the created subsets and FB15K. The number of entities differ because certain freebase entities are not linked to any entity in the other KGs. Furthermore, there were a number of 1:N mappings between FB and WD, which were discarded. The difference in the number of triples is surprising. Partly, this is explainable due to the lower number of entities in *WD15K* (-15% compared to *FB15K*). As a triple gets discarded if its object is not in the linked entity list, fewer linked entities exert a double effect on the number of triples. First, all triples with the unlinked entity as subject are excluded. Second, all triples having the unlinked entity as object are filtered. Moreover, the big number of DB triples before filtering is due to the triples linking entities to pictures and other medias, as well as categories. While the DB SPARQL interface returns such triples, the WD SPARQL interface does not. Despite the difference in the number of triples, the link quality is good. 13725 links are available between FB and DB, 13743 between FB and WD and 13767 between DB and WD.



	Entities	Predicates	Triples	Triples before filtering
FB15K	16'296	1'345	483'142	-
WD15K	13'750	288	128'498	332'333
DB15K	14'601	837	619'793	3'618'282

Figure 4.2: Characteristics of the Datasets

4.2 Parameter Study

To determine values for the word2vec hyperparameters and the newly introduced parameters β and λ, a limited parameter study was carried out, using WD as central model and DB and FB as satellites. Note that the resulting parameters are likely not optimal as the parameter study is not complete and would need to be carried out for each different central model separately. The goal of the study was to get an idea of the behaviour of DELV. Furthermore, the combination of twelve parameters, some of them being continuous, would require considerably more computational power. Otherwise, the optimal parameters could be determined using a gradient-based optimization technique. As SG and negative sampling generally yields better results [Ristoski and Paulheim, 2016] than $CBOW$ and hierarchical softmax, SG with negative sampling will be used for the experiments. The following parameters have been explored: walksPerNode, depth, beta, layer size and negative samples. Recommended values from the literature, as described in Chapter 3, were chosen for the other parameters. A total of 3360 experiments were evaluated over the satellite model DB and FB.

Figure 4.3 shows the results as boxplots for the pair central WD and satellite DB. The results for central WD and satellite FB can be found in Appendix A. While the x axis shows the parameter of interest, the y axis plots the improvement of the achieved rank over the random estimator in percent. E.g., if the experiment states an achieved mean rank of 50, and we expect a random linking to achieve a mean rank of 100, the rank improvement is 50%. For each value of the parameters, the boxplot shows the averaged results over all other combinations of parameters. With averages and medians over 90% for all tests, the results are a positive indication for the quality of DELV.

Clearly, *walksPerNode* introduces the biggest variance in the results. All characteristics improve with rising *walksPerNode*. As the graph walks are created randomly, a higher *walksPerNode* increases the probability that the similarity of structures, occurring in the different KGs, manifests itself in the random walks. Therefore, the embeddings move closer to each other, netting better results with lower volatility. The value of 70 *walksPerNode* are chosen for further experiments.

The depth of the random walks also improves the results. Given a sufficient window size, the depth effectively increases the number of word pairs trained as the length of an

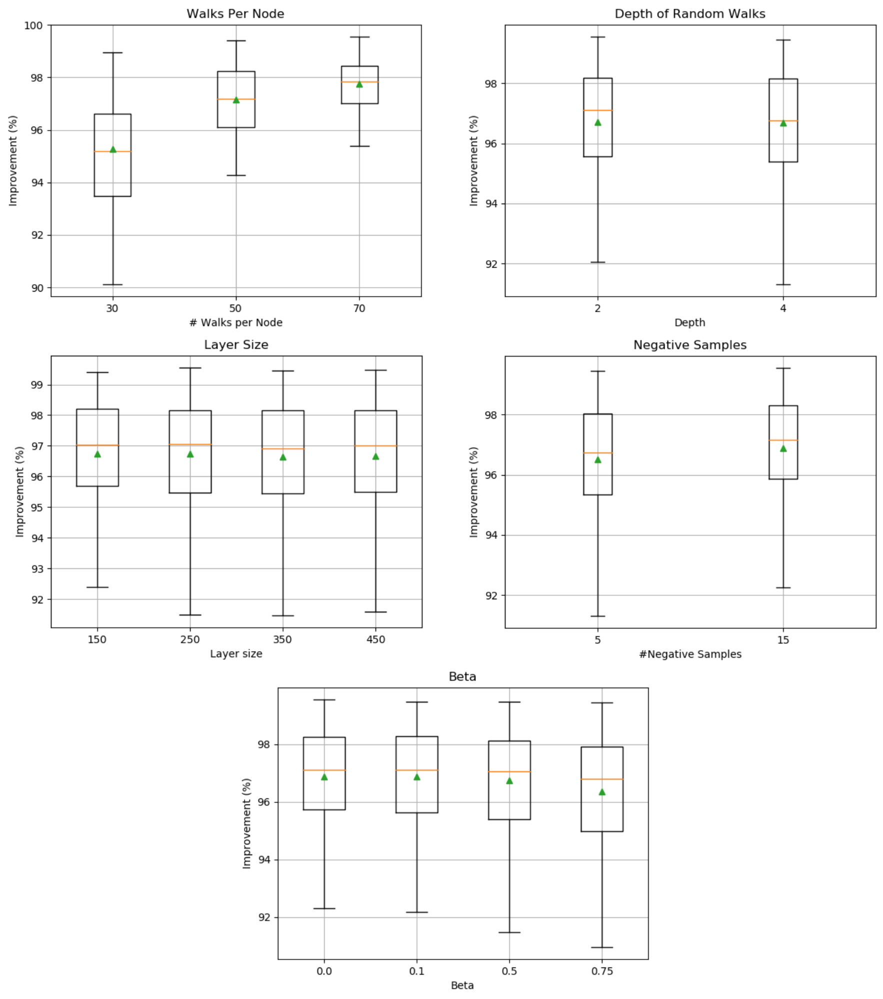


Figure 4.3: Results of the Parameter Study.

input sequence to word2vec is prolonged. On the one hand, this should produce finer word embeddings catching subtler differences in entities. On the other hand, it also reduces the probability of same structures being represented similarly in the random walks. A higher depth should go hand in hand with a higher *walksPerNode*. A depth of 4, effectively producing random walks of length 5, is chosen. Higher values for depth were planned to be tested. But due to the greatly increased running time, those evaluations needed to be aborted.

The results are robust with respect to the layer size. Layer sizes +/- 50 from the recommended values have been tested. The layer size becomes more important with more entities and training data, as more features are needed to capture the differences between the entities. Compared to word2vec models trained on corpuses of billions of words, this experiment has a comparable small corpus in the order of millions of words.

Because the layer size increases the training time, a small value of 150 is chosen.

As a small improvement can be observed with a rising number of negative samples, the value of 15 is chosen. Due to time constraints, no more values could be tested.

