University of Zurich[UZH]

Supervised and Unsupervised Alignment of Knowledge Graphs with pre-trained embeddings

Master's thesis August 14, 2021

Terézia Bucková
of Piešťany, Slovakia

Student-ID: 18-744-888
terezia.buckova@uzh.ch

Advisors:
Matthias Baumgartner,
Daniele Dell'Aglio

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

Acknowledgements

I would like to thank Prof. Abraham Bernstein for the opportunity to work on this master thesis.

I am very grateful to both of my advisors Matthias Baumgartner and Daniele Dell'Aglio. Our weekly meetings were proper scientific "fun" and always helped me to make wanted progress. Thank you for all the interesting questions and insights you proposed and for all of the encouraging words.

Thank you, Matthias, for your technical insight and enthusiasm about this work. Thank you, Daniele, for your advice about writing techniques and challenging point of view on this thesis.

One big thank you belongs to my husband Jozef. Thank you for taking care of little Zuzi during my working hours, and for midnight meals to keep me working. I would like to thank also my parents and parents-in-law for helping. I wouldn't make it without you all.

Zusammenfassung

Knowledge Graphs (Knowledge Graphs, KGs) sind gerichtete Graphen welche real existierende Konzepte und Beziehungen dazwischen modellieren. KGs wurde in den letzten Jahren viel Aufmerksamkeit geschenkt, und es wurden bedeutende Fortschritte bei deren Konstruktion unter verschiedenen Bedingungen erzielt. Jedoch existiert aktuell kein KG welcher jegliches Wissen enthält — um eine ganzheitliche Sicht auf ein Konzept von Interesse zu erhalten muss man daher Informationen aus mehreren KGs zusammenführen.

In der Regel bedeutet dies, dass man verschiedene KGs abgleichen muss um herauszufinden welche ihrer Entitäten sich auf dasselbe Konzepte beziehen. Algorithmen die eine solche Alignierung (alignment) bestimmen können oft verbessert werden in dem sie dafür eine Einbettung (embedding) nutzen welches jede Entität und jede Beziehung durch einen Vektor repräsentiert.

In dieser Arbeit testen wir typische embedding-basierte Alignierungsmethoden für Wörter auf ihre Anwendbarkeit auf KGs, in dem wir die KG embeddings als Eingabe annehmen. Wir zeigen, dass diese Methoden mit typischen KG Alignierungsmethoden in Bezug auf die Hits@k Metrik gleichauf sind, wie auch dass sie die Ergebnisse so ausbalancieren, dass die Entitäten der KGs im embedding als nächste Nachbaren auftreten.

Des weiteren untersuchen wir die Auswirkungen verschiedener embedding-Modelle auf die Alignierung von KGs und kommen zum Schluss, dass die Wahl des Modells das Ergebnis stark beeinflusst. Gleichzeitig zeigen wir, dass die Alignierung um etwa 20 Prozentpunkte schlechtere Ergebnisse in Bezug auf Hits@k liefert wenn die KGs mit unterschiedlichen Modellen eingebettet werden.

Abstract

Knowledge Graphs (KGs), directed graphs representing real-world objects and relations between them, have gained significant attention in the past few years, and progress has been made to construct such KGs in various contexts. However, no current KG holds the complete knowledge and in order to obtain a holistic view about an entity of interest, one must therefore gather data from multiple KGs.

This usually means to align different KGs and to figure out which entities refer to the same real-world objects. The alignment algorithms often benefit from aligning a KG embeddings, in which case every entity, and possibly relation, is represented by an embedding vector.

The embedding methods and embedding-based word alignment techniques in language processing have been researched for a longer period of time. This effort has led to more accurate assumptions about embedding spaces and high performance in alignment tasks in both supervised and unsupervised scenarios.

In our work, we test state-of-the-art word embedding alignment methods using KG embedding spaces as input data. We show that typical word alignment methods are on par with typical KG alignment methods in terms of their hits@k score. Moreover, word alignment methods balance the results so that correctly aligned entities are mutual nearest neighbours in the aligned embedding spaces.

In addition, we investigate the effect of various embedding models on KG alignment and conclude that the choice of the embedding model has a large impact on the final alignment results. At the same time, we challenge the assumption that both KGs have to be embedded by two instances of the same embedding model and show that embedding them with different models yields results up to 20 percentage points worse at hits@k.

Contents

1

Introduction

A Knowledge Graph (KG) is a data structure that stores facts as triples of the form (head entity, relation, tail entity). A multitude of facts about the same entity leads to a directed graph whereas vertices represent real-world entities, and links between vertices indicate the relationship between the entities. A number of such KGs has emerged in the previous years, such as Wikidata[1], DBpedia[2], or YAGO[3] [Heist et al., 2020]. These KGs stand out in their versatility, size, and transgression of domains.

To date, no current KG holds the complete knowledge, and many KGs include facts about the same entity. To get an holistic view about an entity, one must therefore gather data from multiple KGs. Having two KGs, one must know which two vertices represent the same entity. Finding such correspondences between two KGs is the goal of KG alignment. This is a non-trivial task, since entities are often locally identified, and entity attributes like human-readable labels might be missing or difficult to compare, e.g. if they are in different languages.

Recent approaches to KG alignment employed KG embedding models. An embedding model computes a numerical vector (embedding) for each entity (and possibly relation) of a KG, such that related entities have similar embeddings. Such models have been researched thoroughly in the previous years, and several distinctive embedding models have emerged [Wang et al., 2017, Bordes et al., 2013, Sun et al., 2019, Yang et al., 2014]. While embedding spaces from two different model instances may not be readily comparable (e.g. they could be rotated), the core idea to use them for KG alignment is to learn a mapping between them such that similarity between embeddings implies a correspondence between the respective entities.

The main problem this thesis will investigate is: *given two KGs, two embedding spaces learned from them, and optionally a partial mapping function between entities and relation, is it possible to infer the complete mapping function between the two knowledge graphs?*

One of the difficulties of this approach is that we align two previously trained, static KG embedding spaces without any other information. Consider the following example

[1] *https://www.wikidata.org*
[2] *https://www.dbpedia.org*
[3] *https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago*

from the medical domain: You want to align a KG about patients with a KG about diseases. State-of-the-art KG alignment methods would use attributes like entity labels, because they provide a simple means to find an initial mapping between the two KGs. In the medical scenario, patients data are private and hence the entity labels will not be available, rendering state-of-the-art approaches inapplicable. Another typical approach in the KG alignment is joint learning. In our situation, this approach is also not applicable because the training again violates privacy concerns.

In the area of word embeddings, several works explored how to find such a mapping between two previously computed embedding spaces, either in a supervised setting by using a number of known alignments (anchors), or in a completely unsupervised fashion [Alvarez-Melis and Jaakkola, 2018, Conneau et al., 2017]. A natural approach, therefore, is to evaluate these previously published embedding space alignment models in the KG alignment setting.

In a realistic situation, our previous medical example has another caveat: The pretrained embedding spaces of different KGs could be trained by different embedding models. The KGs usually contain thousands of entities and hundreds of relations. Therefore, the embedding training requires a lot of resources and time, which makes retraining for the purpose of alignment costly.

In this thesis we identify two research questions:

1. *Do the word alignment methods used on KGs have results in the same order of magnitude as typical KG alignment methods?*

2. *Do the alignment between different KG embedding models yield results in the same order of magnitude with alignment between the same embedding models?*

We conduct number of experiments with various alignment methods and KG embedding models to find answers to these questions. We try to keep our scenario realistic by aligning two different KGs: fb15k-237 and wd15k-237.

The rest of this document is structured as follows: In Chapter 2 we will describe state-of-the-art word and KG alignment methods. We also list surveys which summarize KGs available on the web and typical KG embedding methods. In the Chapter 3 we introduce preliminary knowledge about embeddings and alignment models used later in the experiments. The Chapter 4 describes KGs used, technical details about KG embedding training, and hyperparameter setting of alignment methods. At the end of the chapter we introduce the evaluation metrics used to evaluate the alignment. In Chapter 5 we describe the motivation of our research questions, the underlying hypotheses, and the experiments we conduct to verify them. The Chapter 6 describes the possible directions of future research. In Chapter 7 we summarize the answers to our research questions and the knowledge we gained during the process of this thesis.

2

Related Work

In this chapter we discuss work that inspired this thesis and similar work in the field of embedding space alignment.

2.1 Word embedding space alignment

2.1.1 Orthogonal Procrustes

Cosine similarity distance metric

[Chen et al., 2018] propose a supervised system for the visualization of embedding spaces. This interactive analytical system is able to compare two spaces via different tasks: display of a word embedding space, display of clusters, display of the nearest semantic neighbours of a word, and display of analogies. Therefore, it gives us an intuitive understanding of the differences and similarities between two vector spaces. To align these two spaces the authors solved the orthogonal Procrustes problem on the set of common prior aligned words. To measure the distance between the embeddings the authors used cosine similarity.

Cross-domain similarity local scaling (CSLS) as distance metric

[Conneau et al., 2017] present a new way of aligning two word embedding spaces without supervision. The authors introduce a new criterion to evaluate the alignment of words between two domains, called Cross-domain similarity local scaling (CSLS). The CSLS uses kNN in a bipartite graph scenario where all words from the source language are connected with the k-nearest-neighbors from the target language and vice versa. The decision if a source word is a translation of a target word is based on the computed similarity of these neighbourhoods. This way the algorithm increases the similarity between correct pairs of marginal vectors and decreases the similarity between incorrect pairs with word hubs.

[Artetxe et al., 2018] present a new approach to align two monolingual embedding spaces without supervision. The authors build on a hypothesis that given the similarity matrices within the vocabularies of two languages the same word in both languages has

very similar distributions of similarity values. The proposed method starts by normalizing the embeddings, so that the dot product of any two embeddings is the same as their cosine distance. The next step is building an unsupervised initial alignment between two spaces. The authors hypothesise that any two embedding spaces are perfectly isometric. Thus the similarity matrices representing word embeddings are just permutations of one another. The rows of matrices are word embeddings, the columns are embedding dimensions. The algorithm finds the permutation of the j-th dimension by sorting every row of the similarity matrices of the two languages. According to the isometry assumption, the same word should have the same sorted vectors. After obtaining this initial alignment, the authors continue with robust self-learning of embedding mappings. The first self-learning step is to compute the optimal orthogonal mapping for the current dictionary (containing entities which are used as anchors). The second self-learning step computes the optimal dictionary given the similarity matrix of the languages. The authors propose four improvements to help the algorithm converge: stochastic dictionary induction, frequency-based vocabulary cutoff, CSLS metric instead of kNN, and a bidirectional dictionary induction. The last step is a symmetric re-weighting of the embedding matrices based on cross-correlation in each component.

Wasserstein distance metric

[Alvarez-Melis and Jaakkola, 2018] suggest a new approach for aligning word embedding spaces based on optimizing the Optimal Transport (OT) problem. The OT problem finds a minimal cost mapping between two sets of discrete points. Further, the authors applied the Gromov-Wasserstein distance which generalizes OT. The Gromov-Wasserstein distance couples embeddings across two embedding spaces without the linear transformation. Across different languages, the Gromov-Wasserstein distance measures the similarity of given pairs of languages.

[Grave et al., 2019] align two embedding spaces between two different models/languages using Procrustes with the Wasserstein distance. The Procrustes algorithm finds a linear transformation between two spaces if some alignments are known already. The Wasserstein distance finds an alignment between two spaces, given a linear transformation between them. The authors combine these two approaches into one single optimization for unsupervised alignment learning. The authors use stochastic optimization to solve the Procrustes problem with Wasserstein distance. During the training, the model uses batches of data, computes the optimal permutation on them, and computes the gradient of the orthogonal matrix. The orthogonal matrix is a solution to the given optimization problem. This approach is not convex and very dependent on the initialization. Thus, the authors also propose the convex relaxation of the problem. The results of the relaxation can be used as a good initialization setting for solving the stochastic optimization.

[Zhang et al., 2017] propose an unsupervised approach to aligning two monolingual embedding spaces based on the Earth mover's distance (EMD). The optimization task can be formulated as minimization of the EMD between the source and target language embedding distributions. The authors undertake two approaches. The first one is called

Wasserstein Generative adversarial net (WGAN). The Wasserstein distance is the generalization of the EMD to continuous distributions. This model generates transformed embeddings of a source language. Together with the original target embeddings, they are passed to the discriminator which estimates the Wasserstein distance. The distance is passed back to the generator which tries to minimize it. The second approach is called EMDOT (OT stands for orthogonal transformation). This model repeatedly switches between two minimization problems. One subproblem is the EMD minimization. The second subproblem aims to find the transformation with the cross-lingual connection from the first subproblem. This problem can be transformed into an extended orthogonal Procrustes problem with a closed-form solution. Moreover, the authors show that the EMD distance is a good approximation for language dissimilarity. It is clearly correlated with the geographical distance between countries using different languages.

2.1.2 Canonical Correlation Analysis (CCA) as distance metric

[Beyer et al., 2020] propose to use the CCA algorithm from [Rastogi et al., 2015] to align different embedding spaces in a supervised way and compare the correlation between them. The authors use it to compare several corpora, always comparing two of them at a time. The CCA algorithm aligns two embedding spaces by maximizing the dimension-wise correlations. These correlations are further used to predict the overall similarity of two embedding spaces, thus the overall similarity of two corpora. The algorithm is also used in a cross-language setting.

[Ammar et al., 2016] propose two approaches to capture multilingual embeddings into one space: MultiCluster and MultiCCA. The MultiCluster algorithm clusters similar words and then trains one embedding per cluster. MultiCCA is based on the CCA algorithm. Authors train embeddings for each language separately. Then, the English embedding space is used as a common space and other embeddings are mapped to English embeddings. English was chosen because it is a high resource language. The authors also propose a new intrinsic evaluation method for multilingual embeddings named multiQVEC-CCA, which better correlates with the evaluation of extrinsic tasks. However, this evaluation method relies on a manually constructed linguistic matrix, which is not appropriate for large KGs.

2.2 KG embedding space alignment

2.2.1 Linear Transformation

[Sun et al., 2018] train alignment-oriented KG embeddings. The proposed method requires a prior alignment **A'** for the set of entities. To calibrate alignment learning, entities from **A'** are used to generate new triples with entities and relations mixed from both KG1 and KG2 respectively. For example, if an entity x is originally from KG1, then one of the new triples is $(x, r2, t2)$ with $r2$, $t2$ from KG2. Furthermore, the authors propose ϵ-Truncated Uniform Negative Sampling where they sample negative triples from

the nearest neighbours of an entity instead of the whole entity set. To leverage the prior alignment, authors use newly aligned entities in a subsequent training. To label new entities, a bootstrapping method is used. Entities can be further relabeled or unlabeled to solve labeling conflicts.

[Liu et al., 2018] propose the MGTransE model, which aligns embeddings of multiple KGs by adopting the bootstrapping method. The proposed model has three parts: a structure embedding model, a semantically smooth embedding model, and an iterative smoothness model. The structure embedding model encodes entities and relations of separate KGs into separate embedding spaces. TransE was used for this purpose. The semantically smooth embedding model aligns these different embedding spaces into one common space. Authors look at three different methods: distance-based (similar entities from two KGs are close), translation-based (special relation between same entities from two KGs), and linear projection (linear rotation and scaling). All the above-mentioned methods use a prior alignment. The iterative smoothness model updates the seed set of prior aligned entities and relations. The authors use a pipeline consisting of two components: the first one searches for the closest neighbour of an unaligned entity, the second one makes sure that entities denoting the same real-world object have common relations.

