LMU LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME
UND DATA MINING

Master Thesis
in Data Science

Relation Disambiguation in Knowledge Graphs

IVICA OBADIC

Supervisor:	Prof. Dr. Matthias Schubert
Advisor:	Christian Frey
Submission date:	15.11.2019

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

Munich, 15.11.2019

.....
IVICA OBADIC

Abstract

Knowledge Graphs are widely used in many different domains such as search engines, social networks and recommendation systems. They are usually created by aggregating data from different domains and data sources which introduces ambiguous entities and ambiguous relations in the Knowledge Graph. Ambiguous entities are resolved with the existing entity resolution approaches. In this thesis, we introduce a novel framework capable of relation disambiguation in Knowledge Graphs. The core component of the presented framework is the proposed Relation Disambiguation algorithm which is able to identify and disambiguate ambiguous relations in a Knowledge Graph. The algorithm splits ambiguous relations into multiple unambiguous relations based on clustering assignments obtained in the latent space of relation-specific entity embeddings. These embeddings are computed on the basis of RESCAL's tensor factorization results for the input Knowledge Graph. The relation disambiguation framework is evaluated on three different benchmark Knowledge Graph datasets and the obtained results show that the Relation Disambiguation algorithm enriches the semantics of a Knowledge Graph which at the same time leads to a RESCAL factorization for a Knowledge Graph with disambiguated relations that achieves improvements in the link prediction performance.

Contents

Chapter 1

Introduction

In today's digital era, data is one of the most important assets that transforms and moves the world forward. Many organizations and business govern their decisions based on the extracted insights from the available data. In order to extract knowledge and uncover the semantics behind the vast amounts of heterogeneous data that are being generated every day, the data is structured into a set of triples which specify the relationships between the existing concepts in the data. The set of triples provides a domain-specific knowledge base which is organized into a **Knowledge Graph**. Knowledge Graphs model the information present in the data in terms of entities which represent real-world concepts and relations which represent different types of relationships that exist between these concepts. The entities and the relations found in the data are organized in a graph structure which encodes the semantics of the domain that has been modeled.

Knowledge Graphs are widely used in many different domains such as search engines, chatbots, social networks, recommendation systems and biomedicine. One of the most popular examples for a commercial Knowledge Graph is the Google Knowledge Graph which is used in search results enhancement and in the Google Assistant. Google Knowledge Graph covers different categories of entities such as artists, paintings, movies, books, countries and cities and models 70 billion triples that exist between the entities. The ability to encode domain semantics in an intuitive way that can be efficiently queried and the potential of discovering valuable insights from the Knowledge Graphs inspired many companies to transform their data in knowledge base which is modeled by a Knowledge Graph. Such example is the Linkedin Knowledge Graph [1] which is used to extract valuable consumer analytics information. The entities in this Knowledge Graph are the Linkedin users, skills, jobs and companies. The relations specify different relationships such as the current company of a

user or users professional skills.

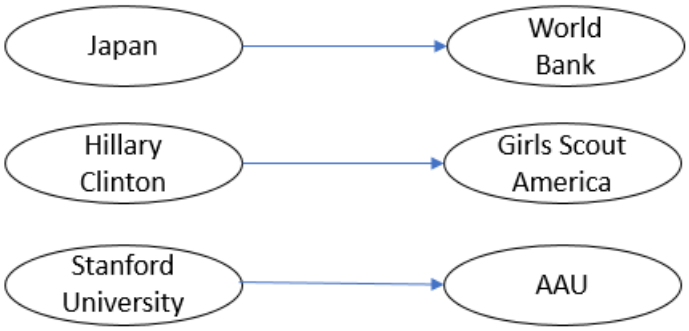

Figure 1.1: Subgraph for the relation *organization_membership* in the Freebase Knowledge Graph

Knowledge Graphs are usually created from individual user contributions and by merging and aggregating data from different domains and data sources. For example, the Freebase Knowledge Graph, which contains general facts about the world, was created by aggregating entities and relations from different online encyclopedias and databases such as Wikipedia, NNDB[1], MusicBrainz, as well as from user-submitted wiki contributions. As a consequence of the creation process, Knowledge Graphs contain ambiguous entities and relations. Ambiguous entities are resolved by the existing Entity Resolution approaches. However, in the focus of this thesis are the ambiguous relations. Figure 1.1 shows a subgraph for the relation *organization_membership* from the Freebase Knowledge Graph. This subgraph illustrates that the relation *organization_membership* models the facts that Japan is a member of the World Bank, Hillary Clinton is a member of the Girls Scout America and that Stanford is a member of the American Association of Universities. These facts indicate that there are multiple semantically different kinds of memberships modeled with only one relation. Therefore, the relation *organization_membership* is ambiguous since organization membership in this context can represent different kinds of memberships.

In this master thesis, we created a novel framework that performs relation disambiguation in Knowledge Graphs. The proposed framework is based on our Relation Disambiguation algorithm which discovers and disambiguates

[1]Notable Names Database (NNDB) is an online database of biographical details of over 40,000 people of note

ambiguous relations into multiple relations with clear semantics. The steps of detecting ambiguous relations and their further disambiguation are based on clustering assignments obtained in the latent space of relation-specific entity embeddings. These embeddings are computed on the basis of RESCAL's tensor factorization results for the input Knowledge Graph.

The RESCAL method models the interactions between the latent features for a relation with an interaction matrix. The interaction matrices for ambiguous relations need to capture interactions on multiple semantic levels. By splitting ambiguous relations into multiple unambiguous relations, each interaction matrix captures only interactions between entities that share the same semantic which is easier to be modeled by the bilinear RESCAL method.
Therefore, within the scope of our framework, we evaluate the effect of the Relation Disambiguation algorithm on the link prediction performance of the RESCAL method by performing experiments on the FB15k, FB15k-237 and WN18 benchmark Knowledge Graph datasets. Additionally, we evaluate the detected ambiguous relations from a semantical point of view. The obtained results show that the Relation Disambiguation algorithm is capable of detecting ambiguous relations. The semantic evaluation has shown that relation disambiguation enhances the semantics of the Knowledge Graph. The improvements in the link prediction performance achieved on all datasets used in the experiments prove that the RESCAL method benefits and is improved from enrichment of the Knowledge Graph semantics.