Because β is a newly introduced parameter, it will be discussed in more detail than the other parameters. As described in Section 1.1, β is a weight which determines if and how fast the satellite entity vectors get adjusted to their correspondent central vectors. It weights the error err_{smax}, originating from the normal word2vec softmax backpropagation, and the error from the difference between the satellite vector and central goal vector (err_{goal}). $\beta = 0$ is equal to copying the central vectors $\in alignment$ at the beginning of the training, while $\beta = 1$ does not use any information at all from the central model. While averages and median do not vary much, the standard deviation clearly worsens with increasing β. Looking at the norm of differences between the satellite vectors and their respective central goal vectors, even high β result in a minimal difference. This is explained by the fact that err_{goal} always points in the direction of the fixed goal, whereas the direction of err_{smax} changes depending on the training sample. Therefore, over thousands of training iterations, the satellite vector will adjust to the value of the goal vector, thereby producing similar results. But if the diminishing learning rate is considered, β makes a difference in the speed of the adjustment. In the end, the β with the best performance, $\beta = 0.1$, is chosen.
The results of the parameter study are robust with respect to other satellite models. Due to the explanations in Chapter 3 and the evidences above, the feasible parameter set chosen for the experiments is:

- walksPerNode = 70
- depth = 4
- layer size = 150
- window = 2
- min count = 1
- epoch = 5
- negative = 15
- $\beta = 0.1$

Furthermore, skip gram with negative sampling is used for all experiments. After having fixed a set of parameters, the results of the experiments are presented next.

4.3 Results

This chapter presents the results and comparisons of the experiments conducted. It is divided into three sections. First, the results of the experiment 1 are presented, in which a single satellite is trained against a central model. Second, the performance of *DELV* is evaluated if two satellites are trained against the same central model, and the entity alignment is performed between the satellite models. Third, a comparison to the baseline method IEAKE [Zhu et al., 2017] is drawn.

The experiments have been carried out for all combinations of central and satellite for the chosen KGs. Only the results for the combination of central FB and satellite DB are shown in detail, while the other results can be found in the Appendix A. As described in Section 3.2.4, the experiments are evaluated using the metrics *mean rank, hits@1* and *hits@10*. Moreover, the mean rank is compared to a random estimator, that is defined as $\frac{\#entities\_central}{2}$. The evluation is done for all entities that are not part of the alignment.

4.3.1 Experiment 1: Central - Satellite

In Experiment 1, one satellite model is trained in dependence of the central model (see Figure 1.1), with a given alignment. With a rising percentage of the alignment ratio, the performance improves, although it simultaneously displays a diminishing utility, as Figure 4.4 shows. Increasing the alignment ratio on low absolute levels greatly affects the performance, while changes on high absolute levels only sparsely impact the results. For some combinations of satellite and central, overly high alignment ratios even affect the results negatively. However, using 95% alignment ratio leaves only about 500-750 entitiy embeddings for the entity alignment. Therefore, we would expect higher deviations when repeating the experiments with high alignment ratios, compared to lower ratios.

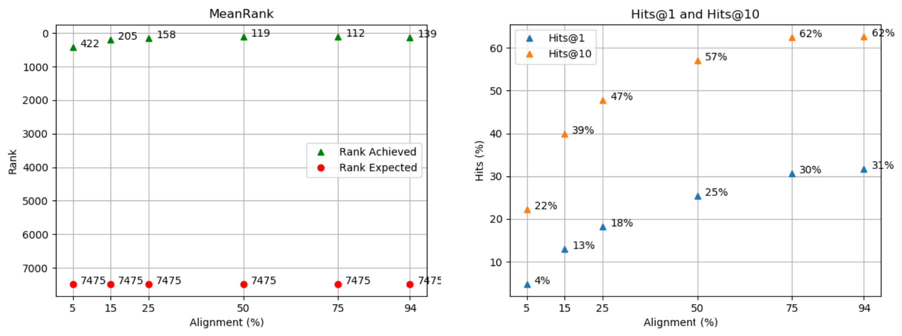
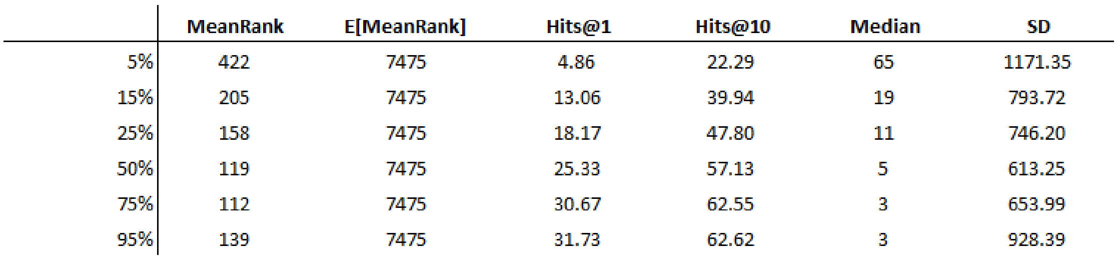


Figure 4.4: Results of Experiment 1. Central: FB, Satellite: DB

The same results, enriched with median and standard deviation of the ranks, are presented in Figure 4.5. The improved performance with a rising alignment ratio is also reflected in a diminishing standard deviation.

Furthermore, the median of the ranks is strictly lower than the mean, which indicates a



	MeanRank	E[MeanRank]	Hits@1	Hits@10	Median	SD
5%	422	7475	4.86	22.29	65	1171.35
15%	205	7475	13.06	39.94	19	793.72
25%	158	7475	18.17	47.80	11	746.20
50%	119	7475	25.33	57.13	5	613.25
75%	112	7475	30.67	62.55	3	653.99
95%	139	7475	31.73	62.62	3	928.39

Figure 4.5: Additional Statistics for Experiment 1

right-skewed distribution of the ranks. Looking at the distribution (Figure 4.6), given an alignment ratio of 25%, 75% of the ranks achieved are below 50 whereas the mean rank is 158. Therefore, the right-skewed distribution of the ranks implies that the average is impaired by outliers on the right-hand side.

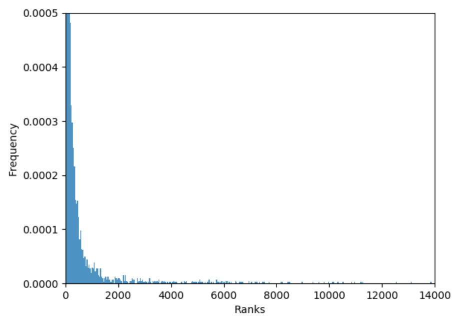
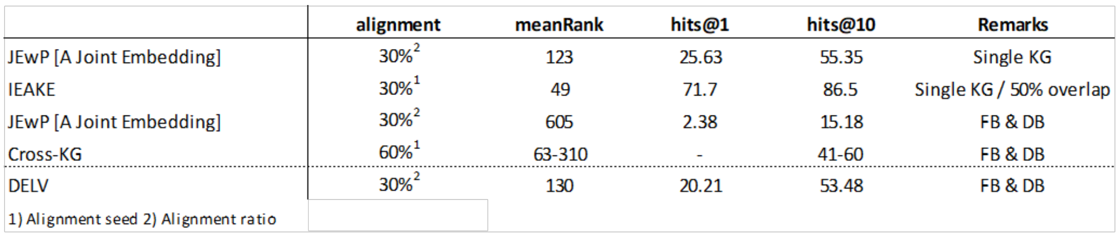


Figure 4.6: Distribution of the Ranks

To contextualize the results, Figure 4.7 summarizes the performance of the baselines most comparable to *DELV*. The first two approaches were tested on a single KG only, whereas the later methods were evaluated on the KGs FB and DB, as the last column indicates. The difference is important: Testing on datasets constructed with a single KG (see Section 2.3) simplifies the problem, thereby producing better results. Furthermore, the datasets on single KGs were often designed with an overlap of triples, e.g. [Zhu et al., 2017], which is neither possible nor desirable for different KGs.