[Trisedya et al., 2019] use attribute embeddings to enhance the performance of the entity alignment task. The proposed model consists of two parts: Predicate alignment and embedding learning. The predicate alignment part aligns relations from two different KGs into a unified namespace. All the triples from the two KGs are transformed into a unified relational space. The next step is to learn the structure and attribute embeddings. The structure embeddings of entities from different KGs are in different embedding spaces. They are learned using TransE, focused on already aligned entities. The attribute embeddings are in the same space even though they are from two KGs because they are trained by a character embedding model. So, every attribute is represented by its character embedding. Similar words/phrases are close in the embedding space. The attribute embeddings are later used to shift the structure embedding spaces to have entity embeddings in the same space as attribute embeddings. All the closest pairs of embeddings are aligned. The authors define a threshold on the cosine similarity of learned embeddings to align only the probable pairs. Moreover, the authors propose to enrich attribute triples with the usage of the transitivity rule.

[Zhu et al., 2017] solve the KG entity alignment problem via joint embedding learning. The authors propose a model which consists of three parts: Knowledge embedding, joint embedding, and iterative alignment. The knowledge embedding part encodes entities using TransE or PTransE. The authors proposed three methods for joint embedding learning: A translation-based Model, a linear transformation model, and a parameter sharing model. The translation-based model assumes that a special relation exists between two aligned entities, so, it performs an alignment-specific translation to obtain joint embeddings. The linear transformation model learns a transformation matrix that translates one embedding space into another. The parameter sharing model forces the aligned entities to have the same embeddings. Thus, the final embeddings are in the

same embedding space. The iterative alignment part calculates new entity pairs which should be aligned using a seed alignment and one of the models described above. The new pairs of entities could be aligned by hard or soft alignments. The hard alignment method forces the embeddings of newly aligned pairs to be the same by averaging them (used only with the parameter sharing model). The soft alignment assigns reliability scores to pairs of newly aligned entities.

[Hao et al., 2016] propose an approach to learn joint embeddings of different KGs. The authors took TransE as inspiration. The approach is supervised using a set of seed entities. The seed entities have embeddings in the embedding space as similar as possible. This way the embeddings are in a common vector space. Further, the other entities to be aligned should be very close. The authors tried two objective functions. In the first case, the transformation matrix is not used, in the second case is.

2.2.2 Graph Convolutional Networks

[Wang et al., 2018b] align multilingual entities from two KGs using Graph Convolutional Networks (GCNs). The GCNs are trained on pre-aligned entities. The approach uses relations as structural information and attributes as another source of information about entities. Every KG is modeled by a single GCN. Every GCN produces two sets of embeddings: structure and attribute.

[Xu et al., 2019] propose another supervised approach to cross-lingual KG alignment using GCN and a topic entity graph. The topic entity graph is a graph structure that consists of a chosen entity and its one-hop neighbours. Relations are preserved only as directed edges without labels. The proposed model consists of four layers: An input representation layer, a local matching layer, a global matching layer, and a prediction layer. The input representation layer uses the GCN to embed the topic entity graphs. The local (node-level) matching layer first calculates the attentive vector of all combinations of entities based on the cosine similarity. Second, the attentive vector is used to calculate matching vectors for all entities in both KGs. The global (graph-matching) layer uses a second GCN to propagate local similarities into a global understanding of the graph structure. This layer overcomes the problem of entities referring to the same real-world object but having a small number of co-occurring neighbours. The prediction layer is a feed-forward neural network that applies the softmax function in the output layer.

[Zeng et al., 2021] propose an unsupervised approach to the entity alignment problem. The model consists of three modules: side information leveraging, unmatchable entity prediction, and progressive learning. The side information (entity names in this case) is used to compute a textual distance matrix which is forwarded to the unmatchable entity module to compute the basic entity alignment. The textual matrix combines a semantic distance matrix (cosine distance between entity names' averaged embeddings) and a string-level feature matrix (Levenshtein distance between entity names). The unmatchable entity module uses the thresholded bi-directional nearest neighbour search (TBNNS) to compute the entity alignments between two KGs. The progressive learning

framework repeats these steps as long as a number of newly aligned entities is above a threshold: First, the model uses GCN to learn entity embeddings. Second, the model uses TBNNS to align entities. The TBNNS threshold increases over time to allow an alignment between more distant embeddings.

[Liu et al., 2020] propose a method for unsupervised entity alignment incorporating images. The model trains joint multi-modal embeddings. The authors use graph structure embeddings, visual embeddings, relation and attribute embeddings. The graph structure embeddings capture both entity and relation proximity. They are trained on united entity and relation sets from KG1 and KG2 via a GCN. The visual embeddings use pre-trained image embeddings to feed a simple feed-forward neural network (FFNN) which outputs the final embeddings. The relation and attribute embeddings are the output of a simple FFNN. All these modalities are combined into a multi-modal representation via a trainable weighted concatenation. The alignment process computes the embedding similarity based on the cosine distance. The cosine distance matrix (rows are source entities, columns are target entities, values are cosine distances) is used by the neighbourhood component analysis (NCA) loss function to align entities. The NCA loss helps to cope with the hubness problem. The hubness problem is caused by entities which are very common and are nearest neighbours of many other entities, while rare entities are nearest neighbours to no entities. Therefore, during the alignment the "hubs" are preferred which lowers the final performance. The alignment process is done iteratively, in every round some seed alignments are added into the training set. In the unsupervised setting, the pseudo-seeds are generated based on the visual embedding similarity.

2.2.3 Neural Networks

[Wang et al., 2018a] propose a rigorous and theory-based approach to compare two neural networks based on simple matches between neurons. The authors use the activation function of a single neuron as a vector. Further, they try to find simple matches for fine-grained comparison and a maximal match for the comparison of clusters of neurons. This work can be seen as a comparison between two vector spaces. The authors use the same neural network architecture with different initialization. Thus, they obtain a one-to-one match between neurons. This is usually not true between two KGs. In our approach, we want to align two embedding spaces that can contain similar but also complementary information.

[Li et al., 2015] propose another approach to find similarities between neural networks. In a one-to-one setting, the authors use within-net or between-net correlation to find similar neurons. Their study shows that both networks learn the same core features, but different rare features. To confirm their results, the authors also try the same mapping with the mutual information measure which shows very similar results. In a many-to-many mapping, authors use the hierarchical spectral clustering method. Again, the authors find that every network contains similar core clusters (e.g. filters learning magenta objects) but other rare features are unique. The authors use four similar

architectures with different initialization.

2.3 Hybrid space alignment

[Wang et al., 2014] propose a model based on TransE and word2vec which can learn a joint embedding space of text and KG embeddings. The model has three parts: text model, KG model, alignment model. The text model embeds words using an algorithm similar to skip-gram. The authors look at word pairs as knowledge triples with unknown relations. To make the computation feasible in sense of a number of relations, the problem is defined as maximizing the concurrence of pairs of words in given text windows. The KG model implements the probabilistic TransE. This model defines the conditional probability of a triple based on a TransE-inspired triplet score. The alignment model compares two approaches: the alignment by Wikipedia anchors and the alignment by names of entities. The alignment by Wikipedia anchors leverage the knowledge that for the most Wikipedia pages there is a unique entity in Freebase. Therefore, most of the anchors in Wikipedia pages refer to a Freebase entity e_v by their surface phrase v. This enables us to crate the word-entity anchors (w,e_v) by replacing v in the original anchor pair (w,v). The alignment by name of entities uses a name graph. The name graph copies a KG but substitutes entities of a KG by corresponding words. The authors use negative sampling as a training strategy.

[Wu et al., 2018] present StarSpace, a general-purpose embedding model. This model can be used to embed KGs as well as textual data. It learns embeddings of entities that are described by a set of features, e.g. a sentence is described by BOW. These features come from a dictionary with a fixed length. The model consists of a positive example generator, a negative example generator, a similarity function, and a loss function. The positive example generator is task-dependent. The negative example generator is inspired by Mikolov's negative sampling. The similarity function is either the cosine similarity or inner product similarity. The loss function is either the margin ranking loss or the negative log loss of softmax. StarSpace can be used in a multitask setting if tasks share some features in the base dictionary. The authors experimented with a variety of tasks such as link prediction for KG, text classification, or sentence matching.

[Toutanova and Chen, 2015] show that enhancing latent models with observed and textual information yields better results in the KG completion task. The authors propose an observed feature model that captures more information from training data. This model is implemented as a variant of path ranking for KG completion. Moreover, the authors propose an approach in which textual relations can be treated the same way as relations in triples, and the same models can be used to capture both types of information.

[Fang et al., 2016] learns joint embeddings of KG entities and text, creates new features based on learned joint embeddings, and proposes a two-layers entity disambiguation model. The proposed system is named EDKate. The model used to learn aligned embeddings is based on a combination of three techniques, namely: Alignment by Wikipedia anchors, alignment by names of entities, alignment by entities' descriptions. Embeddings

learned in the alignment model are used to produce further features: Global context relatedness, local context relatedness, and local entity coherence. Their last contribution is a two-layer disambiguation model which balances the usage of mention-entity prior and other proposed features.

2.4 Evaluation methods

[Schnabel et al., 2015] present an overview of different evaluation methods aimed at comparison of unsupervised word embeddings. Authors evaluate embedding models based on four tasks: relatedness, analogy, categorization (clustering), and selectional preference. Another contribution is the creation of a comparative intrinsic evaluation. A query inventory for this task balances the frequency of the used word, part of speech, and concreteness. Amazon Mechanical Turk was used to perform the experiment. Turkers received one word and had to choose another one from the given options (different word embeddings). Moreover, the authors presented the *Coherence* task. This task studies if a local neighbourhood of words is semantically coherent. Turkers received four words and had to choose which one is from a different cluster. This task is built on the hypothesis that the intruder should be easy to spot because the close neighborhood is semantically similar.

2.5 Surveys

[Wang et al., 2017] compose a survey that presents and explains the most used KG embedding methods. The survey divides the embeddings based on the information used by the models. Explained models includes TransE and its derivatives, RESCAL and its extensions, HolE, ComplEx, SME, and others.

[Heist et al., 2020] compose an overview of knowledge graphs which are available on the web. The mentioned knowledge graphs are Cyc, Freebase, Wikidata, DBpedia, YAGO, CaLiGraph, BabelNet, and DbkWik. The survey contains a comparison of the knowledge graphs on different levels such as the number of instances, the number of edges, the average linkage degree, the number of classes and relations, overall complexity, and the average depth and width of the graph. Also the survey shows what information the graphs contain and how much overlap do they have.

[Ruder et al., 2019] present a survey on cross-lingual embedding models. The authors emphasize the fact that many models proposed in the literature are equivalent in the core approach and the only differences are hyper-parameter settings, optimization strategies, and such. The authors also critically examine the evaluation procedures of embedding models. The survey also contains description of multilingual extensions for cross-lingual embedding models and point on the challenging and unexplored research areas in the cross-lingual embedding domain.

2.6 Summary

In our literature review, we covered a large part of the work done on word and KG alignment. The aspects we focused on are the distance metric and the model used for alignment. Table 2.1 summarizes our findings.

The word alignment methods we considered use separately trained embeddings from two (or more) languages. The supervised approaches use the CCA distance measure or cosine similarity. The unsupervised approaches use CSLS or Wasserstein distance, which should prevent very common words from becoming hubs during the alignment. There are also methods that learn an alignment during the joint learning of embeddings. We don't include such models in our experiments because such an approach is not in the scope of this work. In the following, we will experiment with MUSE and Gromov-Wasserstein OT models. Both of these models use only pre-trained embeddings as input.

The KG alignment models are more uniform in the distance metrics they use than word alignment models. Two prevalent metrics are cosine similarity and energy function based distances. There are some custom distance functions usually connected with the use of GCNs. The models used for KG alignment in our review are either a linear transformation or GCN.

In our work, we want to experiment with KG alignment using word alignment methods. For the supervised methods, we want to exploit the orthogonal Procrustes method as it is not present in typical KG alignment approaches. On the other hand, all of the word alignment papers we considered uses orthogonal Procrustes instead of simple linear transformation.

We will also experiment with unsupervised word alignment methods MUSE and Gromov-Wasserstein OT. These methods use only embeddings as input. In contrast, typical unsupervised KG alignment models take advantage of additional, usually textual information such as entity labels, entity descriptions, or attributes. The textual information makes the initialization of the unsupervised method simpler compared to the initialization of the method based purely on embeddings. Moreover, these methods cannot be used in scenarios where we do not possess such data. In all of these experiments, we will exploit the use of different KG embedding models.

	Supervision	Joint learning	(KG embedding model)	Input	What part of input	Distance functions	Model	Evaluation
(Trisedya et al. 2019)	Unsupervised	Joint learning	TransE	KG	Entities, literals	Cosine similarity	Linear transformation	Hits@k, MR
(Wang et al. 2018b)	Supervised	Separate embeddings	Custom	KG	Entities, literals	Custom	GCN	Hits@k
(Xu et al. 2019)	Supervised	Separate embeddings	Word-based LSTM (entity names)	KG	Entities, topic entity graph	Custom (based on cosine)	GCN	Hits@k
(Liu et al. 2018)	Supervised	Separate embeddings	TransE	KG	Entities	Energy function, l1, l2	Linear transformation	Mean rank, hits@k
(Zhu et al. 2017)	Supervised	Joint learning	TransE	KG	Entities	Energy function, l1,l2	Linear transformation	Hits@k, MR
(Hao et al. 2016)	Supervised	Joint learning	TransE	KG	Entities	l1,l2	Linear transformation	Mean rank, hits@k
(Sun et al. 2018)	Supervised	Separate embeddings	Custom	KG	Entities, relations	Cosine similarity	Translation based	Accuracy, F1
(Zeng et al., 2021)	Unsupervised	Joint learning	Custom	KG	Entities, entity names	Cosine similarity, Levenshtein distance	GCN	Precision, Recall, F1
(Liu et al., 2020)	Unsupervised	Joint learning	Custom	KG	Entities, relations, literals, images	Cosine similarity, NCAloss, CSLS	GCN, ResNet, linear transformation	Hits@k, MRR
(Conneau et al. 2017)	Unsupervised	Separate embeddings	–	Words	–	CSLS	Two-player game, Orthogonal Procrustes	Precision@k
(Alvarez-Melis and Jaakkola 2018)	Unsupervised	Separate embeddings	–	Words	–	Gromov-Wasserstein	Orthogonal Procrustes	Precision@k
(Chen et al. 2018)	Supervised	Separate embeddings	–	Words	–	Cosine similarity	Orthogonal Procrustes	Clustering, analogy, semantics similarity
(Beyer et al. 2020)	Supervised	Separate embeddings	–	Words	–	Dimension-wise correlation of embeddings	CCA	CCA measure score
(Ammar et al. 2016)	Supervised	Separate embeddings	–	Words	–	Dimension-wise correlation of emeddings	MultiCluster, Multi-CCA	Mono/multilingual word similarity, word translation, monolingual/multi QVEC, monolongual/ multi QVEC-CCA
(Zhang et al. 2017)	Unsupervised	Separate embeddings	–	Words	–	Wasserstein	Orthogonal Procrustes	F1
(Grave et al. 2019)	Unsupervised	Separate embeddings	–	Words	–	Wasserstein	Orthogonal Procrustes	Precision@k, Accuracy
(Artetxe et al. 2018)	Unsupervised	Separate embeddings	–	Words	–	CSLS	Orthogonal Procrustes (basically)	Accuracy
Fang et al. 2016	Supervised	Joint learning	Custom	KG+text	Entities, words	Custom features	Disambiguation model	Accuracy, precision&recall
(Wu et al. 2018)	Supervised	Joint learning	–	Anything	–	Cosine similarity, inner product	Neural embedding	Hits@k
(Wang et al. 2014)	Supervised	Joint learning	TransE	KG+text	Triples, literals, documents	None	Maximum likelihood (triples score)	Accuracy
(Toutanova and Chen 2015)	Supervised	Separate embeddings, joint learning	E, DistMult	KG+text	Triples, textual relations	None	Latent features, observed features	MRR,hits@10
(Wang et al. 2018a)	Unsupervised	Separate embeddings	–	Neural network	–	None	Simple and maximum match	Maximum matching similarity

Table 2.1: Summary of related work.