Chapter 2

Related Work

Relational machine learning is a field which studies methods for statistical analysis of graph data. One of the main tasks in this field are the tasks of link prediction, entity resolution and relation disambiguation in Knowledge Graphs [11]. A common first step towards solving link prediction and entity resolution tasks is to encode the entities and relations in the Knowledge Graph into a low dimensional latent-feature representations. There are many different approaches for learning low dimensional representations from Knowledge Graphs which can be grouped into translation methods like TransE [3] and TransH [22], factorization techniques like RESCAL [13] and HolE [12] and neural network approaches like NTN [18] and MLP [5]. There many examples where embeddings produced by these methods are applied in solving the link prediction and the entity resolution tasks. Entity resolution is most often performed only based on the similarity scores between the entities embeddings. However, none of the embeddings produced with any of the above methods were used for relation disambiguation. Currently, there is no related work in the field of relation disambiguation which is applied for Knowledge Graphs.

On the other hand, relation disambiguation is a problem that is widely explored in NLP as part of the relation extraction topic. It is used to disambiguate the different relations extracted between named entities in text. For example, [4] presents an unsupervised approach for relation disambiguation in a text based on Spectral clustering of context vectors created for pairs of entities.

Chapter 3

Preliminaries

3.1 Knowledge Graphs

Knowledge graphs model information represented in triples of the form (*subject*, *relation*, *object*) in a graph structure. The nodes in the graph represent real world entities - the subjects and objects in the triples. The relations in the graph, also called labeled edges, represent the different types of relationships that exist between these entities.

Mathematically, a Knowledge Graph that models N entities and M relations can represented as a 3-dimensional tensor KG of shape (N, N, M). If $M = 1$, then KG corresponds to a single-relational graph's adjacency matrix. Therefore, a Knowledge Graph KG is defined if and only if $N > 1$ & $M > 1$. We define KG_r to be the r-th frontal slice of the tensor KG. KG_r can be seen as an adjacency matrix or a directed unweighted graph for the relation r which specifies the edges between the subjects and objects that are connected by relation r. A tensor entry $KG_{ijr} = 1$ models the triple $(i,\ r,\ j)$ and represents the existence of an edge from the i-th entity to the $j - th$ entity in the adjacency matrix for relation r. We define the subjects in relation r with the following equation:

$$s_r = \{i \mid KG_{ijr} = 1,\ \forall i,\ \forall j \in 1, ..., N\}$$

Similarly, the objects for relation r are defined with the following equation:

$$o_r = \{j \mid KG_{ijr} = 1,\ \forall i,\ \forall j \in 1, ..., N\}$$

While the existing triples are encoded with edges in the Knowledge Graph, there are two different interpretations for the non-existing triples which are

represented with zeros in the Knowledge Graph tensor. The first interpretation is based on the *closed world assumption* which states that non-existing triple indicates false relationship. The second interpretation is based on the *open world assumption* which states that a non-existing triple is interpreted as unknown, meaning that the triple may or may not exist. This interpretation is more justified than the first one since, in general, Knowledge Graphs are known to be very incomplete [11].

3.1.1 Link Prediction

Link prediction, also known as *knowledge graph completion*, predicts the existence of edges in the Knowledge Graph. It is a very important prediction task because it adds missing triples in the Knowledge Graphs which expand the knowledge base of triples.

3.1.1.1 Metrics

The link prediction metrics are based on the edges ranks [2]. All edges for which we try to predict their existence are considered as positive, existing triples. The rank of each positive edge is determined based on the rank of its score against the scores of a set of negative edges generated for the given positive edge. Common metrics for link prediciton results are:

1. MRR: The average of the reciprocal ranks of all positive edges (higher is better, best is 1).

2. $Hits@K$: A fraction of positive edges that rank in top K among their negatives (higher is better, best is 1). Common values used for K are 1, 3 and 10.

3.2 RESCAL

The RESCAL method [13] is a tensor factorization approach which factorizes each frontal slice KG_r as:

$$KG_r \approx ER_rE^T, \ for\ r = 1, ..., M \tag{3.1}$$

In equation 3.1, E is a N x D matrix that contains the embeddings of the entities in KG. These embeddings encode the Knowledge Graph entities into a D dimensional latent-feature vector representations. R_r is an asymmetric D x D matrix which models the interactions of the latent features for the r-th relation in the Knowledge Graph. This equation implies that the entities have

a global and unique latent-feature representation, regardless of their occurrence as subjects or as objects in a relation, as they are represented both times by the matrix E. The asymmetry of the matrix R_r allows modeling the occurrence of the entities as subjects or objects for the relation r by computing different estimates for the existence of the triples $(i,\ r,\ j)$ and $(j,\ r,\ .i)$.
RESCAL can also be seen as a latent-feature model which explains triples via pairwise interactions of latent features [11]. Based on the equation 3.1, the score of a triple (i, r, j) can also be written as:

$$\hat{f}_{ijr} := e_i^T R_r e_j = \sum_{a=1}^{D} \sum_{b=1}^{D} R_{abr} E_{ia} E_{jb} \tag{3.2}$$

This equation implies that RESCAL is a *bilinear model* since the interactions between the entity embeddings are captured using multiplicative terms.

The global entity latent-feature representations in the E matrix can be seen as *semantic entity embeddings* in a sense that two representations are close in the latent-feature space if the corresponding entities are connected to similar entities via similar relations. For instance, if the triples $(i,\ r,\ j)$ and $(p,\ r,\ j)$ exist for the relation r, then in order to correctly predict the existence of these triples by the RESCAL model, the embeddings for the i-th entity and p-th entity must be similar. As a consequence, entities with many similar observed relations will have similar latent-feature representations.

The parameter that has highest influence on the complexity and therefore on the performance of the RESCAL model is the dimensionality D of the latent-feature space. A small value for D can lead to a simple model which is unable to explain the entities semantics and properly model the different interactions between the latent-features. On the other hand, a large value for D can produce a complex model which leads to overfitting in the training procedure.

3.2.1 Training Procedures

The training procedure for the RESCAL factorization aims to find estimates for the entity embeddings matrix E and the interaction matrices $R_r\ , r = 1, ..., M$. The E matrix and the matrices R_r are estimated with one of the following procedures described in this section.