The performance of DELV is comparable with Cross-KG [Cai et al., 2017], although Cross-KG uses an alignment ratio of 60%. DELV clearly outperforms JEwP [Hao et al., 2016] for comparable datasets. It gets outperformed by IEAKE in this comparison although the comparison is distorted as the results for IEAKE originate from a single KG dataset with a 50% overlap. As the authors of [Zhu et al., 2017] have published their dataset, a more detailed comparison between the methods can be found in Section 4.3.3.

When comparing the methods, it is important to keep the conceptual differences in mind. Those methods calculate embeddings for two KGs simultaneously and in a unified vector space. In turn, DELV presents an iterative approach where the embeddings for



	alignment	meanRank	hits@1	hits@10	Remarks
JEwP [A Joint Embedding]	30%^{2}	123	25.63	55.35	Single KG
IEAKE	30%^{1}	49	71.7	86.5	Single KG / 50% overlap
JEwP [A Joint Embedding]	30%^{2}	605	2.38	15.18	FB & DB
Cross-KG	60%^{1}	63-310	-	41-60	FB & DB
DELV	30%^{2}	130	20.21	53.48	FB & DB
1) Alignment seed 2) Alignment ratio					

Figure 4.7: Comparison of DELV to Baselines

the central KG are calculated first. Only then, the satellite model embeddings can be trained in dependence to the existing central embeddings.

Figure 4.8 compares the results over the different combinations of central and satellite models. For low alignment ratios, the performance differences between the KGs are huge, while higher alignment ratios bring the results more into line with each other. Looking at the combinations leading to poorer results, WD is the underperforming KG in this test set up. Having WD as satellite leads to the poorest results, while having it as central produces mediocre results. Likely, this is due to the reduced amounts of triples in the WD dataset (128'498), compared to FB (483'142) or DB (619'793) as explained in Section 4.1.

The next section presents the results of the second experiment.

4.3.2 Experiment 2: Central - Satellite 1 - Satellite 2

As shown in Figure 1.1, the idea of this experiment is to train two satellites against the same central model, and to subsequently evaluate, how well the entity alignment works between the two satellites (see Figure 1.1). Intuitively, worse results compared to the first experiment are to be expected as this approach involves three different KGs. Also, the KGs that are compared were never trained against each other. The experiment has been carried out for all variations of the central model. While the results for the experiment central DB, satellite 1 WD and satellite 2 FB are shown below, the other evaluations can be found in Appendix A. Figure 4.9 shows the result for this experiment. A minimum alignment ratio of 25% is needed to produce reasonable results. Increasing the ratio to 50% between each of the central/satellite pairs results in a mean rank of 189, which is in the range of the results shown in the first experiment.

Again, a right-skewed distribution pulls up the mean as a comparison between mean and median shows (Figure 4.10). The standard deviation is higher than in experiment 1, which is to be expected.

Looking at the alignments in more detail, one has to distinguish between the following three cases (compare with Figure 4.11):

1. **SameEntities:** The set of links between sat1/cent is the same as the set of links between sat2/cent. This means that if an entity e_1 in sat1 is linked to its correspondent entity e_1^* in cent, the correspondent entity e_1' in sat2 is also linked to e_1^*

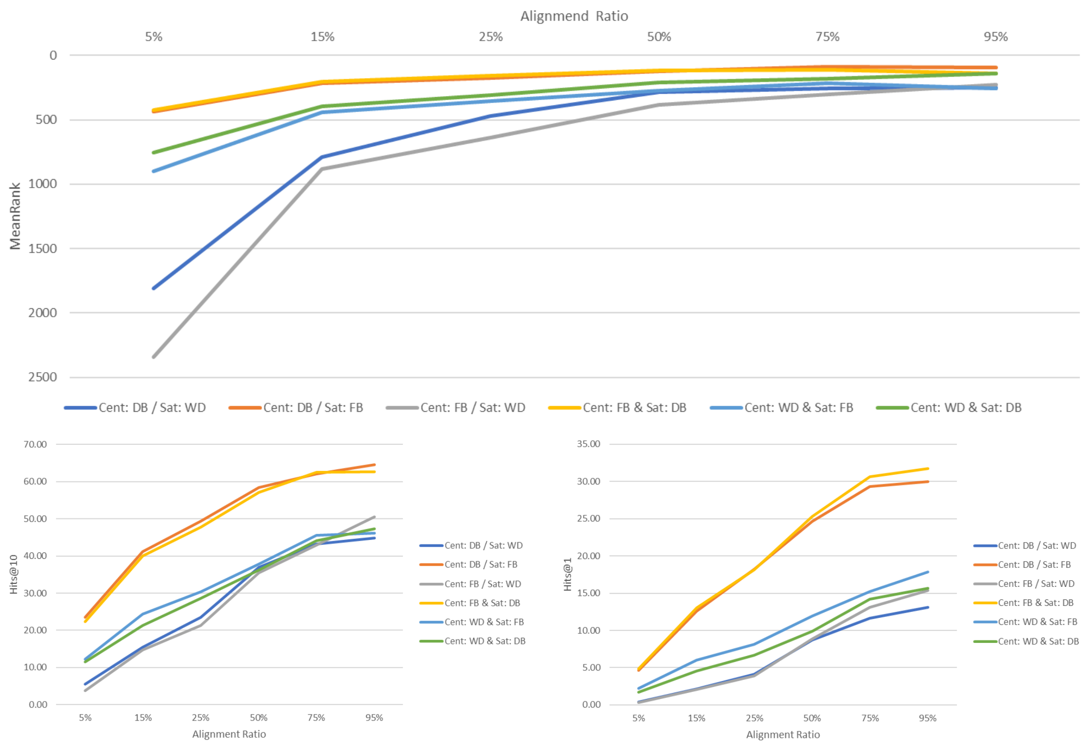


Figure 4.8: Comparison of the Results for Different Central Models

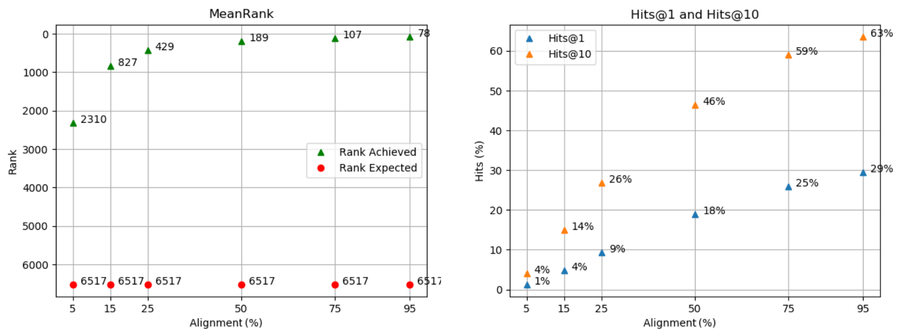


Figure 4.9: Results for the Experiment 2. Central: DB, Satellite1: WD, Satellite2: FB

in cent. In practice, this results in having the same embedding for e_1 in sat1 and e'_1 in sat2.