3

Background

In this chapter we will describe concepts and chosen approaches in more detail.

3.1 Preliminaries

Knowledge Graph A Knowledge Graph consists of triples (h, r, t) where $h, t \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ are relations. A triple denotes some relation between head and tail entities. Entities are real-world objects such as people, places, or movies. We denote the source Knowledge Graph as $\mathcal{G}_S$ and the target Knowledge Graph as $\mathcal{G}_T$.

Word embeddings Word embeddings encode words as dense vectors in a continuous vector space. In general, embeddings can preserve contextual, linguistic, and structural information.

Word embeddings embed every word into a single embedding vector. These vectors have usually between 50 and 500 dimensions. The most used word embedding models are CBOW and Skipgram [Mikolov et al., 2013].

KG embeddings KG embedding models are trained on KG triples, and model entities and relations as dense vectors in the same vector space. We use four different KG embedding models. Our first choice is **TransE**, as described in [Bordes et al., 2013]. This model embeds entities and relations into a single space. A relation is seen as translation between the head and tail entities, e.g. $h + r \approx t$ holds if triple *(h,r,t)* is correct, meaning the triple is part of the embedded KG.

Another embedding model is **RotateE** [Sun et al., 2019] which is build on the same principle as TransE. The RotatE model assign every entity an embedding in a complex space and model relations as rotations. The distance is computed as:

$$d_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|, \tag{3.1}$$

where $\circ$ stands for element-wise product. This way RotatE can model also symmetric relations, which can't be modeled by TransE.

The third used model is **DistMult** [Yang et al., 2014]. This model is built on the RESCAL approach. RESCAL model entities as vectors and relations as matrices. DistMult forces relation matrices to be diagonal. The scoring function is:

$$f_r(h,t) = \mathbf{h}^\top \operatorname{diag}(\mathbf{M}_r)\mathbf{t}, \tag{3.2}$$

where $\mathbf{M}_r$ is a diagonal matrix of relation. This way the number of parameters is much smaller compared to RESCAL, but the model can only deal with symmetric relations.

The last used model is **RDF2vec** [Ristoski and Paulheim, 2016]. It uses the hypothesis of neural language models that words close in context are close in meaning. Thus the model creates graph "sentences" for an entity by random walks with the entity as root. These sentences has the structure entity - relation - entity - relation - ... The Word2vec algorithm is then used to compute dense vector representation for each entity.

3.2 Alignment methods

We implement three alignment methods. The $\mathcal{G}_S$ is transformed using these methods to vector space of $\mathcal{G}_T$. In this section A represents vector space of $\mathcal{G}_S$ and B represents vector space of $\mathcal{G}_T$.

Linear Transformation The benchmark method solves the supervised linear transformation problem as described in [Grave et al., 2019]. We choose some entities as anchors and train the transformation matrix M on them. To obtain the transformation matrix we solve the least squares problem:

$$\min_M \| AM - B\|_2. \tag{3.3}$$

The linear transformation without any constraints generally is the simplest model.

Orthogonal Linear Transformation Constraining the transformation matrix to be orthogonal is one way of potentially improving the alignment performance. An orthogonal matrix ensures that distances between embeddings remain the same. This property was shown to be important for alignment problems by [Xing et al., 2015]. We can restrict the transformation matrix to be orthogonal during the training, this way we solve:

$$\min_Q \| AQ - B\|_2, \tag{3.4}$$

where Q is orthogonal matrix. Alternatively, we can orthogonalize solution of the least squares problem:

$$Q = M(M^T M)^{-\frac{1}{2}}. \tag{3.5}$$

This way we obtain nearest orthogonal matrix to transformation matrix M. In our algorithm, we orthogonalize the linear transformation matrix M.

Procrustes Procrustes analysis finds the transformation matrix if the alignment is known. Many alignment approaches use the solution of Procrustes analysis in supervised methods, or after unsupervised initialization of the anchor points. The transformation matrix is usually restricted to be orthogonal, the corresponding orthogonal Procrustes solves the optimization problem given as:

$$\min_{Q \in \mathcal{O}_d} \parallel AQ - B\|_2^2,$$

where $\mathcal{O}_d$ is set of orthogonal matrices. This optimization problem has a closed form solution $Q^* = UV^T$, where $U\Sigma V^T = A^T B$ is an SVD decomposition [Schönemann, 1966]. Generally, the difference between the linear transformation and the Procrustes analysis is that Procrustes searches the best solution in a wider solution space, using all possible linear transformations (rotation, translation, scaling). Therefore, the Procrustes analysis is a technique to find the best linear transformation.

Optimal transport Optimal transport is an unsupervised approach for aligning two sets of points. It assumes two discrete distributions μ and ν over two sets of embeddings, with $\mathbf{p}$ and $\mathbf{q}$ as probability vectors. Probability vectors can be uniform or bring some additional information, e.g. reflect word frequencies. The transportation map is the alignment between the two sets of points. The original problem finds a transportation map T:

$$\inf_T \left(\sum_{i=1}^{m} \mathbf{p}_i \parallel \mathbf{x} - T(\mathbf{x}) \parallel \mid T_{\#}\mu = \nu \right) \tag{3.6}$$

where m is number of embeddings in two sets. $T_{\#}\mu = \nu$ ensures that the source embeddings maps exactly on target embeddings. In reality, there might be no such mapping. Therefore, the problem is relaxed to find a set of transportation plans $\{\Gamma\}$ which are defined as polytope Π dependent on $\mathbf{p}$ and $\mathbf{q}$. The polytope is a convex closure above all possible transportation plans. It can be specified as the set of solutions to a system of linear inequalities. The transportation plan is a matrix $\Gamma \in \mathbb{R}_+^{n \times m}$. The solution of discrete optimal transport problem then is:

$$\min_{\Gamma \in \Pi(\mathbf{p},\mathbf{q})} \langle \Gamma, \mathbf{C} \rangle, \tag{3.7}$$

where $\langle \Gamma, \mathbf{C} \rangle := \sum_{ij} \Gamma_{ij} C_{ij}$ is the total cost. $\mathbf{C}$ is a matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$. Equation 3.7 can be further regularized to yield strictly convex optimization problem [Cuturi, 2013].

Gromov-Wasserstein Distance The Gromov-Wasserstein framework proposed by [Alvarez-Melis and Jaakkola, 2018] operates on two sets of embeddings. The framework does not require them to be in the same vector space. The vector spaces are captured in within-domain similarity matrices $(\mathbf{C}, \mathbf{p})$ and $(\mathbf{C}', \mathbf{q})$. The loss function is $\mathbf{L} \in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, where $Ł_{ijkl} = L(C_{ik}, C'_{jl})$. The loss can be seen as a cost of matching i to j and k to

l. $Ł_{ijkl}$ is a 4-th order tensor. We need to find a coupling Γ between two spaces. The solution of Gromov-Wasserstein then is:

$$\mathrm{GW}(\mathbf{C}, \mathbf{C}', \mathbf{p}, \mathbf{q}) = \min_{\Gamma \in \Pi(\mathbf{p},\mathbf{q})} \sum_{i,j,k,l} Ł_{ijkl} \Gamma_{ij} \Gamma_{kl}. \tag{3.8}$$

However, this equation is very difficult to solve due to the non-linearity and non-convexionality of the objective. The problem can be optimized by first-order methods with solving the traditional optimal transport in every iteration. The optimal transport coupling Γ^* provides a likelihood that $(w_{src}^{(i)}, w_{trg}^{(j)})$ are translations of each other. Solving equation 3.8 also yields distance between languages if $L = L_2$.

This approach is suitable for small or medium sized datasets. For the large datasets, this approach is used to compute an initial mapping on a subset of the embeddings. This initial mapping is then used as the initialization for the OP as described above.

MUSE unsupervised In the thesis we further explore the unsupervised word alignment method called MUSE from [Conneau et al., 2017] in depth. In this paragraph we introduce it in more detail. The source and target embedding spaces are denoted $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Y} = \{y_1, ..., y_m\}$ respectively.

The unsupervised method has two parts: the domain-adversarial training and the Procrustes refinement procedure. The domain-adversarial method contains a mapping model and a discriminator. The mapping W tries to fool the discriminator from distinguishing between $W\mathcal{X} = \{Wx_1, ..., Wx_n\}$ and $\mathcal{Y}$. The mapping is initialized as identity matrix and further trained, with objective:

$$\mathcal{L}_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}\big(\text{source} = 0|Wx_i\big) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}|(\text{source} = 1|y_i), \tag{3.9}$$

where θ_D are discriminator parameters, and $P_{\theta_D}(\text{source} = 1|z)$ is the probability that a vector z is the mapping of a source embedding. The mapping matrix W is orthogonalized with the equation:

$$W = (1 + \beta)W - \beta(WW^T)W. \tag{3.10}$$

The discriminator try to predict correct origin of the embedding, distinguishing between $W\mathcal{X}$ and $\mathcal{Y}$. The discriminator objective is:

$$\mathcal{L}_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}\big(\text{source} = 1|Wx_i\big) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}\big(\text{source} = 0|y_i\big). \tag{3.11}$$

The output of domain-adversarial training is a one-to-one mapping between source and target embedding spaces. This mapping creates an artificial dictionary between source and target words. The dictionary can contain all words or the words with higher frequency as in [Conneau et al., 2017] to then create the more accurate mapping.

The refinement procedure takes this dictionary as input for supervised Procrustes as described above. The result of the Procrustes method can be used to generate a more accurate dictionary and use it again as input for Procrustes. This way the refinement is iterated.

4

Experimental setup

In this chapter we describe the technical details of our work. The section 4.1 introduces used datasets. We describe the embedding training procedure in Section 4.2. Further, we describe technical details of supervised and unsupervised alignment methods we used in Section 4.3. In Section 4.4 we describe metrics we use to evaluate our experiments.

4.1 Datasets

In our experiments we used two knowledge graphs:

1. fb15k-237,
2. wd15k-237.

fb15k-237 is a subgraph of a larger KG Freebase. The fb15k-237 version contains 14'541 entities and 237 relations. The number of training triples is 272'115. The inverse relations were removed because a lot of test triples could be obtained by inverting training triples [Toutanova and Chen, 2015, Bordes et al., 2013].

wd15k-237 is a subgraph of Wikidata KG. It contains 14'295 entities and 322 relations, and 121'904 training triples. This version of Wikidata KG was created by Mattias Baumgartner. The ground-truth alignment between the entities of wd15k-237 and fb15k-237 is from GitHub site "Datasets for Knowledge Graph Completion with Textual Information about Entities"[1]. We also created the alignment between relations of above mentioned KGs.

4.2 Embedding training

We decided to use four different embedding models:

1. TransE,
2. RotatE,

[1] *https://github.com/villmow/datasets_knowledge_embedding*

3. DistMult,
4. RDF2vec.

The LibKGE[2] library, which is described in [Ruffinelli et al., 2019], was used as a training framework for all embedding methods but RDF2vec. LibKGE uses configuration files to specify the hyperparameters for the embedding training. The library also provides finetuned configuration files for some KGs, fb15k-237 being one of them. We reused the best configuration files for TransE, RotatE, and DistMult on fb15k-237 and used them without further finetuning for both datasets. We endeavored to finetune the hyperparameters for wd15k-237, but this effort was abandoned because the procedure to obtain the best hyperparameters was rather complex and required large amount of resources and time. Therefore, we trained wd15k-237 embedding models with fb15k-237's best configuration files and after obtaining similar link prediction results as for fb15k-237 we decided to keep these models.

The RDF2vec embeddings were trained using pyRDF2vec library[3] introduced in the paper by [Vandewiele et al., 2020]. We trained the fb15k-237 and wd15k-237 embeddings with multiple configurations of hyperparameters.

TransE TransE gives us an intrinsically consistent embedding space. But every run of the model on a dataset will output different embedding space. For the fb15k-237 dataset, LibKGE offers a pre-trained TransE model. Further, we trained one additional embedding for fb15k-237 and two for wd15k-237. This results in two models per dataset which can be perfectly aligned but occupy different embedding spaces. The configuration file was the best configuration for fb15k-237 (which is the same as for the pre-trained model). Our TransE models have 128 dimensions. The negative sampling is used during the training, the batch size is 128 during the training and 256 during the validation. The Adam optimizer is used with early stopping after ten epochs without improvement. We trained on a CPUs, the pre-trained model was trained on GPUs.

In our experiment, we want to align TransE to other embedding models with a higher number of dimensions. Therefore, we also trained a TransE model for fb15k-237 with 256 dimensions. Other hyperparameters remained the same as for the 128-dimensional model.

For the scenario in which you can not train another embedding model but still want to align models with various dimensionality, we append 128 random values close to zero to the TransE model with 128 dimension. This way, we obtained 256-dimensional embeddings without further training.

RotatE As for TransE, we reused the pre-trained model for fb15k-237 and further trained one model for fb15k-237 and two models for wd15k-237 with the configuration file from the pre-trained model. The RotatE embeddings have 256 dimensions. Negative sampling is used during the training, the batch size is 128 during the training and 256

[2] *https://github.com/uma-pi1/kge*
[3] *https://github.com/IBCNServices/pyRDF2Vec*

	max depth (d)	max walks (w)	max epochs (e)
setting 1	4	10	10
setting2	4	20	100
setting3	8	10	100

Table 4.1: RDF2vec hyperparameter settings used in the experiments.

during the validation. The embedder uses the L_2 norm. Early stopping is used after ten epochs without improvement. Our models were trained on CPUs, the pre-trained model was trained on GPUs. The RotatE models required the most time to be trained.

DistMult The setting is similar to RotatE. We used one pre-trained model for fb15k-237, further we trained one model for fb15k-237 and two models for wd15k-237 with the same configuration as the pre-trained model. The DistMult embeddings have 256 dimensions. Negative sampling is used during the training, a batch size of 1024 was used during the training, and 256 during the validation. The training algorithm uses early stopping with a patience of ten epochs. We trained our models on CPUs, the pre-trained model was trained on GPUs. DistMult was the fastest model to train.

RDF2vec RDF2vec embeddings have 100 dimensions. They were used only in the MUSE unsupervised experiments to verify the hypothesis about preferred embedding models. We used three different hyperparameter settings and trained two models with each setting, one with fb15k-237 and one with wd15k-237. The hyperparameters we changed between the trainings are the maximal depth of a walk (d), the maximal number of random walks (w), and the maximal number of epochs (e). In our experiments we use multiple runs of three different hyperparameter settings summarized in Table 4.1.