3.2.1.1 RESCAL-ALS

The matrix E and the matrices R_r are computed by solving the regularized minimization problem:

$$\min_{E,R_r} f(E, R_r) + g(E, R_r) \tag{3.3}$$

where

$$f(E, R_r) = \frac{1}{2}\left(\sum_r \|KG_r - ER_rE^T\|_F^2\right) \tag{3.4}$$

and g is the following regularization term:

$$g(E, R_r) = \frac{1}{2}\lambda\left(\|E\|_F^2 + \sum_{r=1}^{M} \|R_r\|_F^2\right) \tag{3.5}$$

that is added to prevent model overfitting. The regularized minimization problem in equation 3.3 is solved with the alternating least-squares (ALS) approach adjusted for the RESCAL method, presented in [13]. This approach uses a sequence of efficient closed-form updates for the E matrix and R_r matrices. The training method presented in this section interprets the Knowledge Graph based on the closed-world assumption since the minimization problem requires predicting values close to zero for the non-existing triples in the Knowledge Graph.

3.2.1.2 Penalized Maximum Likelihood

The matrix E and the matrices R_r are computed with penalized maximum likelihood training procedure by minimizing the following loss function:

$$\min_{E,R_r} \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r=1}^{M} -\log Ber(KG_{ijr}|\sigma(\hat{f}_{ijr})) + \frac{1}{2}\lambda\left(\|E\|_F^2 + \sum_{r=1}^{M} \|R_r\|_F^2\right) \tag{3.6}$$

where $\hat{f}_{ijr}$ is the prediction of the RESCAL model for the triple (i, r, j) computed with the equation 3.2 and σ is the *sigmoid* function [11]. This loss function can be optimized with the Stochastic Gradient Descent Algorithm (SGD) [17]. This training method also interprets the Knowledge Graph based on the closed-world assumption.

3.2.1.3 Pairwise Loss

The training procedures described in sections 3.2.1.1 and 3.2.1.2 estimate the parameters of the RESCAL model by using the existing triples as well as all

of the non-existing triples in the Knowledge Graph. Since Knowledge Graphs are generally sparse, considering all of the non-existing triples can lead to scalability issues during training [11]. An alternative approach is the pairwise loss training procedure. In order to define this method, we introduce the following sets of triples of size T: D^+ is a set of all existing (positive) triples in the Knowledge Graph and D^- is a set of non-existing (negative) triples created by generating one negative triple for each of the positive triples. The pairwise loss method obtains the estimates of the RESCAL parameters by minimizing the following objective function:

$$\min_{E,R_r} \sum_{i=1}^{T} L\Big(\hat{f}(x_{i+}), \hat{f}(x_{i-})\Big) + \frac{1}{2}\lambda\left(\|E\|_F^2 + \sum_{r=1}^{M} \|R_r\|_F^2\right) \tag{3.7}$$

where (x_{i+}) and (x_{i-}) are triples from the sets D^+ and D^- and $\hat{f}_x$ is the RESCAL prediction for a triple x given in the equation 3.2. L is the following *margin-based ranking loss function*:

$$L(\hat{f}^+, \hat{f}^-) = \max(1 + \hat{f}^- - \hat{f}^+, 0) \tag{3.8}$$

where $\hat{f}^+$ and $\hat{f}^-$ are the scores for a positive and a negative triple, respectively. The main advantage of the pairwise loss training procedue is that it relaxes the closed-world assumption through the usage of the margin-based ranking loss function. This loss function does not require the negative triples to be predicted close to zero, they just need to be sufficiently smaller than the corresponding positive triples. This training method is also optimized by SGD or any of its extensions.

3.3 Spectral Clustering

Spectral clustering [10] is a clustering algorithm that aims to find a partitioning of the graph that produces the optimal *graph cut*. The Spectral clustering algorithm computes clusters by performing the following steps for a given set of p data points $x_1, ..., x_p$ and some notion of similarity s_{ij} between all pairs of data points x_i and x_j:

1. Construct a *similarity graph* from the data points.
2. Compute a *Laplacian Matrix* based on the similarity graph.
3. Perform *eigendecomposition of the Laplacian Matrix* from step 2.
4. Find $\boldsymbol{k}$ clusters in the space of the first $\boldsymbol{k}$ eigenvectors of the Laplacian matrix using for example the *k-means* algorithm.

3.3.1 Similarity Graph

The similarity graph for a set of p points represents a weighted symmetric adjacency matrix of shape (p, p). It models the local neighborhood relationships that exist between the data points [8]. There are three different types of similarity graphs:

- **k-nearest neighbor graphs**: In this type of graphs, a node i is connected to a node j if j is among the k-nearest neighbors of i. However, this definition leads to a construction of a directed similarity graph because the neighborhood relationship is not symmetric. There are two ways to transform this graph into undirected graph:
  - The directions of the edges are ignored and node i is connected to node j with an undirected edge if i is among the the k-nearest neighbors of j or j is among the k-nearest neighbors of i. This graph is usually called *k-nearest neighbor graph.*
  - A node i is connected to a node j if i among the the k-nearest neighbors of j and j is among the the k-nearest neighbors of i. This graph is called *mutual-k-nearest neighbor graph.*
- The **ε-neighborhood graph**: This similarity graph connects all nodes whose pairwise similarity is greater than ε.
- **The fully connected graph**: This similarity graph connects nodes with positive pairwise similarity.

The edges in these similarity graphs are weighted by the corresponding pairwise similarities between the nodes.

3.3.2 Eigengap heuristic

Choosing the optimal number of clusters is a general problem for all clustering algorithms. A heuristic suitable for the Spectral clustering algorithm is the *eigengap heuristic* presented in [8]. This heuristic considers the eigenvalues of the Laplacian matrix. The aim of this heuristic is to choose the number of clusters such that all eigenvalues $\lambda_1, ..., \lambda_k$ are very small, but λ_{k+1} is relatively large. The major justification for this heuristic comes from the fact that in the case that there are k connected components in the similarity graph, the eigenvalue 0 has multiplicity k and $\lambda_{k+1} > 0$ for the eigenvalues of the Laplacian matrix.