2. **MixedEntities:** The two sets of links between sat1/cent and sat2/cent are neither disjoint nor the same.

3. **DifferentEntities:** The two sets of links between sat1/cent and sat2/cent are disjoint.

	MeanRank	Expect. MeanRank	Hits@1	Hits@10	Median	SD
5%	2310	6517	1.32	4.09	1192.5	2760.51
15%	827	6517	4.90	14.98	205.5	1590.46
25%	429	6517	9.28	26.79	57	1111.45
50%	189	6517	18.87	46.44	12	712.66
75%	107	6517	25.83	59.14	5	528.55
95%	78	6517	29.54	63.54	3	364.63

Figure 4.10: Additional Statistics for Experiment 2

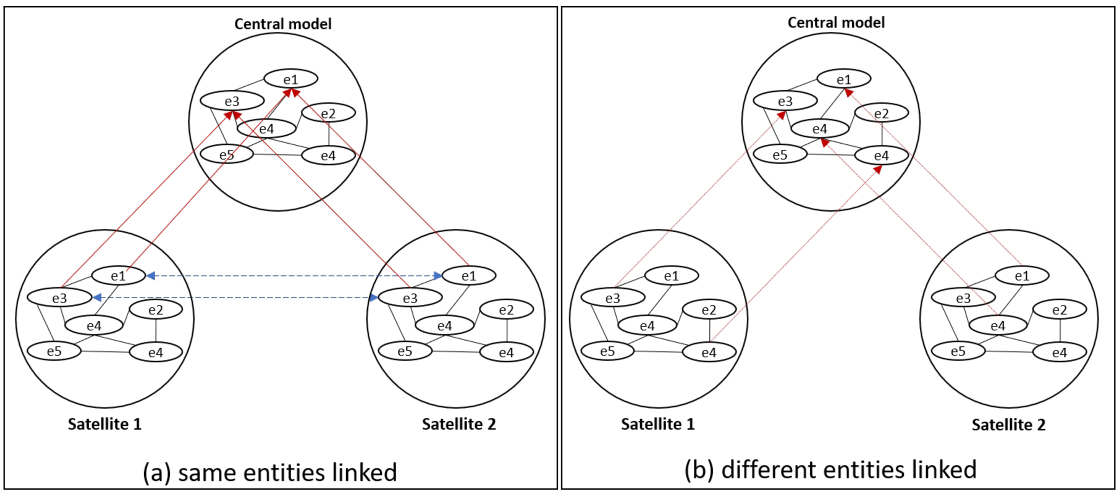


Figure 4.11: The Different Alignment Cases

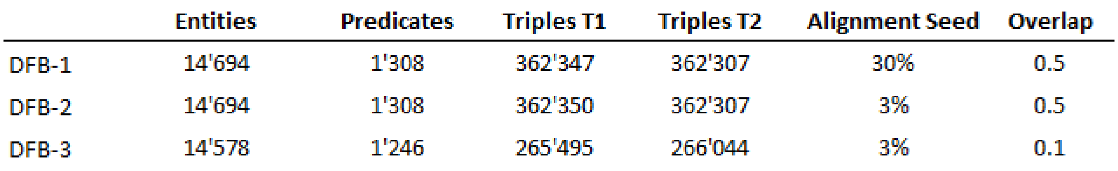
Intuitively, the first is the best case, the second is the likely case while the third is the worst case performance wise. While the results presented above correspond to the second case, Figure 4.12 shows the numbers for the other two cases with an alignment ratio of 25%. To test those cases, the alignments have been prepared accordingly. As expected, the cases 1 and 3 set the boundary for case 2. Moreover, the results of case 2 are close to the *diffEnt* case numbers. This suggests that only a few entities were linked over all three KGs in the second case.

	Alignment	meanRank	hits@1	hits@10	Median	Stdev
sameEntities	25%	314	26.10%	37.89%	31	901
mixedEntities	25%	429	9.28%	26.79%	57	1111
differentEntities	25%	486	8.48%	24.00%	73	1227

Figure 4.12: Performance for the Different Alignment Cases

4.3.3 Comparison to IEAKE

This section lays out a direct comparison between *DELV* and *IEAKE* [Zhu et al., 2017]. In the first experiment, *DELV* was evaluated on the datasets used in this thesis. The results in this chapter compare *IEAKE* and *DELV* on the basis of the datasets used in their paper. After consultation with the authors and following the detailed process described in the paper, their datasets *DFB-1, DFB-2* and *DFB-3* were reconstructed and used for testing. As described in detail in Section 2.3, entity alignment on these datasets constitute a simpler problem compared to the datasets used in experiment 1 and 2. Figure 4.13 describes the characteristics of the data sets. The overlap refers to the amount of triples the two graphs T1 and T2 share. E.g., 50% of the triples in T1 are the same as in T2 for DFB-1. Note that compared to *IEAKE*, a difference in the number of triples exists as only the training section of *FB15K* has been used, whereas in *IEAKE* the whole *FB15K* was utilized.



	Entities	Predicates	Triples T1	Triples T2	Alignment Seed	Overlap
DFB-1	14'694	1'308	362'347	362'307	30%	0.5
DFB-2	14'694	1'308	362'350	362'307	3%	0.5
DFB-3	14'578	1'246	265'495	266'044	3%	0.1

Figure 4.13: Characteristics of the Rebuilt IEAKE Datasets

Looking at the results (Figure 4.14), *DELV* clearly outperforms *IEAKE* in terms of the mean rank, achieving an outstanding rank of 10 on *DFB-1* with a low standard deviation of 168. Turning to hits@1 and hits@10, the picture is not as clear. While still outperforming on *DFB-1*, *IEAKE* reports better results for *DFB-2* and *DFB-3*. This can be attributed to the technique of soft alignment, described in Section 2.3. After a fixed amount of training iterations, for all non-aligned entities in the first KG, the most similar entity in the second KG is calculated. If the difference between the entities is smaller than a certain confidence score, the entities will be considered aligned for the rest of the training. Therefore, *IEAKE* is able to continuously update and find new entity alignments while training, whereas *DELV* does not include such a soft alignment process. While this technique pushes the hits@1 and hits@10 ratios, it could also adversely affect the mean rank if *IEAKE* finds wrong alignments during the training process. This is because the incorrect alignment gets propagated for the rest of the training which likely distorts the embeddings.

Furthermore, the author also tried to run *IEAKE* on the dataset used in this thesis. Unfortunately, no meaningful results could be produced. The results were so poor, that the author was unsure whether the performance needs to be attributed to an error when adapting the *IEAKE* code to the new datasets or whether the method really underperforms on these datasets.