4.3 Alignment methods

The datasets are divided into a 20% test and 80% training set. The test set is common for all experiments. The training set is common for all supervised experiments. The unsupervised experiments required all entities $\mathcal{E}$ as input.

We also experimented with different sizes for the training set for supervised experiments. Therefore, we chose random entities from the training set to be 10%, 30%, 50%, 60%, and 70% of the whole entity set $\mathcal{E}$.

We were also curious if the more frequent entities would create better unsupervised alignment. For these experiments we ordered entities by the number of triples they were in and chose the most or least frequent 100, 200, 500 and 1'000 – 8'000 entities.

4.3.1 Supervised alignment methods

Linear transformation (LT) Our implementation of linear transformation uses the PyTorch least squares[4] function to compute the transformation matrix M. Only the training set is used for the computation. Then, all source embeddings are translated into the target embedding space and compared to the original target embeddings.

Orthogonal linear transformation (OLT) The orthogonal linear transformation implementation also uses the least squares function to compute the transformation matrix. The obtained matrix is then orthogonalized, i.e. the closest orthogonal matrix is computed using Equation 3.10. The procedure is otherwise the same as for the linear transformation.

Orthogonal Procrustes (OP) The orthogonal Procrustes uses the scipy *orthogonal_procrustes* function [5]. This function uses orthogonal transformations such as rotation and reflection. The result is a square matrix with dimension $n \times n$, where n is the dimension of embedding.

MUSE supervised (MUSE-s) The supervised MUSE algorithm uses a set of anchor points as the initial dictionary between two KGs. It then loops through computing orthogonal Procrustes on a given dictionary and enhancing the dictionary with the results. Then, it uses the enhanced dictionary as input in the next loop. This is called iterative Procrustes. The default setting runs five such loops. One very important hyperparameter is the method used for computing alignment dictionaries. The default setting is CSLS, which can be changed to nearest neighbour search or inverted softmax. Another hyperparameter is the maximum size of vocabulary, which is set to 200'000.

4.3.2 Unsupervised alignment methods

MUSE unsupervised (MUSE-u) MUSE [Conneau et al., 2017] is one of the state-of-the-art algorithms in unsupervised word embedding alignment. We adopted this approach and computed the alignment for KG entity embeddings without any changes. MUSE unsupervised has two parts:

- adversarial training,
- Procrustes refinement procedure.

The ideas behind these parts are described in Section 3.2 in more depth. In this paragraph, we will describe the hyperparameter settings.

The adversarial part tries to find a mapping between $\mathcal{G}_S$ and $\mathcal{G}_T$. This mapping is initialized as an identity matrix. The matrix is orthogonalized after every epoch with

[4] *https://pytorch.org/docs/stable/generated/torch.lstsq.htmal*
[5] *https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.orthogonal_procrustes.html*

$\beta = 0.001$ as default. The optimizer of the mapping is the stochastic gradient descent with a learning rate of 0.1.

The discriminator is a small neural network with two layers with 2048 hidden dimensions. The default optimizer is the stochastic gradient descent with a learning rate of 0.1. The input dropout is 0.1. The discriminator runs five steps every epoch. The discriminator loss feedback coefficient is set to 1 which means that the loss is not changed at all before the backward run, if it would be less than 1 the loss would be smaller than original loss. For our purposes, we disabled choosing of the most frequent words by the discriminator, because we did not provide such information in embedding order. The discriminator smooths predictions by default by 0.1.

Overall, the adversarial training runs for five epochs with one million iterations in each epoch. The batch size is 32. The learning rate decay is 0.98 with a minimum learning rate of $1E^{-6}$. If the validation metric decreases, the learning rate is shrunk by a 0.5 ratio.

The refinement procedure has five iterations as a default setting. Other hyperparameters of iterative Procrustes are the same as for supervised MUSE – CSLS as dictionary creation method, 20'000 maximum size of the vocabulary.

OTalign (OT) The OTalign[6] algorithm implements the Gromov-Wasserstein distance with optimal transport as described in Section 3.2. The algorithm transforms both source and target embedding spaces into some other common vector space. The default metric is cosine similarity. Otherwise, OTalign has only one scalable parameter, the sinkhorn entropy regularization λ. The Sinkhorn-Knopp algorithm is a matrix-scaling procedure that is used in a loop after computing a pseudo cost matrix $\hat{C}_\Gamma$. The larger the λ, the denser the optimal coupling Γ^*. Weaker regularization leads to sparser solutions but the optimization problem becomes more non-convex thus harder to solve.

4.4 Evaluation

In our setting, we are trying to measure how well the alignment method works between two knowledge graphs $\mathcal{G}_S$ and $\mathcal{G}_T$. Common metrics used in KG evaluation are Hits@k, Mean Rank, Mean Reciprocal Rank, and Adjusted Mean Rank [Ruffinelli et al., 2019, Ali et al., 2020].

At the beginning of every evaluation method, we transform an entity embedding from the $\mathcal{G}_S$ embedding space into the $\mathcal{G}_T$ embedding space. Then, we measure the distance between the transformed embedding from $\mathcal{G}_S$ to all of $\mathcal{G}_T$'s original embeddings. In the following we will discuss the above mentioned metrics in detail:

- Hits@k – The original $\mathcal{G}_T$ embeddings are sorted by their distance to a $\mathcal{G}_S$ embedding and the k closest embeddings are chosen. If the real alignment of the $\mathcal{G}_S$ embedding is within the k closest $\mathcal{G}_T$ embeddings the model earns a point. This

[6] *https://github.com/dmelis/otalign*

process is done for all the $\mathcal{G}_S$ embeddings from the test set. This metric tells us the probability that the true target entity of an embedding is in the k nearest neighbours of the transformed source embedding. The higher the better.

- Mean Rank (MR) – $\mathcal{G}_T$ embeddings are sorted by the distance from the transformed embedding. The rank (position) of the real translation of the original $\mathcal{G}_S$ embedding is gathered and is summed for all the test embeddings from $\mathcal{G}_S$. The sum of all the ranks is divided by the number of test entities from $\mathcal{G}_S$. The result is the average rank of the correct translation from $\mathcal{G}_S$ to $\mathcal{G}_T$. The lower the better.
- Mean reciprocal rank (MRR) – the procedure is the same as for MR, but we sum the reciprocal ranks. The reciprocal rank is one divided by the actual rank of an embedding. If the real translation is always first, then the result is one. If the real translation is always last, the result is close to zero.
- Adjusted Mean Rank (AMR) – is adjusted to the number of entities in the different KGs. AMR makes the results comparable between different datasets. In MR, every single transformed entity is compared with all the entities from $\mathcal{G}_T$. If we sum the number of entities every single transformed embedding is compared to, and divide MR by this number, we get the AMR. The equation is AMR $= \frac{MR}{\frac{1}{2}\sum_{t \in \mathcal{K}_{test}} \mathcal{E}}$.

  Therefore, we can compare alignment results between different KGs.

Some of the experiments are evaluated in two directions:

- transformed $\mathcal{G}_S$ to $\mathcal{G}_T$,
- $\mathcal{G}_T$ to transformed $\mathcal{G}_S$.

This way we can see if the performance of the algorithm is balanced, if it is similar for both evaluation directions, or not.

5

Results

In this chapter we present our research questions about KG alignment and describe the hypothesis and experimental results that offer an answer to these questions.

We identified two research questions:

1. Do the word alignment methods used on KGs have results in the same order of magnitude as typical KG alignment methods?

2. Do the alignment between different KG embedding models yield results in the same order of magnitude with alignment between the same embedding models?

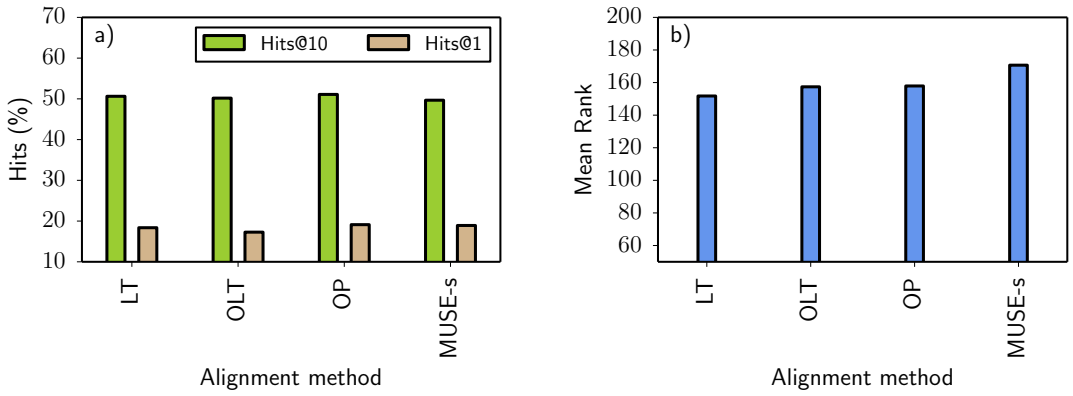
The motivation behind the first question is that the area of word alignment is older than for KG alignment. During our related work research, we noticed that e.g. LT is common for KG alignment but for word alignment, the orthogonal constraint is typically applied. Seeing such behaviour, we decided to apply some text alignment methods to KG alignment and evaluate their usefulness.

The second question is motivated by a real-life scenario. For large KGs, it is pricey to compute an embedding model with good performance. But at the same time, it is unlikely that two different KGs are embedded by the same embedding model. Therefore, we test whether the performance decreases dramatically when aligning two different embedding models.

We report the performance in terms of hits@{1,10} and mean rank (MR). The hits scores show if the entities are aligned in the correct neighbourhoods. The MR shows if the algorithm aligns all of the entities well, or just some subset of them.

5.1 RQ1: Word alignment methods on KG data

In this section, we describe our hypotheses about the performance of word alignment methods on the KG embedding spaces. In Subsection 5.1.1 we compare LT as typical KG alignment method with OLT and OP as typical supervised word alignment methods, and MUSE-s as state-of-the-art supervised word alignment method. In Subsection 5.1.2 we discuss the effect of the size of a training set on performance. In Subsection 5.1.3 we compare unsupervised methods MUSE-u and OT with the supervised ones, and we dive a little bit deeper into the MUSE-u algorithm.



(a) Hits@1 and hits@10 results

(b) Mean rank results

Figure 5.1: The performance of alignment between fb15k-237 to wd15k-237 both embedded by TransE. The reported results are average of the evaluation directions.

5.1.1 Orthogonality restriction

The orthogonalization of the transformation matrix causes an increase in the alignment performance. According to the literature, orthogonality in the word alignment approaches helps the alignment method to achieve higher hits@k than LT. The orthogonality of the transformation matrix M keeps the distances between the embeddings the same even after the transformation. Then, the distribution of the embeddings in the space stays the same. This helps the alignment to be more accurate if the two word embedding spaces have a similar distribution. We want to verify if KG alignment has the same behaviour.

To test this hypothesis we compare the LT to OLT, OP, and MUSE-s. We experiment with aligning fb15k-237 to wd15k-237 using the TransE embedding model. We measure hits@1, hits@10, and MR to see if the method is trying to align all of the entities or if it just chooses some entities easier to align and ignores the others.

Figure 5.1 shows the average hits@{1,10} and mean rank of fb15k-237 to wd15k-237 alignment with TransE embeddings. The average refers to the average between source to target and target to source evaluation directions. All the methods are supervised and trained with the same 50% training set and evaluated with the same 20% test set.

Figure 5.1a shows that the LT is on par with the orthogonal alignment methods in terms of hits. All of the methods have the average hits@10 around 50% and average hits@1 around 20%. Figure 5.1b shows the mean rank results. The LT has the best performance in terms of mean rank. The difference between LT and OLT is 6 ranks, it is the same as between LT and OP. The MUSE-s has the worst MR of 170. MUSE-s chooses a subset of entities that are easier to align and focuses on them while training the alignment and ignores the entities that are more difficult to align. This results in a lower MR score but similar hits@10 when compared with other orthogonal alignment

methods.

However, the average results are not always the same for all alignment methods (compare Figure 5.1 and Figure 5.2). The results depend on the embedding model used for the alignment.

The orthogonality did not bring expected improvement in the performance of the alignment methods. This can be caused by different distributions of entities in embedding spaces. The main idea why orthogonalization improves the performance in the word alignment is that the distributions of two languages are similar. The proof of such an assumption for KG embedding spaces requires further experimentation.

An iterative orthogonal Procrustes of MUSE-s improves results. Our implementation of KG alignment uses a very simple OP algorithm. We expect MUSE-s to perform better because of the iterative approach, where the alignment dictionary is enhanced in every iteration. Also, the CSLS metric used to compile the dictionaries uses only mutual nearest neighbours which causes the dictionaries to be more reliable.

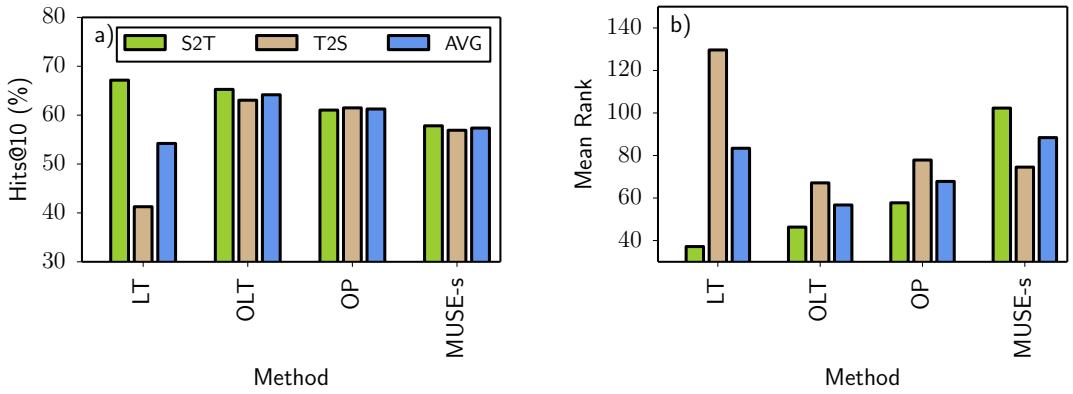
We compare all orthogonalized methods in terms of hits@{1,10} and MR. The reported results are the average of both evaluation directions. Figure 5.1 shows that the MUSE-s algorithm performed worst in our setting in both measured metrics, being even worse than LT. The difference between hits of MUSE-s, OP, and OLT is not substantial. On the other hand, the difference in MR is 13 ranks (from 157 to 170) between LT and MUSE-s.

The KGs are probably very sensitive to the wrong alignment and mistakes from one iteration propagate through the whole refinement procedure. The worst performance in mean rank is caused by the CSLS metric. This metric chooses the mutual nearest neighbours into the alignment dictionary. Therefore, some entities are never added which causes the alignment to be shifted towards the entities in the dictionary. The more common or easier to align entities are preferred while the marginal entities are not considered.

The orthogonal alignment methods have similar performance for both evaluation directions, the LT does not. Because the nearest neighbour metric is not symmetric, we want to see if the transformed source embeddings are unambiguous for both evaluation directions or not. If the alignment is balanced, meaning that the hits@10 is almost the same for both evaluation directions, then the correctly aligned source entities are mutual nearest neighbours of the true target entities.