Multiple larger eigengaps can occur in the eigenvalues of the Laplacian matrix. In order to find the position of the first larger eigengap, for each of the p eigenvalues we define an **eigengap score** that is computed with the following equation:

$$eigengap\_score_n = \begin{cases} 0, & \text{if } n = 1 \\ w \sum_{i=1}^{n-1} -\lambda_i + (1-w)|\lambda_n - \lambda_{n\text{-}1}|, & n \in 2, ..., p, w \in [0, 1] \end{cases} \tag{3.9}$$

As can be seen in this equation, a weight w is introduced for the computation of the eigengap score. It represents a trade-off between the eigengap in the n-th position and the negative cumulative sum of eigenvalues in the first $n-1$ positions. This equation gives preference towards computing higher eigengap scores for the first larger eigengaps. After computing the eigengap scores, the optimal number of clusters k is found by selecting the position of the highest eigengap score with the following equation:

$$k = \arg\max_k \{eigengap\_score_k \mid k \in 1, ..., p\} \tag{3.10}$$

3.4 Hierarchical Clustering

Hierarchical clustering [9] is a clustering algorithm which builds a hierarchy of clusters with the following procedure: Initially, each data point is assigned into its own cluster. In each step of the algorithm, based on the similarity measure between the data points and on the linkage criteria, the two most similar clusters are merged into one cluster. The algorithm terminates either when the desired number of clusters is obtained or when there are no clusters whose similarity is greater than a predefined similarity threshold.

A common measure used to evaluate the clusters computed with Hierarchical clustering is the *silhouette score* [16]. It is a measure that indicates how similar an object is to its own cluster compared with other clusters. The range of the silhouette score is between -1 and +1. A value close to 1 indicates that the object is well matched to its own cluster and poorly matched to other clusters. The silhouette measure for a set of data points with corresponding clustering assignments is calculated by taking the average of the silhouette scores computed for all data points.

Chapter 4

Relation Disambiguation Framework

A core component of the proposed Relation Disambiguation Framework in this thesis is the **Knowledge Graph Relation Disambiguation algorithm** which is able to detect and disambiguate ambiguous relations in a Knowledge Graph. Our algorithm is based on the assumption that an ambiguous relation contains multiple semantically different groups of entities such that subject entities tend to densely connect to object entities only within the same semantic group and sparsely connect to entities in different semantic groups. Therefore, the proposed algorithm aims to group subject entities into subject clusters and object entities into object clusters for a relation and then discovers semantic groups in a relation based on the observed connection patterns between these clusters. The algorithm outputs a new Knowledge Graph that contains unambiguous relations from the input Knowledge Graph as well as new relations that were produced in the relation disambiguation step.

In order to test our hypothesis that relation disambiguation improves link prediction performance of the RESCAL model, as part of our framework, the new Knowledge Graph with disambiguated relations is also factorized with the RESCAL method. This factorization is then used to compare link prediction performance between the RESCAL model computed for the input Knowledge Graph and RESCAL model computed for the Knowledge graph with disambiguated relations.

To summarize, the Relation Disambiguation Framework provides the following main functionalities:

1. **RESCAL tensor factorization on the input Knowledge Graph**.

2. **Computation of relation-specific subject and object embeddings**.
3. **Identification of semantic subgroups with respect to a specific relation**.
4. **Detecting ambiguous relations**.
5. **Disambiguation of ambiguous relations**.
6. **Evaluation on the effect of Knowledge Graph relation disambiguation on link prediction performance of the RESCAL method**.

The first section in this chapter presents the general steps of the Relation Disambiguation algorithm. The idea and intuition behind each of the steps in the algorithm are afterwards explained in detail in the subsequent sections in this chapter. The last section of this chapter explains the required changes implemented in the framework for computing the link prediction metrics on a Knowledge Graph with disambiguated relations.

4.1 Relation Disambiguation Algorithm

Algorithm: Knowledge Graph Relation Disambiguation

input: $\boldsymbol{KG}$ - Knowledge graph tensor of shape $(\boldsymbol{N}, \boldsymbol{N}, \boldsymbol{M})$, where
$\boldsymbol{N}$ = number of entities in the knowledge graph,
$\boldsymbol{M}$ = number of relations in the knowledge graph

1 **Factorize** $\boldsymbol{KG}$ using RESCAL
2 **for** *each relation* $\boldsymbol{r}$ *in* $\boldsymbol{KG}$ **do**
3 Compute **embeddings for subject entities** in relation $\boldsymbol{r}$ using RESCAL factorization
4 Compute **embeddings for object entities** in relation $\boldsymbol{r}$ using RESCAL factorization
5 **Cluster subjects** for relation $\boldsymbol{r}$ by subject embeddings
6 **Cluster objects** for relation $\boldsymbol{r}$ by object embeddings
7 **Detect whether relation $\boldsymbol{r}$ is ambiguous** using subject clusters and object clusters
8 **if** *relation* $\boldsymbol{r}$ *is ambiguous* **then**
9 **Disambiguate relation $\boldsymbol{r}$** by splitting it onto $\boldsymbol{k_r}$ unambiguous relations
10 **else**
11 **Copy the slice for relation k into the output knowledge graph**
12 **end**
13 **end**

output: Disambiguated knowledge graph tensor of shape $(\boldsymbol{N}, \boldsymbol{N}, \boldsymbol{K})$ where $\boldsymbol{N}$ = number of entities in $\boldsymbol{KG}$, $\boldsymbol{K} >= \boldsymbol{M}$ and

$$\boldsymbol{K} = \sum_{\boldsymbol{r}=\boldsymbol{1}}^{\boldsymbol{M}} \boldsymbol{k_r},\ \boldsymbol{k_r} = \begin{cases} \textbf{Splits for relation } \boldsymbol{r}, & \text{if } \boldsymbol{r} \text{ is ambiguous relation} \\ \textbf{1}, & \text{otherwise} \end{cases}$$

4.2 Relation-Specific Subject and Object Embeddings

As mentioned in section 3.1, Knowledge Graphs are represented as a 3-dimensional tensor where the slice for each relation represents an adjacency matrix that specifies the existing edges within a relation. Therefore, the first step towards identifying semantic groups in a relation in the input Knowledge Graph is to encode the entities into embeddings which represent real-valued latent-feature vectors representations. These embeddings allow notion of sim-

ilarity to be defined between the entities with respect to a relation which can be used to cluster the entities in a relation into groups.