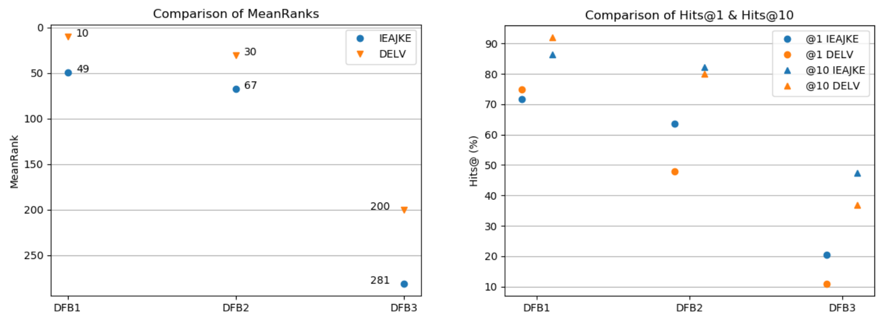


Figure 4.14: Comparison of *DELV* and *IEAKE*

4.3.4 Word Vector Evaluation

While entity alignment is the main purpose of this thesis, word2vec and rdf2vec were originally designed to produce word vectors which can be used in standard NLP tasks such as classification, speech recognition and so forth. Therefore, it is interesting to see whether a satellite model still yields meaningful word vectors. A full evaluation of the word vectors, as shown in [Ristoski et al., 2017], is out of scope for this thesis. Instead, a classification task shall provide indications.

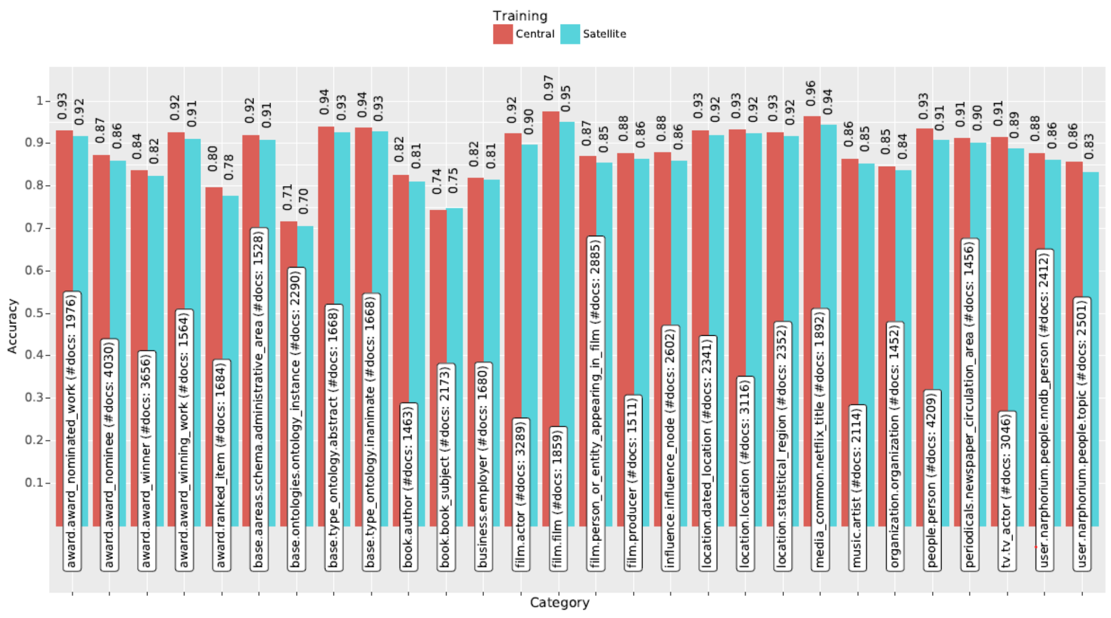


Figure 4.15: Classification with Respect to Freebase Categories for a Central and a Satellite Model

Figure 4.15 presents the results of the classification task, in which the word vectors of the central and satellite model were used to predict the correspondent freebase categories for an entity. In 50% of the categories, an accuracy above 90% can be reported while, except for two cases, an accuracy above 80% resulted for the rest. While the satellite embeddings underperform consistently, the difference is minimal. This suggest that the embeddings produced by DELV can still be used as high-quality word vectors in standard NLP tasks.

Furthermore, the first seven country and capital pairs in a FB satellite model, trained against a DB central model, have been plotted in Figure 4.16, using the dimensionality reduction algorithm t-SNE [Maaten and Hinton, 2008]. As put in [Ristoski et al., 2017]: “The figure illustrates the ability of the model to automatically organize entities of different types, and preserve the relationship between different entities”. Figure 4.16 shows that the model maps similar entities, such as capital and country, close to each other. Even more importantly, the model preserves the relationship between the entities. Visually spoken, the distances in the plot between different country-capital pairs are similar, which is also an indication for high quality word vectors.

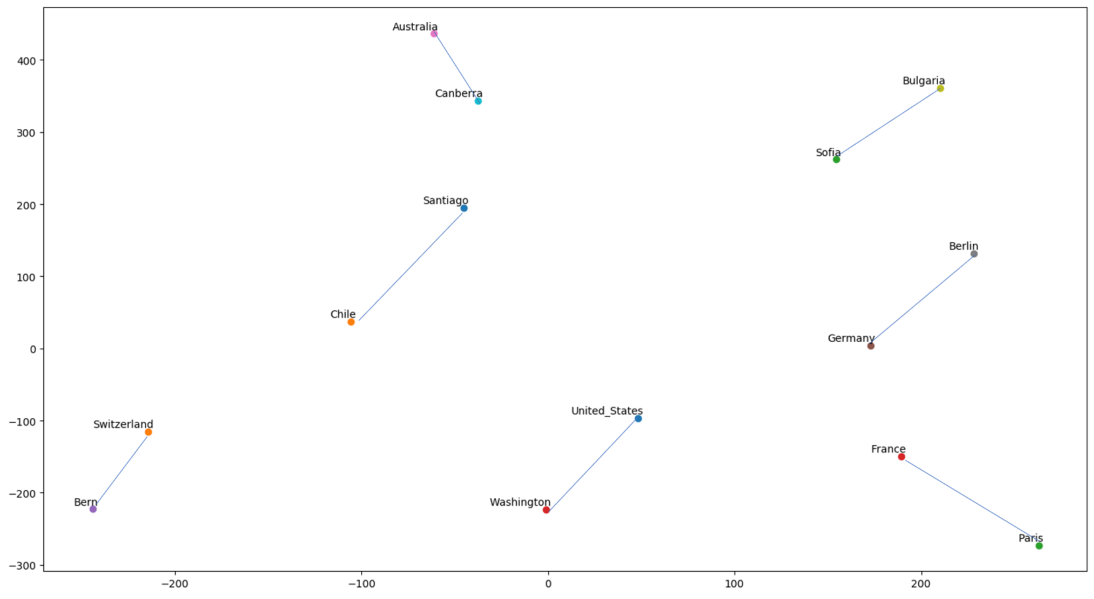


Figure 4.16: Projection of Capital-Country Pairs using t-SNE

5 Conclusion

In this thesis, an iterative, embedding-based, machine learning approach for entity alignment has been presented. The method Dependent Learning of Entity Vectors (DELV) is based on word2vec and rdf2vec and manipulates the word2vec training process in such a way that the similar structures of KGs are reflected in similar entity embeddings. After summarizing relevant literature, this thesis further explored the idea and intuition behind *DELV*. Presented as a machine learning pipeline, the implementation was explained and illustrated using code snippet where applicable. Then, the datasets used, and their construction, were discussed. *DELV* was tested on real world datasets, including up to three different KGs. The performance of *DELV* is compared to similar, embedding-based baselines. Given a comparable setup and dataset, *DELV* outperforms most of the present existing, embedding-based alternatives. Also, the experiments conducted demonstrate the feasibility of the method in the context of multiple KGs. While the embeddings are optimized for entity alignment, they can still be used as word vectors, as a classification task indicates.