We align wd15k-237 to fb15k-237 embedded by the RotatE model. To compute the alignment we use all supervised embedding methods. For the training, we use a 50% training set and for evaluating our common 20% test set. The hits@10 and MR are measured for both evaluation directions, from transformed source to target (S2T) and from target to transformed source (T2S). We also report the average between these evaluation directions (AVG).

In Figure 5.2a we show the hits@10 score of the different evaluation directions of all supervised alignment methods. We observe that the direction of evaluation changes



(a) Hits@10 results

(b) Mean Rank results

Figure 5.2: Comparison of evaluation directions of LT, OLT, OP, and MUSE-s. The aligned datasets are wd15k-237 to fb15k-237 embedded by RotatE. We use a 50% training set. The S2T stands for the evaluation direction from transformed source to target, T2S stands for the evaluation from target to transformed source, AVG is the average of these two directions.

results substantially for LT. LT performs better when evaluated from transformed $\mathcal{G}_S$ to $\mathcal{G}_T$. OLT, OP, and MUSE-s have similar results for both directions with differences of about one percentage point. The average between the evaluation directions has the lowest performance for LT caused by the imbalance.

Figure 5.2b shows the mean rank results. LT has the best S2T results. MUSE-s has the worst average results. It is the only method that has better hits@10 for S2T evaluation but worse MR. Overall, the difference in mean rank is more visible than for hits@10 for all methods.

The results show that for LT the transformed source embeddings are nearest neighbours of correct target embeddings. Therefore, the correct pairs are easily found. In contrast, the target embeddings are not mutual nearest neighbours with correct source embeddings.

For the orthogonal alignment methods, the alignment neighbourhoods are more similar. The correctly aligned entities are mutual nearest neighbours with their target entity pairs. Therefore, there is no significant difference between evaluation directions in means of hist@10.

OLT has an interesting behaviour. The method computes the transformation matrix M in the same way as LT and then finds the nearest orthogonal matrix of M. Even this simple constraint balances the overall performance. However, it pays the price of a small performance decrease.

5.1.2 Hyperparameters changing performance

The performance increases with the number of anchor points. It is laborious to obtain ground truth alignment between two KGs. Therefore, we want to know how the performance depends on the number of anchor points, i.e. if there is some boundary after which the results won't get significantly better, or if there is linear-like relation between the performance and the number of anchors.

To test this hypothesis we create training sets of 10%, 30%, 50%, 60%, 70%, and 80% of the anchors between fb15k-237 and wd15k-237 (further called training sets). Both datasets are embedded with TransE. The entities for the training sets are chosen randomly. The remaining 20% is used as a test set for all of the experiments. The training sets are disjoint with the test set. We measure average hits@10 and hits@1.

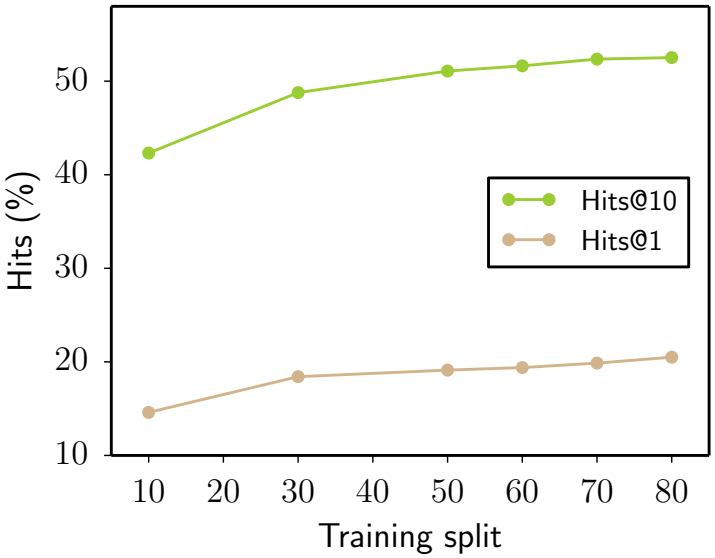
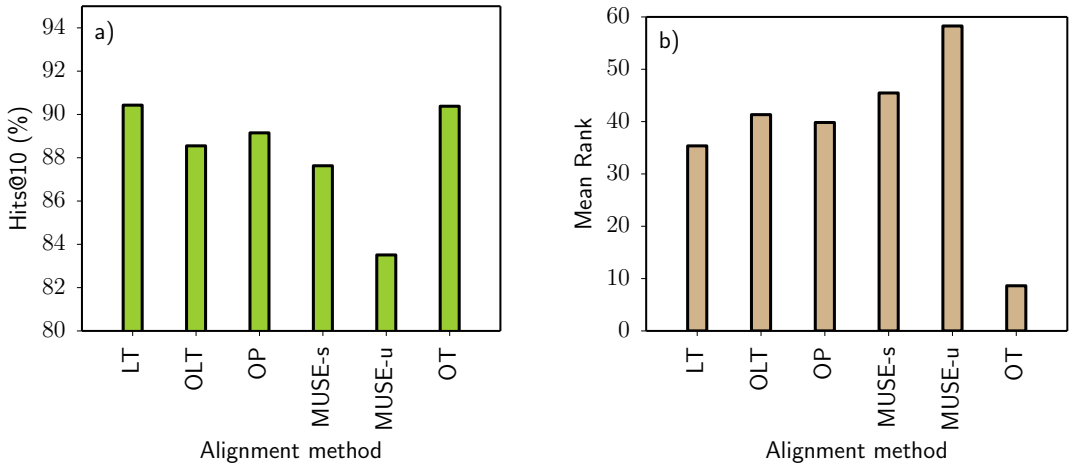


Figure 5.3: Comparison of 10%, 30%, 50%, 60%, 70%, and 80% training sets of fb15k-237 to wd15k-237 alignment. Both KGs are embedded with TransE. OP is used as the alignment method. The size of the training set is the percentage part of the whole entity set $\mathcal{E}$.

Figure 5.3 shows that with more anchor points the results are always better. But, with 50% and more, there is not a big difference in hits@10 (the difference is one percentage point between 50% and 80%). This means that in a real-world scenario after having 50% of the entity set $\mathcal{E}$ as anchor points, the results won't improve much. However, for large KGs 50% is already a substantial number of anchor points. The results show that the difference between 10% and 50% training set is nine percentage points (42% vs 51%). Therefore, even for the small training sets the performance remains in the same order of magnitude.

5.1.3 Unsupervised text alignment algorithms

First, we quickly summarize the unsupervised algorithms. MUSE-u applies adversarial training to train a preliminary mapping between two spaces and then continues as MUSE-s with the iterative orthogonal Procrustes refinement procedure. The OT algorithm first computes intra-language similarities. Then in the loop, the algorithm computes a pseudo cost matrix (the loss of the mapping). The matrix is used as input to the Sinkhorn algorithm which outputs a transportation map Γ (the mapping between two spaces) which is again used to compute the pseudo cost matrix. This loop stops when Γ converges to the optimal mapping. For large datasets, the optimal mapping is learned for a smaller subset of the entities. Then this optimal mapping is used as the initial dictionary for the Procrustes-like algorithm.

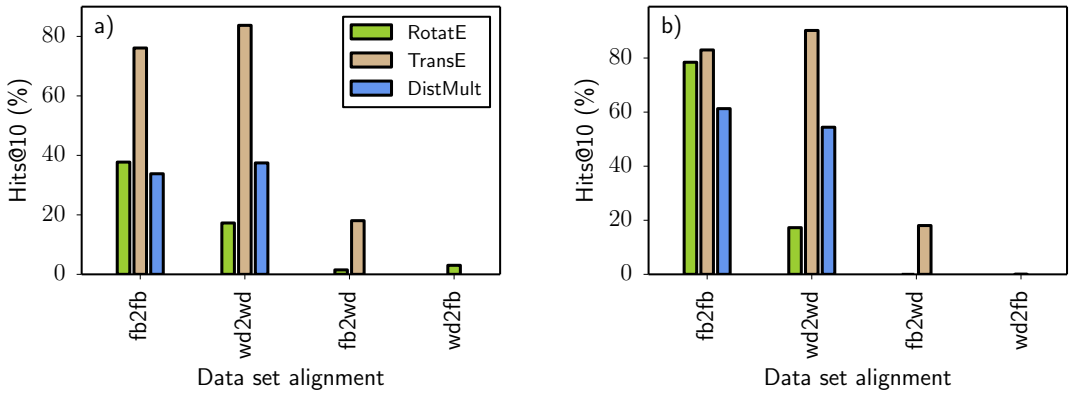


(a) Hits@10

(b) Mean rank

Figure 5.4: The hits@10 results of wd15k-237 to wd15k-237 TransE alignment. The embedding models are two separate runs of TransE. In this figure we compare all alignment methods.

When applied on KGs, unsupervised methods for text alignment have a performance at least on par with OP. The unsupervised methods are very useful in real-world scenarios. The KGs on the internet are changing quickly and to have current alignment between them is very costly and requires human resources if the alignment should be correct. It is very convenient to have an unsupervised algorithm to align two different KGs without having ground-truth data. The MUSE-u and OT perform on par with supervised methods for word alignment. Likewise, we expect the performance to be at least as good as for OP when aligning KGs. This is because both of the unsupervised methods use OP as a refinement procedure after finding initial mapping.



(a) MUSE unsupervised

(b) Gromov-Wasserstein OT

Figure 5.5: The performance of unsupervised methods varied from random baseline (hits@10 below one percent) to results on par with OP (wd2wd TransE hits@10 90%). The x-axis shows which two KGs are aligned. The color of the bar shows which embedding models are used to embed both of the datasets.

We experiment with two unsupervised algorithms MUSE-u and OT. Both of these algorithms take the whole entity sets of $\mathcal{G}_S$ and $\mathcal{G}_T$ as an input. It is important to note that in this experiment we align the same dataset, embedded by two different runs of the same embedding model, i.e. wd15k-237 to wd15k-237 embedded by two different TransE model instances. The test set is the same 20% of the entity pairs as for the supervised experiments. In our experiments, we measure the average hits@10 and MR to compare unsupervised algorithms with supervised ones.

We observe very interesting behaviour. In some cases when we use the same KG and the same embedding method the results of unsupervised alignment methods are on par with supervised results as is shown in Figure 5.4. The MUSE-u has the worst hits@10 but it is still above 83%. OT is on par with LT with hits@10 slightly above 90%. OT MR is 8.63 what is the best average score obtained in wd15k-237 to wd15k-237 alignment.

In Figure 5.5 we show results of unsupervised methods on different KG pairings embedded by the same embedding model. The TransE embedding models performed best for both datasets and for both unsupervised alignment methods. The MUSE-u alignment between fb15k-237 RotatE embedding models (hits@10 37%) yields similar performance to alignment between fb15k-237 DistMult models (hits@10 34%). The OT alignment of fb15k-237 RotatE models results in hits@10 78%, similar performance to TransE models with 82% hits@10.

The alignment between wd15k-237 DistMult models yields better results than between wd15k-237 RotatE for both MUSE-u and OT. The only embedding model which yields results above 1% for alignment between fb15k-237 and wd15k-237 is TransE with a

performance of about 19% hits@10 for both methods.

The results show that two different KGs are not trivially aligned by the unsupervised methods. This points out that it is not a simple task to obtain initial mapping with only embedding spaces.

A deep dive into unsupervised MUSE

In the following, we chose to further experiment with MUSE unsupervised and potentially find the reason why the alignment is not working as well as the word alignment results stated in [Conneau et al., 2017]. We chose MUSE-u because it also has a supervised mode against which we can compare the results. We observed that the supervised part of the unsupervised setting (after obtaining pseudo anchors from the unsupervised adversarial training) which implements orthogonal Procrustes enhances results even from a small set of anchors. Therefore, we decided to further only test the adversarial part (see Section 3.2) to find the reason why the unsupervised alignment is unable to align at least 10% of entities correctly (which is enough for results around 40% hits@10 as we know from previous experiments, as seen in Subsection 5.1.2).

The hits@10 and MR stated in the experiments described in this section are the results of the adversarial part of MUSE unsupervised if not stated otherwise.

Unsupervised alignment of two pre-aligned embedding spaces has a performance on par with a supervised method. We hypothesize that the different KGs might have too dissimilar spaces. So, it is too difficult for the MUSE-u to find an initial alignment passed to the iterative OP. Therefore, already aligned spaces should pose a much simpler problem and the results should be similar to the MUSE-s from which we take the already aligned embedding spaces.

MUSE-s outputs the source embedding space transformed into the target space. The target space remains the same. We used these two spaces (transformed source space and target space) as input to MUSE-u. Since we are interested in the "different KG, same embedding" scenario we will focus on fb15k-237 to wd15k-237 alignment. TransE was used as an embedding method for both KGs.

We measured hits@10 and MR. In this case, the results are after the whole MUSE-u training including Procrustes refinement. The reported numbers are the average of both evaluation directions.

In Figure 5.6 we see that the difference between the supervised results and the unsupervised with initialized embeddings is substantial. The results of the initialized experiments are similar to the results of MUSE-u without transformed source embedding space.

The hypothesis is rejected. A reason for this might be that the input spaces were too close to each other so the problem was too simple for MUSE-u adversarial training. The MUSE-u transformation matrix is initialized as an identity matrix. If it remained the same through the whole training the results would be the same as for MUSE-s. But the adversarial training changed the identity matrix to some other linear transformation. In

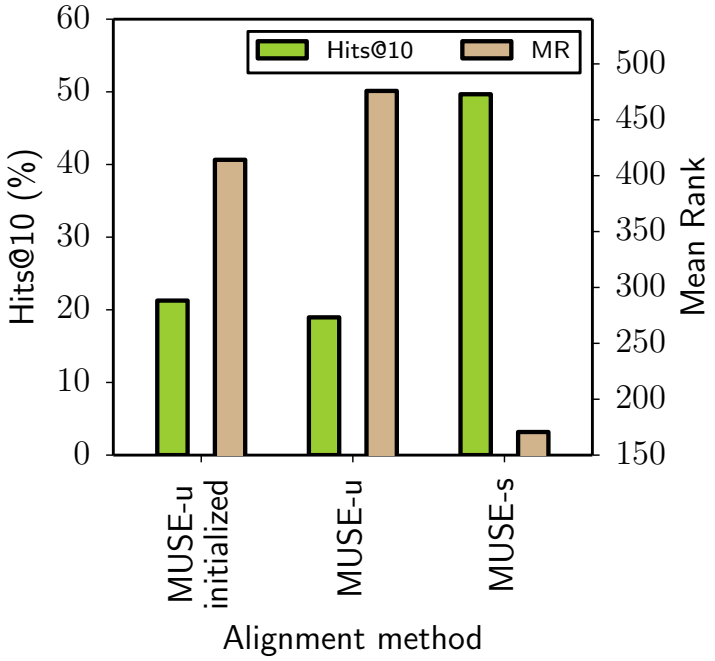


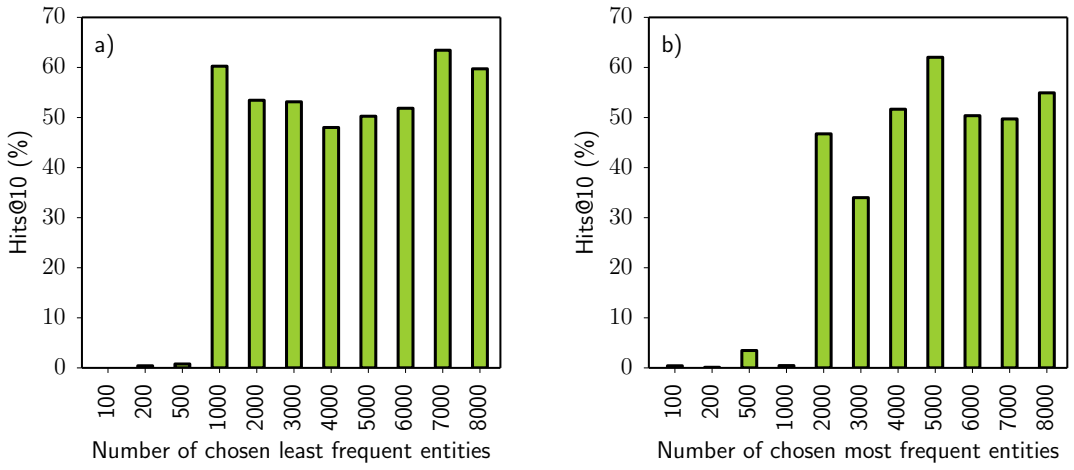
Figure 5.6: Results of MUSE-u with input from MUSE-s compared with original MUSE-s 50% training set and MUSE-u results, alignment between fb15k-237 to wd15k-237 embedded by TransE.

the end, the entities which should be aligned were further from each other than at the beginning of the training.