Semantic embeddings for the entities in the input Knowledge Graph are obtained by factorizing the input Knowledge Graph with the RESCAL method. As explained in section 3.2, the matrix E produced by the RESCAL factorization holds the global embeddings for the entities in the Knowledge Graph. However, entities in the Knowledge Graph usually appear in multiple relations which means that some latent features are more important than other latent features for certain relations. For example, the entity *Hillary Clinton* in the Freebase Knowledge Graph participates in relations like *organization_membership*, *religion* and *celebrity_friendship*. One can assume that important features for determining *Hillary Clinton's* membership to the organization *Girls Scouts of America* are her gender and nationality. However, her gender and nationality are probably far less discriminative features for predicting her friendships with celebrities for which her profession is a more important feature for this relation.
This example illustrates that each relation has certain semantics and the global entity embeddings are unable to capture the specific semantics for a given relation. Therefore, in order to obtain semantic embeddings for a relation, our aim is to compute entity embeddings specific for each relation that are able to capture the most discriminative features in each relation. Moreover, we would like to have two sets of relation-specific embeddings for a relation - the first set should contain embeddings for the subjects in a relation and the other set should contain embeddings for the objects in a relation. Additionally, each embedding should encode the information whether the corresponding entity appears as a subject or as an object in a given relation.

In order to obtain relation-specific subject and object entity embeddings for a given relation r, we use the interaction matrix R_r, also provided by the initial RESCAL factorization on the input Knowledge Graph.
Concretely, let e_k be the vector containing the global embedding for the k-th entity in the knowledge graph. This embedding vector is obtained by subsetting the k-th row of the global entity embedding matrix E, $e_k = E[k, :]$. Then, the relation-specific embedding for the subject i in the relation r is computed with the following equation:

$$s_{ir} = e_i R_r \tag{4.1}$$

On the other hand, the relation-specific embedding for object j in the relation r is computed with the following equation:

$$o_{jr} = R_r \left(e_j\right)^T \tag{4.2}$$

Based on the above equations, the relation-specific embeddings for the subjects in the relation r can be seen as a linear transformation applied on the global embeddings for the subjects in relation r by taking linear combinations with the rows of the interaction matrix R_r. The relation-specific embeddings for the objects in the relation r can also be seen as a linear transformation applied on the global embeddings for objects in relation r. The only difference is that the rotation on the global object embeddings is performed by taking linear combinations with the columns of the interaction matrix R_r.

Figure 4.1 presents the relation-specific subject and object embeddings for the entities in the relation *organization_membership* These embeddings are computed with the steps and formulas presented in this section. Their dimensionality is reduced to 2 dimensions by applying the t-SNE algorithm [20].
The left subplot in this figure shows that subject embeddings preserve the semantic similarity between the subjects entities in this relation. As can be seen in this subplot, the embeddings for country entities like Germany and Libya are located close in the latent-feature space to embeddings for other country entities such as United Kingdom and Central African Republic. On contrary, the countries are positioned far from the two other obvious groups of entities that can be noted in the latent-feature space: universities and famous persons like scientists, artists and politicians.
Similar can be said for the object embeddings shown on the right subplot where it can be seen that entities which represent association of countries like European Union and African Union lie in a neighborhood with entities that represent institutions in which countries participate such as World Bank and World Trade Organization. This embedding neighborhood lies far from the neighborhood which mainly consists of academic, art and female organizations.

4.3 Clustering Subject and Object Embeddings in a Relation

The relation-specific embeddings are used as an input to two independent clustering steps:

- **Clustering subjects in a relation with the Spectral clustering algorithm**. The **cosine similarities** between subject embeddings are used to construct an **ϵ-neighborhood similarity graph** for the subjects in the relation.
- **Clustering objects in a relation with the Spectral clustering algorithm**. The **cosine similarities** between object embeddings are

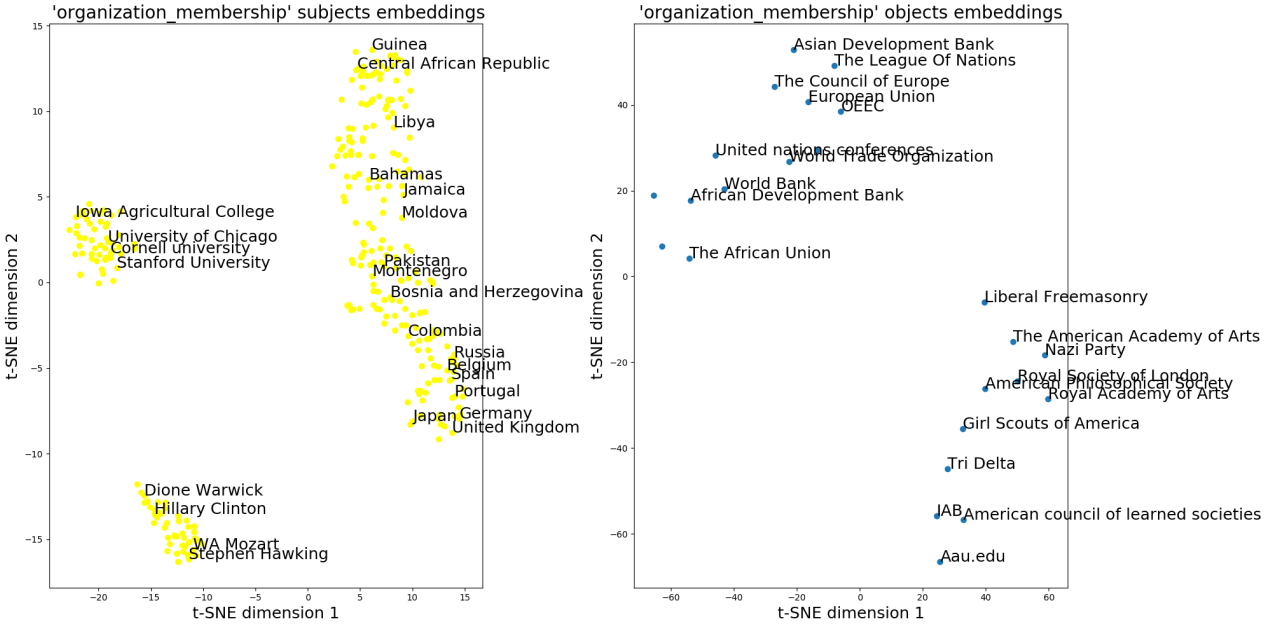

Figure 4.1: Subject embeddings and object embeddings for relation *organization_membership*

used to construct an ε**-neighborhood similarity graph** for the objects in the relation.