DELV differs from previous methods due to its iterative nature, yielding multiple advantages. Embeddings for unlimited satellite KGs can be learned against a central KG without the need to retrain the latter. As the satellites get trained against the same central, it is also possible to compare satellites among themselves. Moreover, *DELV* allows for computation in a distributed and local fashion. This allows *DELV* to be used for datasets which cannot be published (e.g. due to licensing reasons), and avoids the need of having all datasets available in memory and at the same time.

A detailed comparison of *DELV* to the baseline method *IEAKE* showed, that while outperforming in terms of mean rank, *DELV* is inferior when looking at hits@1 and hits@10. To further improve the method, a soft alignment approach, as presented in [Zhu et al., 2017, Cai et al., 2017], could be considered in the future. Moreover, the way in which the graph walks are generated, could be improved as well. Currently, the paths are produced randomly. An alternative could be that the edge selection is based on a frequency metric. E.g., DBpedia offers a dataset called “out-degree” that includes the number of links emerging from a Wikipedia article and pointing to another article. A higher out-degree implies a closer relationship between the respective concepts. If the edge selection were based on such a metric, more meaningful sequences could be produced, and the structure of a KG could be mapped better.

References

[Achichi et al., 2016] Achichi, M., Cheatham, M., Dragisic, Z., Euzenat, J., Faria, D., Ferrara, A., Flouris, G., Fundulaki, I., Harrow, I., Ivanova, V., et al. (2016). Results of the ontology alignment evaluation initiative 2016. In *OM: Ontology Matching*, pages 73–129. No commercial editor.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

[Bollacker et al., 2008] 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. AcM.

[Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

[Cai et al., 2017] Cai, P., Li, W., Feng, Y., Wang, Y., and Jia, Y. (2017). Learning knowledge representation across knowledge graphs.

[Cheatham et al., 2015] Cheatham, M., Dragisic, Z., Euzenat, J., Faria, D., Ferrara, A., Flouris, G., Fundulaki, I., Granada, R., Ivanova, V., Jiménez-Ruiz, E., et al. (2015). Results of the ontology alignment evaluation initiative 2015. In *10th ISWC workshop on ontology matching (OM)*, pages 60–115. No commercial editor.

[Chen et al., 2017] Chen, M., Zhou, T., Zhou, P., and Zaniolo, C. (2017). Multi-graph affinity embeddings for multilingual knowledge graphs.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

[de Vries, 2013] de Vries, G. K. (2013). A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 606–621. Springer.

[Goldberg and Levy, 2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

[Guan et al., 2017] Guan, S., Jin, X., Jia, Y., Wang, Y., Shen, H., and Cheng, X. (2017). Self-learning and embedding based entity alignment. In *Big Knowledge (ICBK), 2017 IEEE International Conference on*, pages 33–40. IEEE.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

[Hao et al., 2016] Hao, Y., Zhang, Y., He, S., Liu, K., and Zhao, J. (2016). A joint embedding method for entity alignment of knowledge bases. In *China Conference on Knowledge Graph and Semantic Computing*, pages 3–14. Springer.

[Hassanzadeh and Consens, 2009] Hassanzadeh, O. and Consens, M. (2009). Linked movie data base. *Linked Data on the Web (LDOW2009)*, 16.

[Hayes and Patel-Schneider, 2014] Hayes, P. J. and Patel-Schneider, P. F. (2014). Rdf 1.1 semantics. w3c recommendation, february 2014. *World Wide Web Consortium. Retrieved from https://www. w3. org/TR/2014/REC-rdf11-mt-20140225*.

[Heuer, 2016] Heuer, H. (2016). Text comparison using word vector representations and dimensionality reduction. *arXiv preprint arXiv:1607.00534*.

[Hinton et al., 1986] Hinton, G. E., McClelland, J. L., Rumelhart, D. E., et al. (1986). Distributed representations. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(3):77–109.

[Jansen, 2017] Jansen, S. (2017). Word and phrase translation with word2vec. *arXiv preprint arXiv:1705.03127*.

[Lacoste-Julien et al., 2013] Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., and Ghahramani, Z. (2013). Sigma: Simple greedy matching for aligning large knowledge bases. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 572–580. ACM.

[Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2009] Mikolov, T., Kopecky, J., Burget, L., Glembek, O., et al. (2009). Neural network based language models for highly inflective languages. In *Acoustics,* *Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4725–4728. IEEE.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

[Rehurek and Sojka, 2010] Rehurek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

[Ristoski and Paulheim, 2016] Ristoski, P. and Paulheim, H. (2016). Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer.

[Ristoski et al., 2017] Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., and Paulheim, H. (2017). Rdf2vec: Rdf graph embeddings and their applications. In *Semantic Web Journal*. SWJ.

[Rong, 2014] Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

[Ruder, 2016] Ruder, S. (2016). On word embeddings-part 2: Approximating the softmax.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

[Schmachtenberg et al., 2014] Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer.

[Suchanek et al., 2011] Suchanek, F. M., Abiteboul, S., and Senellart, P. (2011). Paris: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, 5(3):157–168.

[Sun et al., 2017] Sun, Z., Hu, W., and Li, C. (2017). Cross-lingual entity alignment via joint attribute-preserving embedding. In *International Semantic Web Conference*, pages 628–644. Springer.

[TensorFlow, 2017] TensorFlow (2017). Vector representations of words.

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

[Yan et al., 2017] Yan, J., Xu, C., Li, N., Gao, M., and Zhou, A. (2017). Optimizing model parameter for entity summarization across knowledge graphs. *Journal of Combinatorial Optimization*, pages 1–26.

[Yanardag and Vishwanathan, 2015] Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM.

[Zhu et al., 2017] Zhu, H., Xie, R., Liu, Z., and Sun, M. (2017). Iterative entity alignment via joint knowledge embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4258–4264. AAAI Press.