An alignment trained with the entities that occur most frequently in KG triples have higher performance than the alignment trained on the entities with the rarest occurrence. In [Conneau et al., 2017] the authors remove the least frequent words from the dictionary during the training because they are lowering the performance. We want to test this setting. The number of occurrences of the entity in the training triples, further denoted as entity frequency, reflects how many connections with other entities it has. More frequent entities capture the structure of the KG better. Also, embeddings of more frequent entities are iterated more often in the embedding training. Therefore, they are trained better than embeddings of less frequent entities.

To validate this hypothesis we experiment with wd15k-237 to wd15k-237 alignment. We used two separate runs of TransE. We chose to experiment with the alignment between the same KG because the results are about 70% hits@10 for the 50% training set, so we have a wider range of possible performances.

We select the 100, 200, 500, 1'000 – 8'000 most frequent and also the same numbers of the least frequent entities. We measure hits@10 of the same test set as for other



(a) Least frequent entities

(b) Most frequent entities

Figure 5.7: The x axis shows the number of training entities of MUSE-u. We align wd15k-237 to wd15k-237 embedded by TransE. We report average hits@10.

experiments. In this setting the entities from test set might occur in the training set.

In Figure 5.7a we see that including the least frequent entities in the training set is beneficial for the alignment. The performance jumps from random to 60% hits@10 for as little as 1'000 least frequent entities. The performance of least frequent entities with 1'000 and more anchors varies from 48% to 60% hits@10. The performance of the most frequent entities with 2'000 and more anchors varies from 34% to 62% hits@10.

In the Figures 5.8 and 5.9, we see the TSNE plot of transformed source wd15k-237 embedding space and target wd15k-237 embedding space. The grey dots represent the target space. The red dots represent the correctly aligned entities in terms of hits@10, and the blue dots represent the wrongly aligned entities in terms of hits@10.

There are three main clusters and multiple smaller ones. In Figure 5.8 with M trained on the 8'000 most frequent entities, the correctly aligned entities (red dots) form visible small clusters. While in the Figure 5.9 with M trained on 8'000 least frequent entities, the correctly aligned entities are distributed more evenly.

The hypothesis is rejected. When using the least frequent entities in the training set we generalize the margin areas better, so the transformation fits the target space better. While with the most frequent entities the margin areas are not taken into account during the training and the correctly aligned entities form small clusters.

The RDF2vec embedding model performs better than other KG embedding models when using MUSE-u. MUSE-u originally used fastText[1] embeddings. RDF2vec creates random graph walks which are represented as strings, then trains

[1] *https://github.com/facebookresearch/fastText*

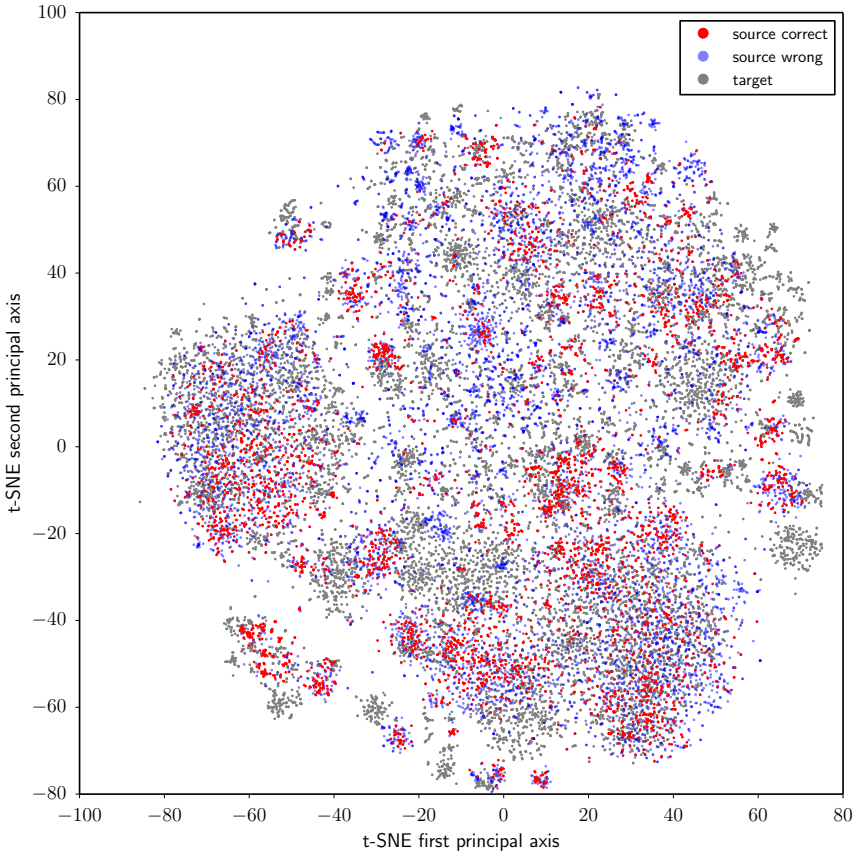


Figure 5.8: Alignment of two full wd15k-237 TransE embedding spaces with mapping computed on 8000 most frequent entities. Source correct refers to the correctly aligned source entities, source wrong refers to the wrongly aligned source entities, target refers to the entities from original target space.

word2vec embeddings on these strings. We expect the RDF2vec embedding spaces to be more similar to fastText embedding spaces than TransE or RotatE ones. Therefore, we expect the results for fb15k-237 to wd15k-237 alignment to be better than for other embedding methods.

We experiment with four combinations of datasets: fb15k-237 to fb15k-237, wd15k-237 to wd15k-237, fb15k-237 to wd15k-237, and wd15k-237 to fb15k-237. We embed the source and target KG by the two instances of RDF2vec with the same hyperparameter setting. The hyperparameters we change are: d – the max depth of a walk, w – the max number of walks, and e – the max number of epochs.

Figure 5.10 shows hits@10 results for three different RDF2vec settings. For the individual alignment, we used two RDF2vec models trained with the same hyperparameter setting. The results show that the alignment behaviour is the same as with other em-

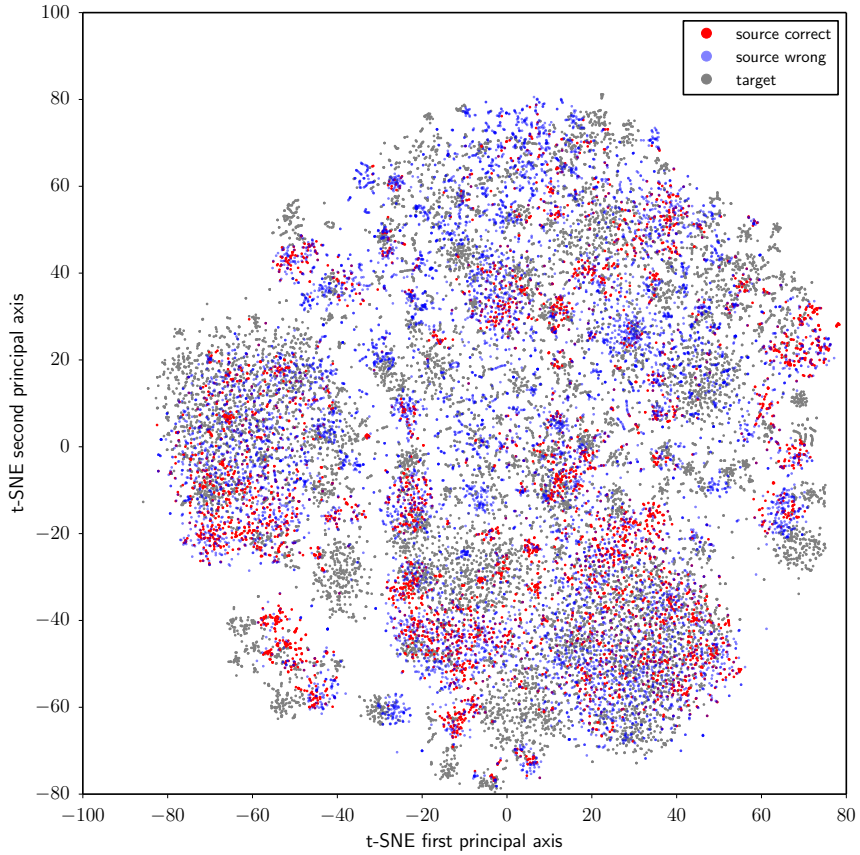
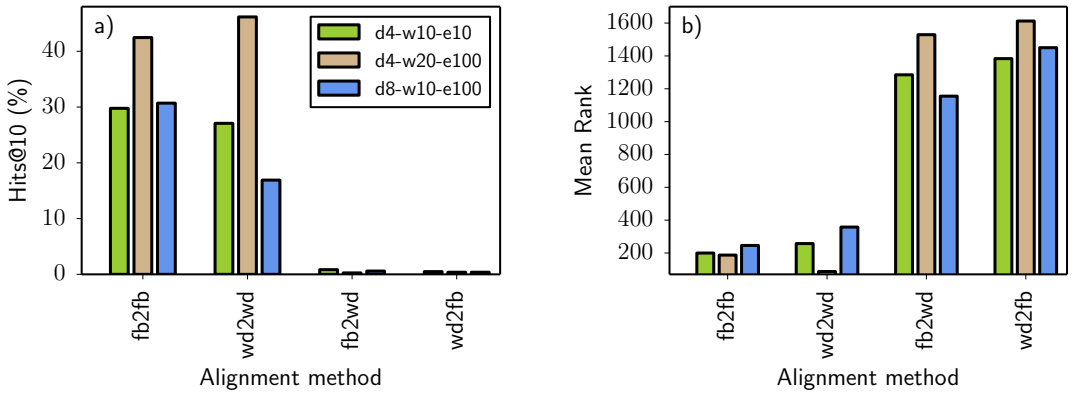


Figure 5.9: Alignment of two full wd15k-237 TransE embedding spaces with mapping computed on 8000 least frequent entities. Source correct refers to the correctly aligned source entities, source wrong refers to the wrongly aligned source entities, target refers to the entities from original target space.

bedding methods (Figure 5.5). The alignment for two different KGs remains on par with the random baseline.

Inspecting the KGs, we found that relations differ between them: they have different names, there are generalizations and specifications of each other in both directions. In light of this knowledge, we explain our findings. The difference between relations reflects in the different connectivity of nodes. The same entities in both KGs have different incoming and out-going edges. Therefore, the random walks generated by RDF2vec create very different embedding spaces what leads to random-like performance for fb15k-237 to wd15k-237 alignment.

The removal of some relations from fb15k-237 and alignment to full fb15k-237 leads to lower hits@10 than full fb15k-237 to full fb15k-237 alignment. We



(a) average hits@10

(b) average MR

Figure 5.10: Alignment between two instance of the same RDF2vec setting. The hyper-parameters are: d – max depth of random walk, w – max number of random walks, e – number of epochs. The x axis shows which two KGs are aligned.

are curious about the role relations play in the alignment of different embedding spaces. After seeing the difference between fb15k-237 and wd15k-237 relations we anticipated that their role is very important in shaping the embedding space.

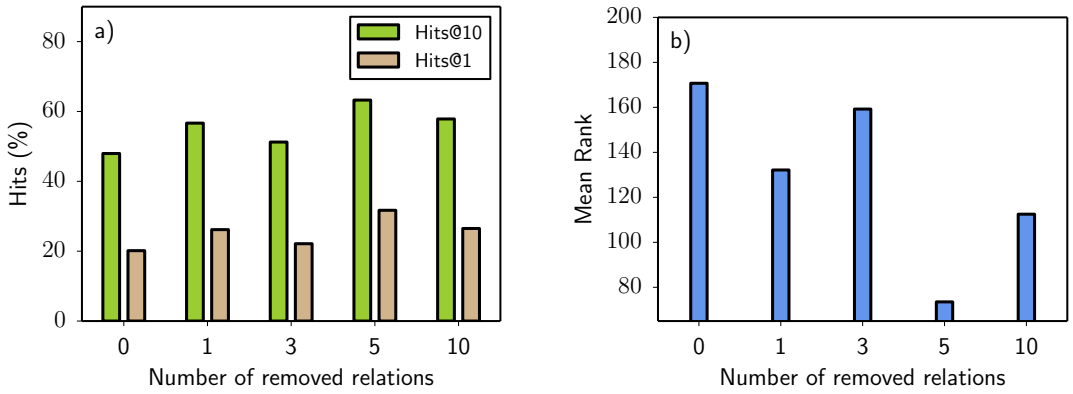
Therefore, we experiment with the alignment of two TransE runs of fb15k-237. One run is trained on a full fb15k-237 training set. The other run has the same number of entities, but n relations are chosen and triples that contain these relations are removed from the triple set (training, validation, and test set). The removed relations are chosen from the middle of the relation's degree distribution with an average number of triples around 300. The results in Figure 5.11 are the state of the MUSE-u after the adversarial training.

The results for a different number of removed relations do not differ that much from each other. However, the performance with five and ten removed relations is surprisingly high. The number of relations removed might be too small to affect the alignment in terms of falling into random-like performance. Also, the structure of the other relations is still the same because we are aligning the same KG. This might be the reason why are the results still similar to the baseline or even better (full fb15k-237 to full fb15k-237 alignment).

5.1.4 Discussion

The answer to the first research question, if LT performs comparably to the word alignment methods, is yes for supervised methods, but not entirely for unsupervised methods.

Word alignment across languages is probably simpler because, for example, European languages have similar behaviour, similar dependencies, or similar relations between



(a) Average hits@10 and hits@1

(b) Average MR

Figure 5.11: MUSE-u alignment of fb15k-237 pre-trained TransE to fb15k-237 TransE without one, three, five, and ten relations. The removed relations were chosen from the middle of relation's degree distribution by random choice. The reported results are average of both evaluation directions.

words. Moreover, relations between words are not defined and there is no strict number of them so the embedding model can model very different aspects of language. Then, the distributions of word embedding spaces are very similar. After finding correct anchors the alignment is some "simple" orthogonal linear transformation.

Whereas for KGs the two knowledge graphs could be very different in terms of relations, contained entities, knowledge about entities, or in-going vs. out-going relations. The relations are strictly defined and the model has more restrictions on modeling some other dependencies between entities and relations than word embedding models. This results in very different embedding spaces of two KGs which could be aligned if some anchor points are known but it is very difficult to find these anchors without supervision.