In both clustering steps, the **normalized Laplacian** matrix is constructed from the similarity graph and the number of clusters is chosen with the eigengap heuristic explained in section 3.3.2.
The figures 4.2 and 4.3 show the eigenvalues of the normalized Laplacian matrices obtained with application of the clustering steps for the relation *organization_membership*. In both figures, the first three eigenvalues are very small and the gap is largest between the 3rd and the 4th eigenvalue (in figure 4.2 first two eigenvalues are zeroes). The figure 4.1 shows that subject and object embeddings are well separated in the latent-feature space. Consequently, the eigenvalues for the graph Laplacians confirm the intuition that there are no overlapping clusters for the subjects and objects in this relation. By employing the eigengap heuristic, the optimal number of clusters for subjects and objects in this relation is set to 3.

Figure 4.4 visualizes the clustering assignments for subject entities and object entities in the relation *organization_membership* computed with Spectral clustering algorithm by setting the number of clusters to 3 for the both cluster-

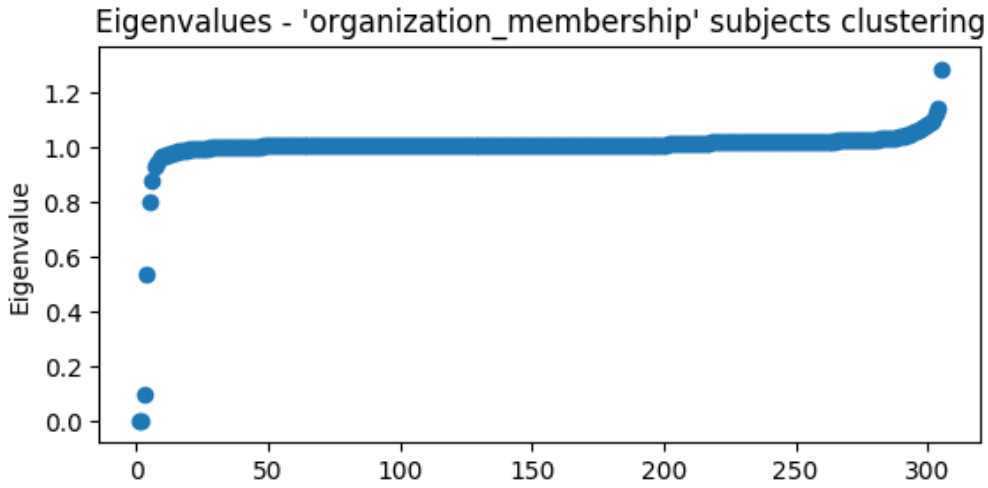

Figure 4.2: Normalized Laplacian eigenvalues - subjects clustering for relation *organization_membership*

ing steps. The color of each entity embedding in the left and right subplots of this figure marks the cluster for the corresponding entity. The left subplot illustrates that Spectral clustering is able to group semantically similar subjects into same clusters since the largest cluster consists of country entities and the two other clusters consist of university entities and famous individuals entities. The pink objects cluster on the right subplot has clear semantic meaning because it groups entities representing organizations whose members are different countries. The other two object clusters consist mainly of academic institutions. However, a bit ambiguity in the semantics of these two clusters is added by the existence of entities such as political parties or business organizations like IAB (Internet Advertising Bureau).

4.4 Identifying Ambiguous Relations

Ambiguous relations in the knowledge graph are detected with the following procedure:

1. A bipartite graph is constructed for a relation where one set of nodes consists of subject clusters in the relation and the other set of nodes consists of object clusters in the relation. The weight of an edge from a node of the subject cluster to a node of the object cluster in the bipartite graph represents the aggregated number of edges from the input knowl-

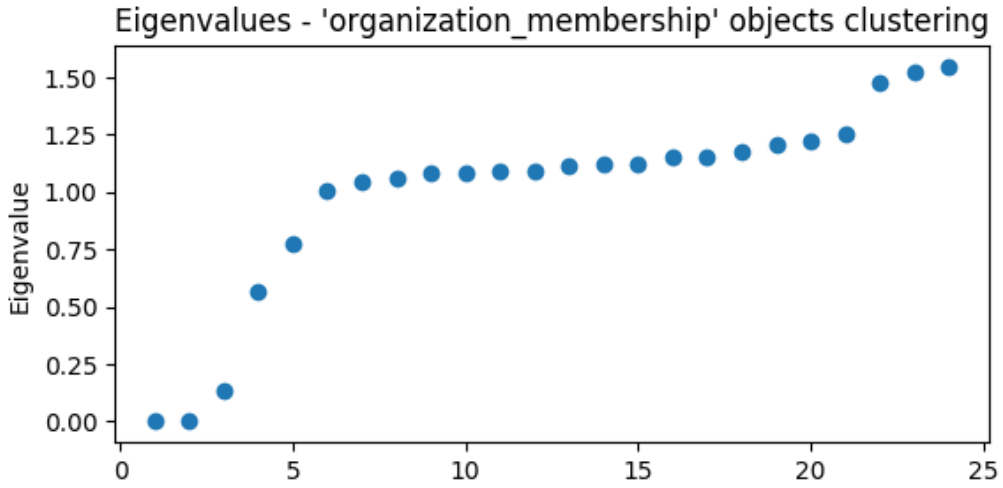

Figure 4.3: Normalized Laplacian eigenvalues - objects clustering for relation *organization_membership*