A

Appendix

A.1 Results of Experiment 1

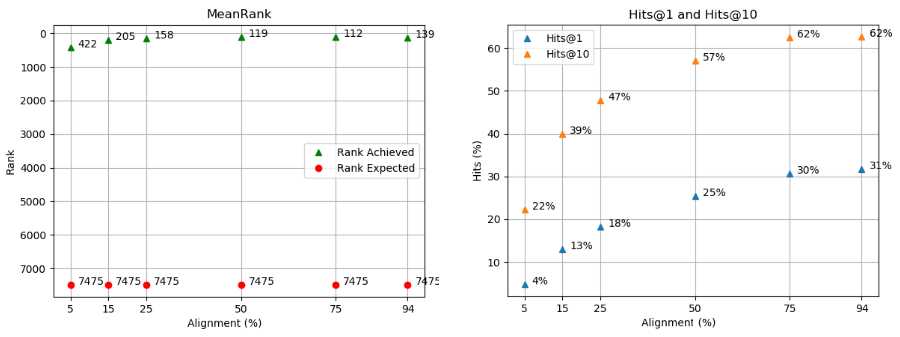


Figure A.1: Central: FB, Satellite: DB

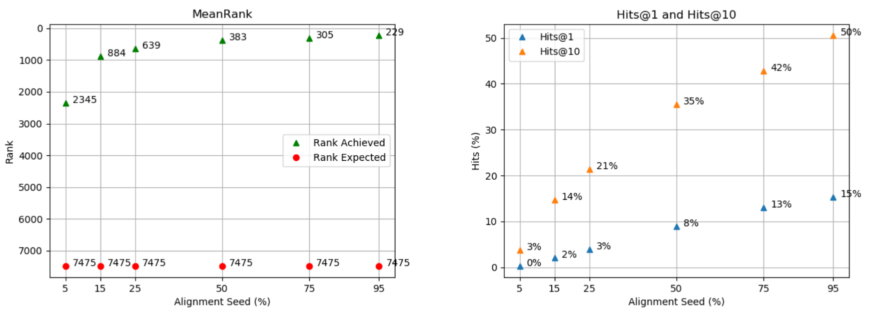


Figure A.2: Central: FB, Satellite: WD

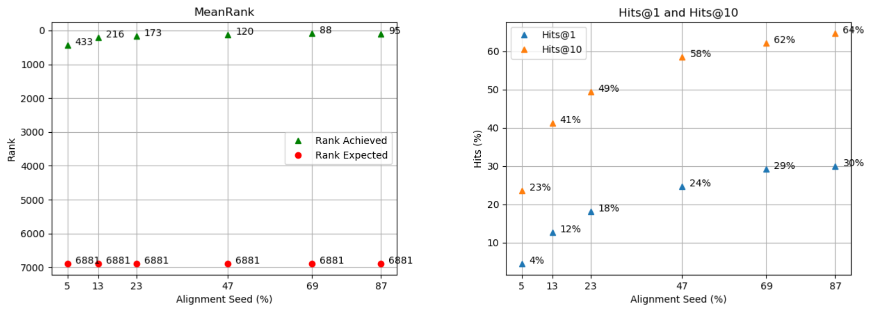


Figure A.3: Central: DB, Satellite: WD

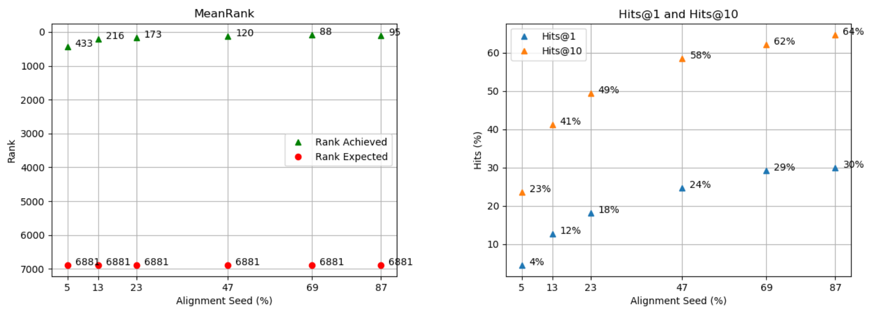


Figure A.4: Central: DB, Satellite: FB

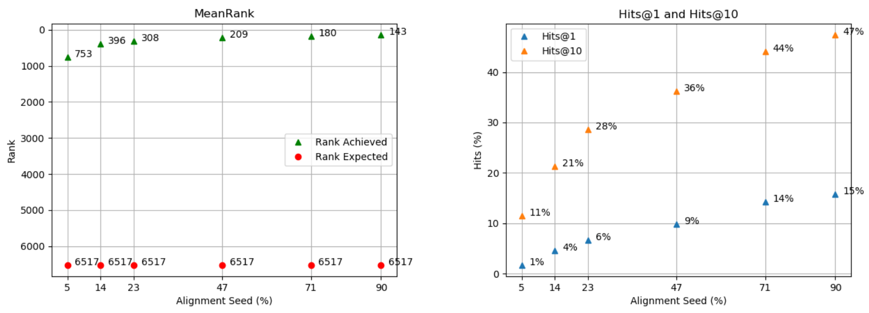


Figure A.5: Central: WD, Satellite: DB

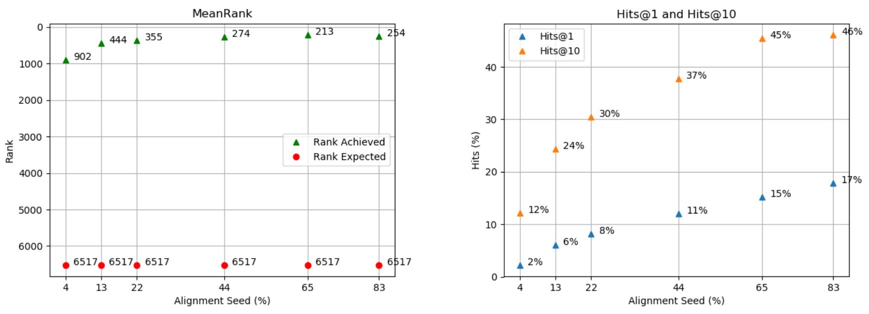


Figure A.6: Central: WD, Satellite: FB

A.2 Results of Experiment 2