The majority of word alignment methods use an orthogonal transformation matrix. This results in a more balanced performance in the price of lower hits@10. The usefulness of balanced transformation depends on the use case of the alignment. If we want to be able to find the correct nearest neighbour from both directions, then the orthogonalization helps. If we are interested only in the source to target alignment then the simple LT is a better choice. The summary of hypotheses and their outcome is in the Table 5.1.

Hypothesis	Outcome
The orthogonalization of transformation matrix cause an increase in the alignment performance.	✗
An iterative orthogonal Procrustes of MUSE-s improves results.	✗
The orthogonal alignment methods have similar performance for both evaluation directions, the LT does not.	✓
The performance increases with the number of anchor points	✓
When applied on KGs, unsupervised methods for text alignment have a performance at least on par with OP.	✗
Unsupervised alignment of two pre-aligned embedding spaces has a performance on par with a supervised method.	✗
An alignment trained with the entities that occur most frequently in KG triples have higher performance than the alignment trained on the entities with the rarest occurrence.	✗
The RDF2vec embedding model performs better than other KG embedding models when using MUSE-u.	✗
The removal of some relations from fb15k-237 and alignment to full fb15k-237 leads to lower hits@10 than full fb15k-237 to full fb15k-237 alignment.	✓

Table 5.1: The summary of hypotheses behind the RQ1 and their outcome (accepted, rejected).

5.2 RQ2: Alignment of different KG embedding models

The second research question asks if the alignment between two different KG embedding models yield worse results than alignment between two instances of the same embedding model. Our motivation is that training an embedding model for large KG requires a lot of resources and time. We test if it is needed to obtain results comparable with the same model alignment.

In this section we compare the performance of TransE, RotatE, and DistMult embedding models in the alignment setting. We also compare different alignment methods and their performance when aligning two different embedding models. In the rest of the section, we experiment with the correlation between the alignment performance of two embedding models and their internal link prediction score.

Aligning embedding models trained by different embedding methods for the same KG yield lower hits@10 than aligning embedding models trained by the same embedding method. In the real world, we want to align two separately trained embedding spaces of two KGs. These embedding spaces are not necessarily trained by the same embedding model. For large KGs it is very pricey and time-consuming to retrain the embedding model, e.g. RotatE for our small fb15k-237 dataset took six days to be fully trained. Therefore, we want to know if the decrease in the performance makes it infeasible to align different KG embedding models or not.

We perform a series of experiments with fb15k-237 embedded by the pre-trained RotatE embedding model as source embedding space. As target space, we use fb15k-237 embedded by four different embedding models: RotatE, DistMult, TransE256, and TransE0. The training procedure of the embedding models is described in Section 4.2. Quick recapitulation: RotatE and DistMult are trained with LibKGE's best configuration. TransE256 is trained with 256 dimensions instead of 128 reported as the best hyperparameter. TransE0 is the pre-trained 128-dimensional TransE model appended with close to zero values to get 256 dimensions to match the dimensions of the other models. We want to test the performance of such an embedding to find if it can be a cheap way to change the embedding dimensionality without loss of the performance.

In the experiments, we align the fb15k-237 RotatE model to other embedding models (RotatE, DistMult, TransE256, TransE0). We experimented with all of the alignment methods but report only OP performance because it yields the best balanced results. The RotatE to RotatE alignment is considered the upper bound. We measured hits@10 and MR. The results show the average between evaluation directions.

Figure 5.12 shows that the alignment of the embedding models created by the same embedding method has the highest hits@10 score. The second highest score has RotatE to DistMult alignment. The alignment between models that have the same number of dimensions originally, as the best hyperparameter setting, performs better than alignment to a model in which a number of dimensions are artificially changed (like TransE0). The mean rank reflects the same behaviour. However, the difference between the target embedding models is much greater, ranging from an MR of 11 for RotatE to an MR of

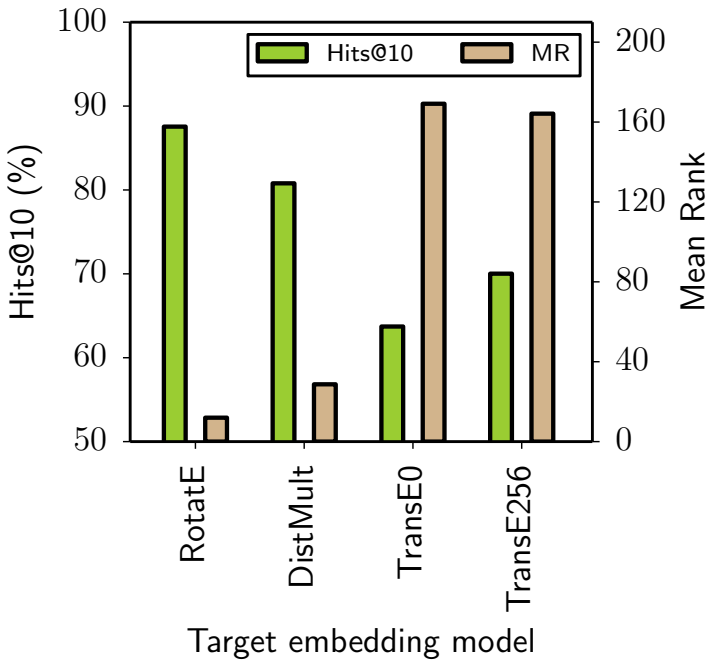


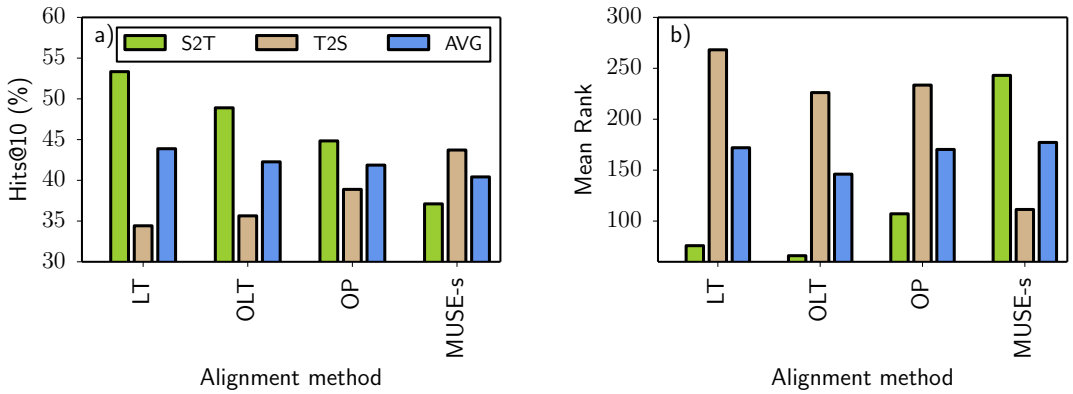
Figure 5.12: Alignment of fb15k-237 pre-trained RotatE to fb15k-237 embedded by all different models. Report average hits@10 and MR by OP 50% training set.

110 for TransE256. TransE0 has the worst hits@10 but the mean rank is a bit better than for TransE256.

Our experiment confirms the hypothesis. Aligning KGs embedded by the same embedding model yields the best results. However, if the dimensionality of the target embedding model (best hyperparameter setting) is the same as for the source embedding model the alignment might be still in the same order of magnitude. This holds for an alignment of two very similar KGs.

Orthogonal alignment methods have similar performance for both evaluation directions. LT perform the best in the setting where the same embedding model is applied on different KGs when evaluated from transformed source to target (see Section 5.1.1). The performance of orthogonal methods is more balanced in both evaluation directions. Therefore, we assume the same behaviour also for the setting where every KG is embedded by a different embedding model. The different embedding models capture different properties of KGs. We want to investigate if e.g. wd15k-237 embedded by DistMult is more similar to fb15k-237 embedded by TransE than to fb15k-237 embedded by DistMult.

In the following, we describe the experiment with the alignment of wd15k-237 embedded by DistMult to fb15k-237 embedded by TransE with 256 dimensions. We retrain the



(a) Hits@10 results

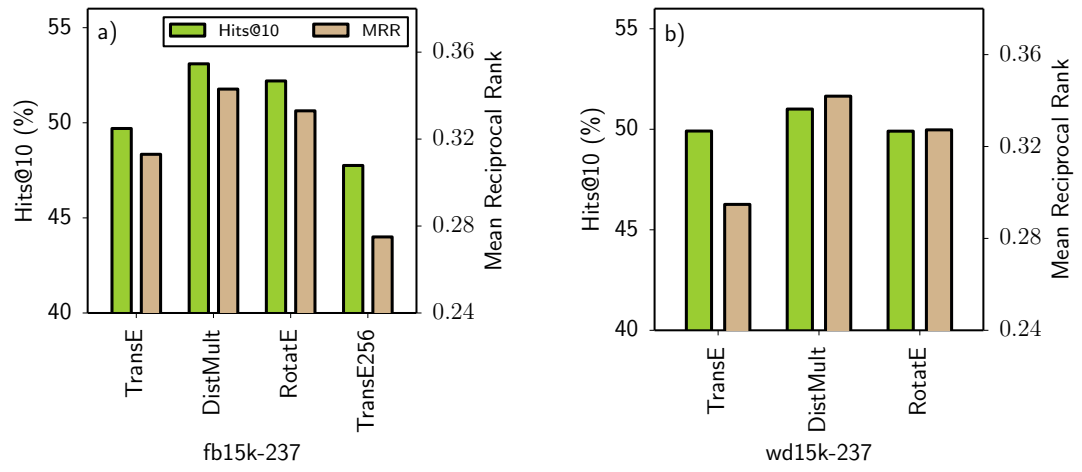
(b) Mean Rank results

Figure 5.13: Comparison of LT, OLT, OP, and MUSE-s. The aligned datasets are wd15k-237 embedded by DistMult to fb15k-237 embedded by TransE256. The S2T stands for the evaluation direction from transformed source to target, T2S stands for the evaluation from target to transformed source, AVG is the average of these two directions. The unsupervised methods for this setting yield random results.

fb15k-237 TransE model to have 256 dimensions because the best hyperparameter setting for fb15k-237 DistMult has 256 dimensions. We rather increase the dimensionality of TransE than decrease the dimensionality of DistMult. More dimensions can capture more properties but fewer dimensions capture fewer properties which might negatively affect the alignment.

Figure 5.13 shows that the average performance of both evaluation directions is similar for all of the alignment methods. LT is the least balanced alignment method. It performs the best in the evaluation direction transformed from source to target and the worst the other way around. In this scenario, the most balanced method is the OP. MUSE-s has a higher hits@10 and lower MR when evaluated from target to transformed source. The unsupervised models are not included in the plot because their performance is below one percent hits@10.

The alignment of different KGs embedded by different embedding models is the most difficult alignment scenario. This is also visible on the results in Figure 5.13. The orthogonal methods are again more balanced than LT but less balanced than in alignment of the same embedding models in Figure 5.2. The LT has the best performance in transformed source to target evaluation direction. The experiments show that the orthogonalization of the transformation matrix during the training helps to balance the results, however, at the cost of lowering hits@10.



(a) fb15k-237 models

(b) wd15k-237 models

Figure 5.14: Difference between models used in the experiments in link prediction hits@10 and MRR.

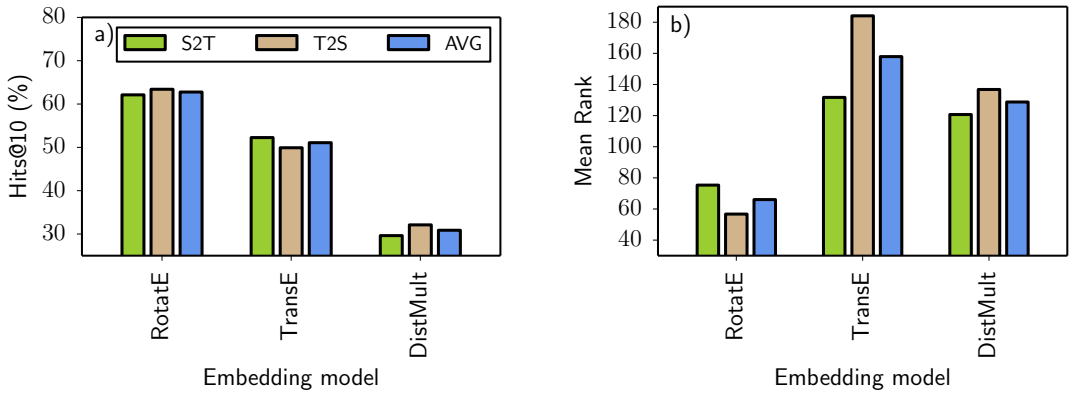
The higher the link prediction performance of the respective KG embedding models, the higher their alignment performance. We have two embedding models that are accurate in the link prediction task. Meaning that they capture the features of a KG properly. In contrast, we have two models which have lower hits@10 in the link prediction task. We expect the alignment hits@10 to be better for the two models with higher hits@10 in the link prediction task.

To create a baseline for further experiments with different embedding models, we conduct experiments where we compare the alignment of fb15k-237 to wd15k-237 embedded by the same embedding model. We conduct three experiments in which the datasets are embedded by TransE, RotatE, and DistMult. We also measure the hits@10 and MRR of the link prediction task of respective embedding models.

In Figure 5.14 we compare the link prediction hits@10 and MRR of embedding models used in these experiments. Figure 5.14a shows that the highest hits@10 for fb15k-237 embedding model is obtained by the DistMult model, the second is the RotatE model, and the third is the TransE model.

For the wd15k-237 models in Figure 5.14b is the situation a little bit different. The best model in terms of MRR is DistMult, the second is RotatE and the third place belongs to TransE. In terms of hits@10, the best model is again DistMult. The difference between RotatE and TransE is about one percentage point hits@10.

In Figure 5.15 we compare the performance of the alignment between fb15k-237 and wd15k-237 embedded by the same embedding model. The alignment method is OP with 50% training set. The best performing alignment is between RotatE embedding models, the second-best is between TransE embedding models and the worst is between DistMult



(a) fb2wd hits@10

(b) fb2wd MR

Figure 5.15: Comparison of alignment from fb15k-237 to wd15k-237 embedded by the same embedding method. The figure shows average hits@10 of OP 50% training set.

models.

The intrinsic link prediction task is not correlated with the alignment performance of the embedding models, implying that other factors affect the alignment performance. In link prediction hits@10 varies from 47% to 53%. The difference between aligned fb15k-237 RotatE and wd15k-237 RotatE models is three percentage points for the link prediction task. For DistMult models, there is no difference (both 49%), for TransE models there is a difference of two percentage points. The models are on par with each other in means of link prediction hits@10.

However, the difference in alignment performance is much greater. We argue that the link prediction score reflects how well is the KG represented by the embedding model for intrinsic tasks. The alignment shows if we can compare the different embedding models to each other. It might be that KG embedding models have enough degrees of freedom in the embedding space so that they can represent the same KG differently.

Aligning an embedding model with high hits@10 in link prediction to a model with low hits@10 in link prediction yields worse results than an alignment the other way around. The question is if the results of aligning two embedding models depend on the choice of source and target model and dataset. We expect the link prediction to be a good intrinsic task to evaluate the ability of an embedding model to reflect important properties of KG. We expect the alignment between two models with high link prediction results to be better than alignment between a model with high link prediction results and a model with low link prediction results.