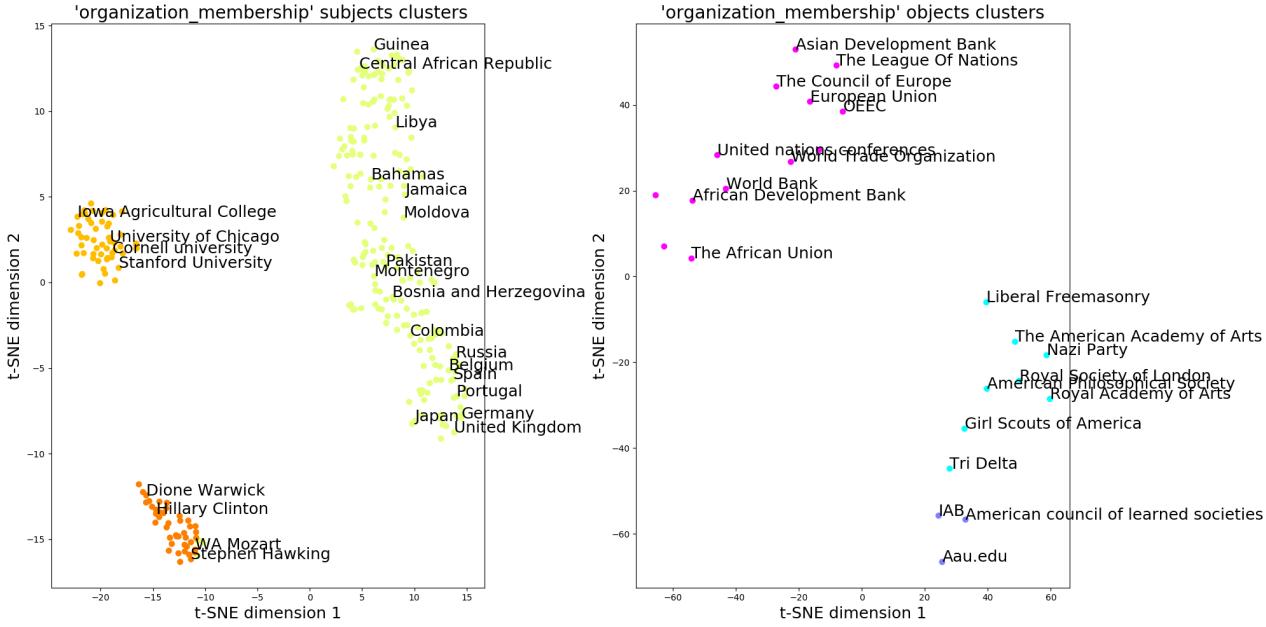

Figure 4.4: Subject clusters and object clusters for relation *organization_membership*

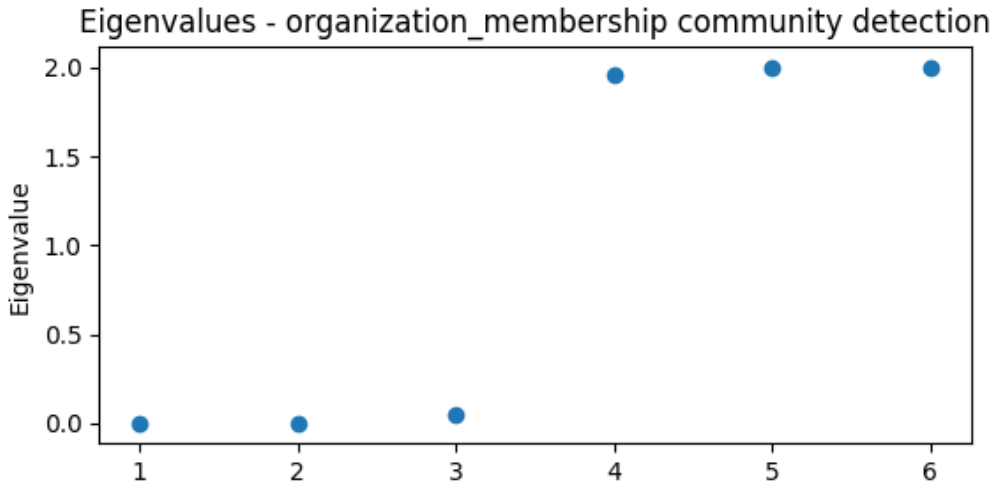

Figure 4.5: Normalized Laplacian Eigenvalues - Community detection on the bipartite graph for the relation *organization_membership*

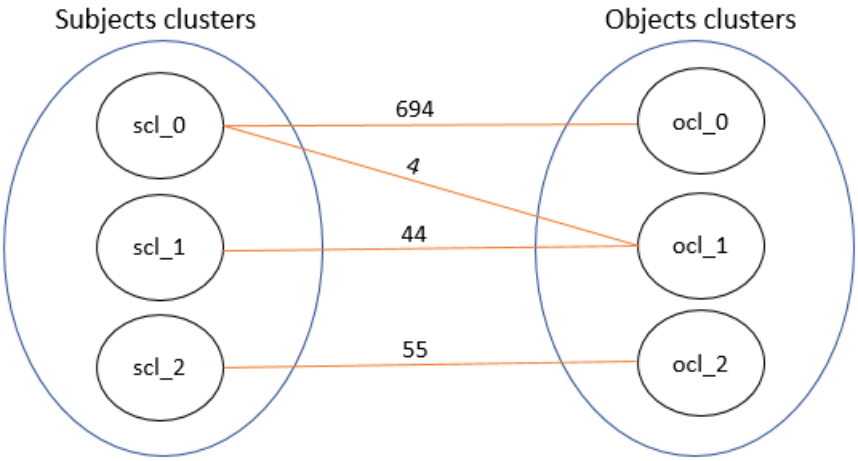

Figure 4.6: Bipartite graph for the relation *organization_membership*

edge graph going from any of the entities in the subject cluster node to any of the entities in the object cluster node.
The bipartite graph for the relation *organization_membership* is shown in the figure 4.6. The three nodes in the left part of the graph represent the three clusters obtained by clustering the subject embeddings in the relation. Similarly, the three nodes in the right part of the graph represent the three clusters obtained by clustering the object embeddings in the relation. The subject cluster labeled with scl_0 in the bipartite graph is the cluster which contains the country entities. The object cluster labeled with ocl_0 contains the different organizations in which the countries participate. Therefore, the edge weight **694** for the edge from *scl_0* to *ocl_0* means that there are in total 694 edges in the input knowledge graph from countries to the organizations whose members are different countries.

2. The bipartite graph from the previous step is used to detect communities in a relation. The communities are detected with Spectral clustering algorithm and the number of communities is again chosen with the eigengap heuristic. The similarity graph for the Spectral clustering is directly constructed from the edge weights in the bipartite graph.
   The eigenvalues of the normalized Laplacian matrix for the bipartite graph of the relation *organization_membership* are shown in figure 4.5. The largest eigenvalue gap between the 3rd and the 4th eigenvalue indicates the existence of three non-overlapping communities in this relation. These three identified communities are shown in figure 4.7. The cluster nodes which are part of the same community are displayed with same color in the plot.

3. **A relation is defined as ambiguous if the spectral clustering of the bipartite graph detects more than one community**.