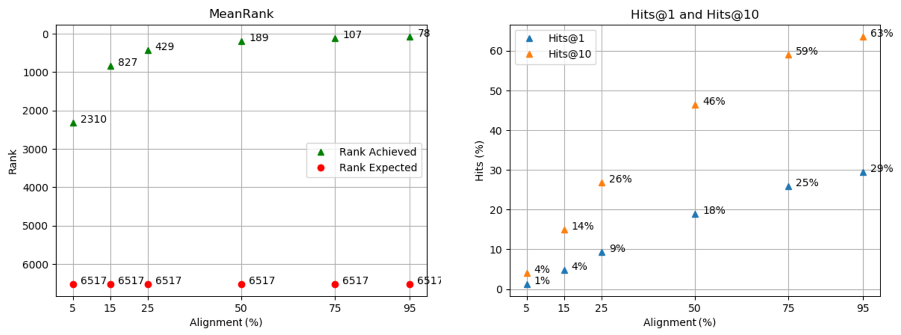


Figure A.7: Central: DB, Satellite1: WD, Satellite2: FB

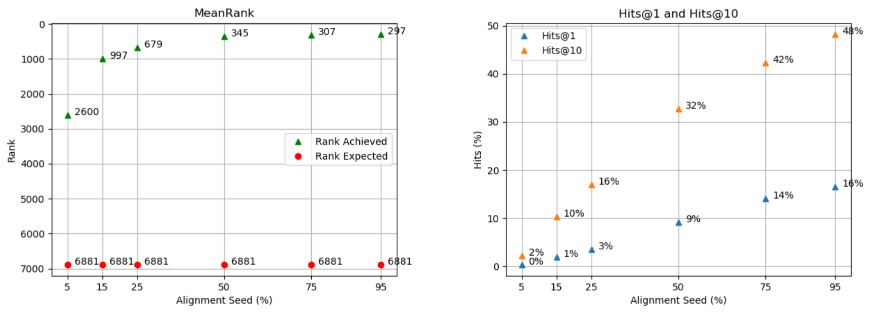


Figure A.8: Central: FB, Satellite1: DB, Satellite2: WD

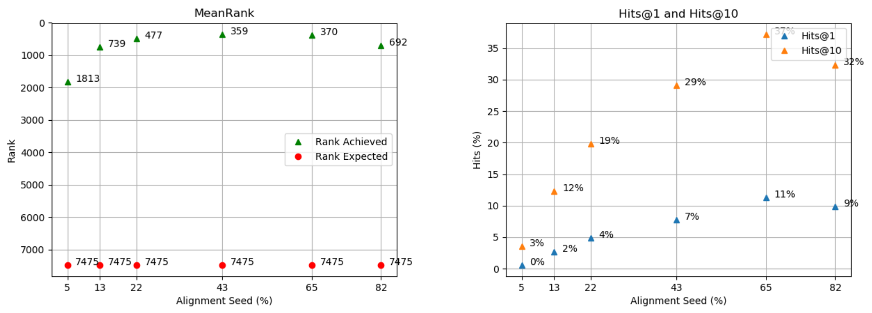


Figure A.9: Central: WD, Satellite1: FB, Satellite2: DB

A.3 Results of the Parameter Study

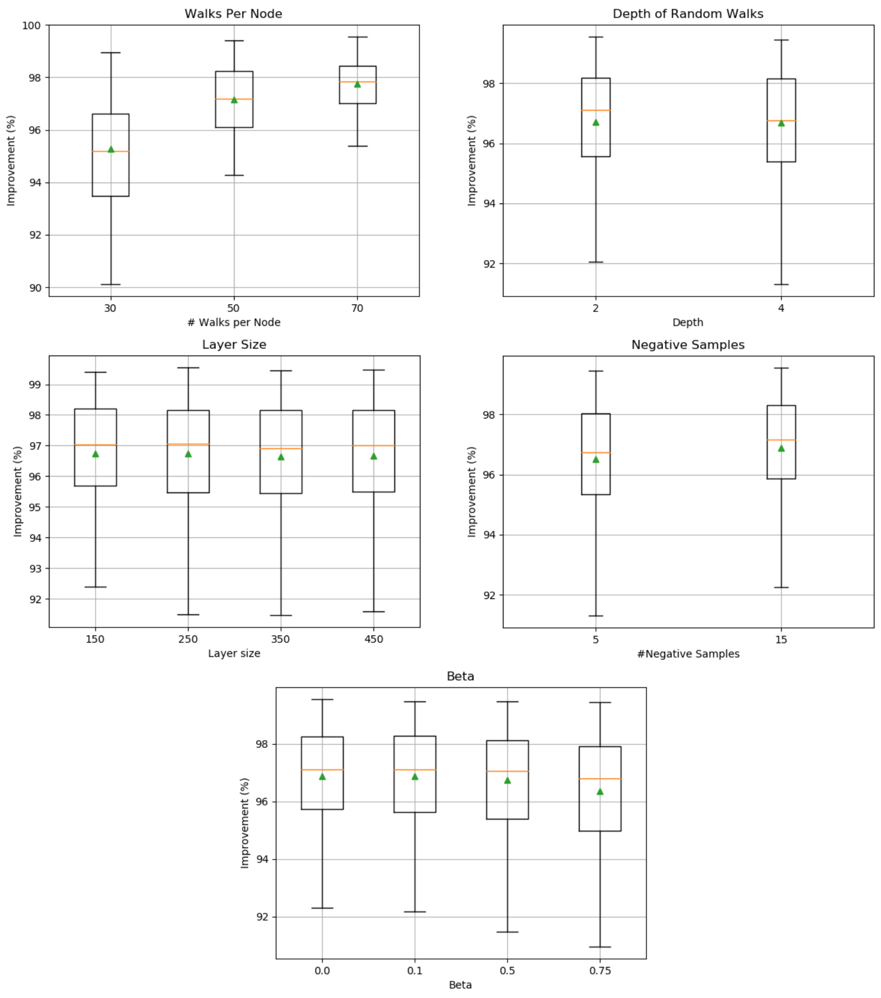


Figure A.10: Parameter study: Central: WD, Satellite: FB

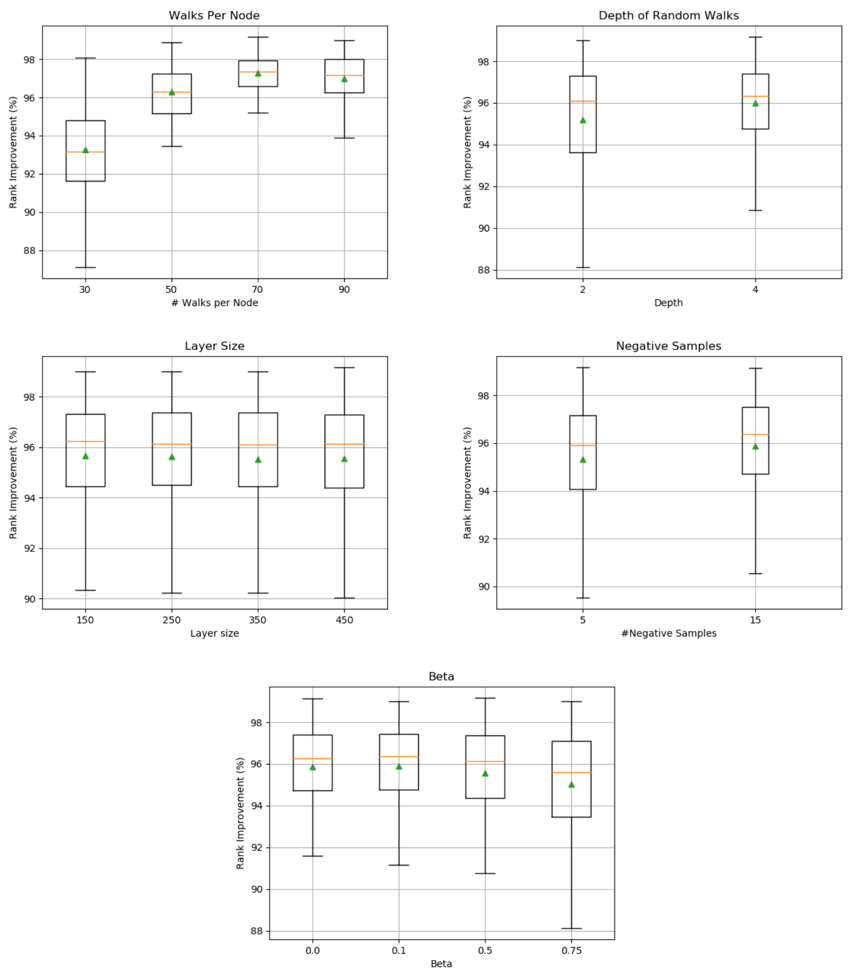


Figure A.11: Parameter study: Central: WD, Satellite: DB

List of Figures