We experiment with alignment between TransE256 on fb15k-237 (low link prediction hits@10 score) and DistMult on wd15k-237 (high link prediction hits@10 score).

Figures 5.16a and 5.16b show the results of an alignment from fb15k-237 to wd15k-237. Figures 5.16c and 5.16c show the results of the alignment from wd15k-237 to fb15k-237. We computed the alignment between the given two models with all of the alignment methods.

We observe that the choice of the source embedding model affects the results. The method with the most consistent alignment performance is OP. LT shows the highest difference between the two settings. The fb15k-237 TransE256 to wd15k-237 DistMult case has a hits@10 score of about 4% when evaluated from target to transformed source. However, for alignment wd15k-237 to fb15k-237, the score is about 34%.

The hypothesis is rejected. The high to low case yields better results than low to high. This shows that results depend on the choice of source and target embedding space, but the link prediction task is not correlated with the alignment performance. The properties of embedding spaces that affect the alignment should be studied further.

5.2.1 Discussion

Finding alignment between two KGs embedded by two different embedding models is the most complicated scenario. The different embedding spaces capture different properties of KGs which affects the performance of an alignment.

Our experiments show that the supervised algorithms have hits@10 still about 40%. On the contrary, the unsupervised algorithms are not able to find the initial alignment, so, the final results are below one percent.

The embedding spaces of the two KGs are very different. Even if the two KGs contain the same entities, the relations, the in-coming, and out-going edges might be very unalike what creates two embedding spaces that cannot be trivially aligned. The usage of two different embedding models just adds to the complexity of this task.

Further, our experiments show that for the supervised models the performance relies on choosing the correct source and target embedding model. The reason why some embedding models are more suitable to embed source spaces and some are more suitable to embed target spaces during the training is still unclear and requires further experimentation.

We also compare the alignment performance between two runs of RotatE, DistMult, and TransE respectively. We conduct these experiments to obtain the upper bound on the performance of alignment between different embedding models. The RotatE model works the best when aligned with itself yielding the hits@10 about 60%. TransE works comparably well with hits@10 about 50%. On the other hand, the DistMult model when aligned to itself has performance only about 30% hits@10.

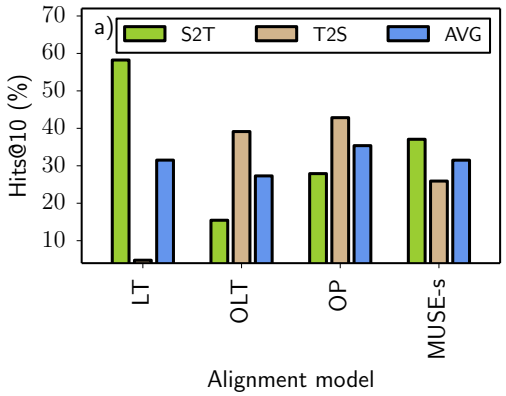
It is interesting that when we align DistMult to TransE the performance is still about 50% average hits@10, but the alignment of TransE to DistMult yields average hits@10 only about 30%. This suggests that the target model is more important than the source model because it defines the space in which we are looking for the alignment.

Our final observation is that the link prediction metric is not suitable for the prediction of the alignment performance of the models. To find such a metric remains the task for

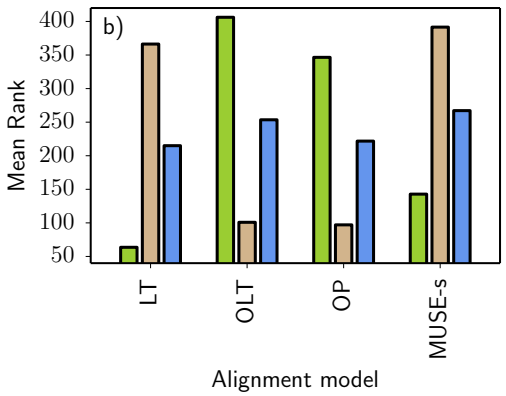
Hypothesis	Outcome
Aligning embedding models trained by different embedding methods for the same KG yield lower hits@10 than aligning embedding models trained by the same embedding method.	✓
Orthogonal alignment methods have similar performance for both evaluation directions.	✓
The higher the link prediction performance of the respective KG embedding models, the higher their alignment performance.	✗
Aligning an embedding model with high hits@10 in link prediction to a model with low hits@10 in link prediction yields worse results than an alignment the other way around.	✗

Table 5.2: The summary of hypotheses for RQ2 and their outcome.

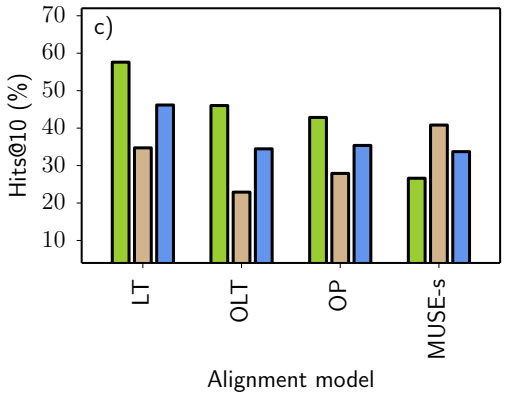
future research. Table 5.2 summarizes the hypotheses for RQ2, if they are rejected or accepted.



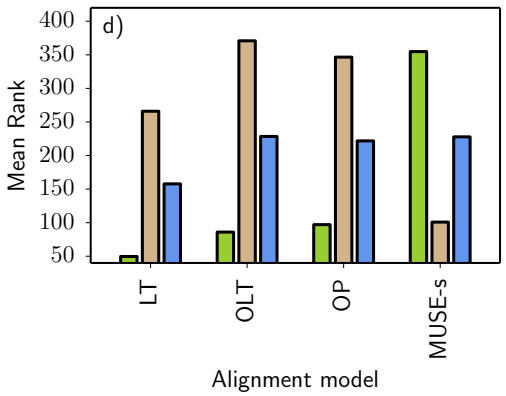
(a) fb15k-237 TransE256 to wd15k-237 DistMult



(b) fb15k-237 TransE256 to wd15k-237 DistMult



(c) wd15k-237 DistMult to fb15k-237 TransE256



(d) wd15k-237 DistMult to fb15k-237 TransE256

Figure 5.16: Comparison of alignment from fb15k-237 TransE256 to wd15k-237 DistMult and the other way round. On the x axis are all supervised alignment methods. The figures show the different directions of evaluation together with the average of these directions.

6

Future Work

This work presents a broad study of KG alignment using word alignment methods and shows results for alignment between different embedding models. The more experiments we conducted the more interesting questions popped up.

The interesting and understudied area is the behaviour of KG embedding spaces. Our hypothesis is that unlike the word embedding spaces of two languages the embedding spaces of two KGs do not have a similar distribution of embeddings. The correct assumptions about the embedding spaces could help to improve the alignment algorithms.

Another research area might be finding the correct embedding model to embed a concrete KG for the alignment task. Some KG embedding models might be easier to align, enhancing the whole performance of an alignment. As we observed, the choice of the source and target embedding model makes a difference of about 20 percentage points.

For now, we have no means how to measure if a concrete KG embedding model is suitable for the alignment task or not. We observe that the link prediction task is not correlated with the alignment performance. It would be nice to have some indication if a model is suitable for an alignment or not. Therefore, we propose to investigate if there are some correlations between intrinsic and extrinsic tasks for KGs which can point this out.

Unsupervised alignment remains an open issue. In MUSE-u, the reason why the adversarial part of the algorithm does not work for some setups is not clear. We incline to the idea that two KGs are transformed into two not trivially matchable spaces which pose too hard a problem for the unsupervised initialization algorithms.

If we want to stick to the adversarial training, then we could try to enhance the results by adjusting the data. We would further experiment with adding the relation embeddings. For example, the relation embeddings might be added into the training set to have more training points. We can also try to align entities and relations separately, then combine the transformation matrix in some sophisticated way to obtain more accurate alignment.

On the other hand, we can try to replace the adversarial training and find some unsupervised initialization that is able to find necessary anchor points for iterative Procrustes alignment. But for this case, we would need to find the reason why the adversarial training does not work.

We can also investigate the importance of relations in shaping the KG embedding space. For example, we could create new KGs by choosing relations that can be unambiguously aligned between fb15k-237 and wd15k-237. The entity set would contain the entities which are connected by these relations. We would then experiment with an alignment of these new KGs to see if the results of an alignment can be better compared to KGs that contain relations that can not be aligned.

7

Conclusions

In this thesis we answer two research questions:

1. *Do the word alignment methods used on KGs have results in the same order of magnitude as typical KG alignment methods?*

2. *Do the alignment between different KG embedding models yield results in the same order of magnitude with alignment between the same embedding models?*

The answer to the first question is yes for the supervised alignment methods and not entirely for the unsupervised alignment methods. The supervised word alignment methods yield a performance in the same order of magnitude as LT which we consider the typical KG alignment method according to our literature review.

The unsupervised alignment methods are on par with the supervised alignment methods for a specific case: alignment between the same KG embedded by two runs of the same embedding model. This is the simplest alignment scenario which resembles the word alignment scenario the most. The embedding spaces have the same distribution because they embed the same KG. The reason why the initial mapping cannot be found for other scenarios (alignment between different KGs or different embedding models) remains a question for future research.

Our hypothesis is that two different KGs are more different than two different languages in terms of relations and dependencies between entities or words. This causes the KG embedding space we want to align to be not trivially matchable.

The answer to the second research question is no. The performance of the best alignment of two different embedding models is about 20 percentage points worse compared to the best alignment between the same embedding models. The scenario in which we want to align two different KGs each embedded by a different embedding model poses a hard problem even for supervised alignment. Therefore, the results are not on par with the alignment between the same embedding models. The unsupervised algorithms perform randomly (hits@10 below 1%) in this scenario.

In the following, we summarize the key findings of this thesis:

1. Word alignment methods (orthogonal methods) yields the same average results as LT.

2. The orthogonalization balances the KG alignment in terms of direction of evaluation.
3. Embedding spaces of two different KGs are not trivially matchable by the unsupervised alignment algorithms.
4. The adversarial part of the MUSE-u cannot find suitable anchor points between two different KGs.
5. The alignment between two different embedding models works in a supervised way, but the results are lower than with the same embed models.
6. The choice of source and target embedding spaces in the alignment of two different KG embedding models changes the results.
7. The link prediction performance of KG embedding models is not correlated with the alignment results.

References

[Ali et al., 2020] Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., and Lehmann, J. (2020). Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *arXiv preprint arXiv:2006.13365*.

[Alvarez-Melis and Jaakkola, 2018] Alvarez-Melis, D. and Jaakkola, T. S. (2018). Gromov-wasserstein alignment of word embedding spaces. *arXiv preprint arXiv:1809.00013*.

[Ammar et al., 2016] Ammar, W., Mulcaire, G., Tsvetkov, Y., Lample, G., Dyer, C., and Smith, N. A. (2016). Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

[Artetxe et al., 2018] Artetxe, M., Labaka, G., and Agirre, E. (2018). A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. *arXiv preprint arXiv:1805.06297*.

[Beyer et al., 2020] Beyer, A., Kauermann, G., and Schütze, H. (2020). Embedding space correlation as a measure of domain similarity. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2431–2439.

[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 *Neural Information Processing Systems (NIPS)*, pages 1–9.

[Chen et al., 2018] Chen, J., Tao, Y., and Lin, H. (2018). Visual exploration and comparison of word embeddings. *Journal of Visual Languages & Computing*, 48:178–186.

[Conneau et al., 2017] Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

[Cuturi, 2013] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

[Fang et al., 2016] Fang, W., Zhang, J., Wang, D., Chen, Z., and Li, M. (2016). Entity disambiguation by knowledge and text jointly embedding. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 260–269.

[Grave et al., 2019] Grave, E., Joulin, A., and Berthet, Q. (2019). Unsupervised alignment of embeddings with wasserstein procrustes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1880–1890. PMLR.

[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.

[Heist et al., 2020] Heist, N., Hertling, S., Ringler, D., and Paulheim, H. (2020). Knowledge graphs on the web-an overview.

[Li et al., 2015] Li, Y., Yosinski, J., Clune, J., Lipson, H., and Hopcroft, J. E. (2015). Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pages 196–212.

[Liu et al., 2020] Liu, F., Chen, M., Roth, D., and Collier, N. (2020). Visual pivoting for (unsupervised) entity alignment. *arXiv preprint arXiv:2009.13603*.

[Liu et al., 2018] Liu, W., Liu, J., Wu, M., Abbas, S., Hu, W., Wei, B., and Zheng, Q. (2018). Representation learning over multiple knowledge graphs for knowledge graphs alignment. *Neurocomputing*, 320:12–24.

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

[Rastogi et al., 2015] Rastogi, P., Van Durme, B., and Arora, R. (2015). Multiview lsa: Representation learning via generalized cca. In *Proceedings of the 2015 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pages 556–566.

[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.

[Ruder et al., 2019] Ruder, S., Vulić, I., and Søgaard, A. (2019). A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

[Ruffinelli et al., 2019] Ruffinelli, D., Broscheit, S., and Gemulla, R. (2019). You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

[Schnabel et al., 2015] Schnabel, T., Labutov, I., Mimno, D., and Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 298–307.

[Schönemann, 1966] Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

[Sun et al., 2019] Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

[Sun et al., 2018] Sun, Z., Hu, W., Zhang, Q., and Qu, Y. (2018). Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, volume 18, pages 4396–4402.

[Toutanova and Chen, 2015] Toutanova, K. and Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

[Trisedya et al., 2019] Trisedya, B. D., Qi, J., and Zhang, R. (2019). Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 297–304.

[Vandewiele et al., 2020] Vandewiele, G., Steenwinckel, B., Agozzino, T., Weyns, M., Bonte, P., Ongenae, F., and Turck, F. D. (2020). pyRDF2Vec: Python Implementation and Extension of RDF2Vec. IDLab.

[Wang et al., 2018a] Wang, L., Hu, L., Gu, J., Wu, Y., Hu, Z., He, K., and Hopcroft, J. (2018a). Towards understanding learning representations: To what extent do different neural networks learn the same representation. *arXiv preprint arXiv:1810.11750*.

[Wang et al., 2017] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

[Wang et al., 2018b] Wang, Z., Lv, Q., Lan, X., and Zhang, Y. (2018b). Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

[Wang et al., 2014] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601.

[Wu et al., 2018] Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. (2018). Starspace: Embed all the things! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Xing et al., 2015] Xing, C., Wang, D., Liu, C., and Lin, Y. (2015). Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

[Xu et al., 2019] Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., and Yu, D. (2019). Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605*.

[Yang et al., 2014] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

[Zeng et al., 2021] Zeng, W., Zhao, X., Tang, J., Li, X., Luo, M., and Zheng, Q. (2021). Towards entity alignment in the open world: An unsupervised approach. *arXiv preprint arXiv:2101.10535*.

[Zhang et al., 2017] Zhang, M., Liu, Y., Luan, H., and Sun, M. (2017). Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1934–1945.

[Zhu et al., 2017] Zhu, H., Xie, R., Liu, Z., and Sun, M. (2017). Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, volume 17, pages 4258–4264.

A

Appendix

The results of the experiments conducted during this thesis together with the alignment codes and descriptions of the evaluation pipelines are available at *https://gitlab.ifi.uzh.ch/ddis/Students/Theses/2020-embedding-space-alignment*.

List of Figures

List of Tables