Our aim by clustering RESCAL subject and object embeddings is to group semantically similar entities into same clusters. The pattern of connections between these clusters of subject and object entities in a relation is examined with the construction of the bipartite graph. If subject clusters nodes tend to densely connect to only certain object clusters nodes and sparsely connect to all other object clusters nodes, then this pattern is a clear indicator for the existence of semantically different groups of entities within a relation. Moreover, such bipartite graph exhibits strong community structure where each community contains entities from the input Knowledge Graph that share similar semantics. For example, the blue community in figure 4.7 for the relation *organization_membership* corresponds to a subgraph of the graph for the relation in the input Knowledge Graph whose entities are different countries and

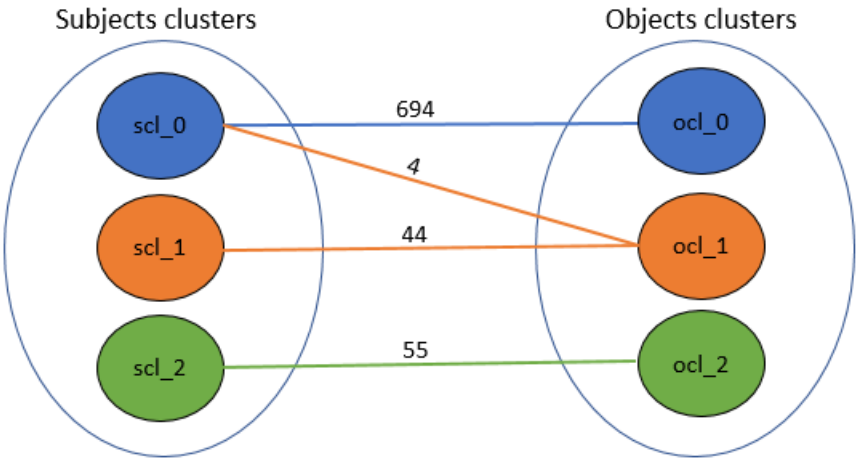

Figure 4.7: Detected communities in the relation *organization_membership*

organizations in which these countries participate. The orange community corresponds to another subgraph of the graph for the relation in the input Knowledge Graph whose entities are famous individuals and organizations like academies or political parties in which these individuals are members. The green community corresponds to the last non-empty subgraph of the graph for the relation in the input Knowledge Graph that contains universities and the academic organizations Aau.edu (Association of American Universities) and American council of learned societies in which universities are members. These subgraph additionally contains three edges from companies to IAB entity.

Based on the above example which justifies our intuition, we determine the ambiguity of a relation based on the number of detected communities in the bipartite graph with Spectral clustering. If only one community is detected in the bipartite graph, then the corresponding relation is not considered as ambiguous and remains part of the output knowledge graph. Otherwise, relations having more than one community are considered as ambiguous and are disambiguated with the algorithm described in the next subsection.

4.5 Relation Disambiguation

Ambiguous relations in the knowledge graph are disambiguated by applying the following two steps:

1. An ambiguous relation is splitted into k new unambiguous relations,

where k is equal to the number of communities in the bipartite graph for the relation.
Referring to the example for the relation *organization_membership*, this relation is splitted into the following three unambiguous relations:

- **organization_country_membership**
- **organization_university_membership**
- **organization_person_membership**

2. The final step in relation disambiguation involves reassignment of the edges from the ambiguous relation to the newly created unambiguous relations. The edges of an ambiguous relation are reassigned by considering the following two cases:

   - **Ambiguous relation edge in the dataset used for training**: Ambiguous edges in the training set are reassigned to only one of the newly created unambiguous relations based on the edges community memberships. Specifically, an edge for which the subject entity cluster and object entity cluster belong to the same community in the bipartite graph is assigned to the unambigous relation for the corresponding community. Otherwise, the given edge is assigned to the relation corresponding to the community in which the cluster of the edge object entity belongs.
   - **Ambiguous relation edge in the datasets used for validation and testing**: The relation disambiguation framework detects the ambiguity of the relations in the knowledge graph only based on the edges in the training set. Therefore, community memberships of subject entity and object entity for an edge found in the test set are unknown. Consequently, the unique unambiguous relation assignment for the edge can't be determined and the edge is splitted into k new edges, one for each of the newly created unambiguous relations. More formally, if there are n edges for an ambiguous relation in the test set, then after relation disambiguation the test set contains $n * k$ edges for the disambiguated relation.

4.6 Link Prediction for Knowledge Graph with Disambiguated Relations

The edge assignment procedure for ambiguous relations requires modified computation of the link prediction metrics, explained in section 3.1.1. Concretely,

in order to evaluate the effect of relation disambiguation on link prediction, the metrics are computed with the following procedure:

1. MRR, Hits@10, Hits@3 and Hits@1 scores are calculated for each edge in the test set.

2. Edges in the test set are grouped by triples having same values for *(subject, parent ambiguous relation, object)*. Hence, the number of groups is equal to the number of edges in the initial test set and each group consists of:

   - **One edge** if the edge belongs to a relation which was not identified as ambiguous with the steps in section 4.4
   - k_i **edges** introduced when reassigning an edge from the i-th ambiguous relation which was disambiguated into k_i new relations.

3. Link prediction metrics scores are summarized on group level by taking the maximum value for each metric over the scores for the edges in a group. For example, Hits@10 score for the i-th group on the disambiguated test dataset with test T edges is computed by taking the maximum value for the Hits@10 scores of the edges in the i-th group with the following formula:

$$Hits@10_{group_i} = \max_{edge_j \in group_i} Hits@10_j,\ j \in 1, ..., T. \tag{4.3}$$

4. The link prediction metrics for the entire dataset are computed by averaging the metrics scores over the groups in the test set.

An edge in the initial test set represents the existence of the link from subject entity to object entity in the graph for the edge relation. When the relation of the edge is detected as ambiguous, the disambiguation procedure aims to find the most plausible unambiguous relation to assign the edge to. Hence, the edge is replaced with an edge for each of the newly created unambiguous relations for the relation. As a consequence, not all edges in the new test set represent the true existence of links in the knowledge graph. Therefore, edges in the test set of the knowledge graph with disambiguated relations are grouped such that one group corresponds to an edge for an ambiguous relation in the initial test set. The most plausible unambiguous relation assignment for the edge is the relation in the group that provides highest scores for the link prediction metrics. The relation disambiguation framework has positive effect on link prediction if the averaged link prediction metrics on the test set produced with the relation disambiguation algorithm are higher than the averaged link prediction metrics on the initial test set.

Chapter 5

Experiments

5.1 Datasets and Implementation

5.1.1 Knowledge Graph Datasets

The experiments in this master thesis are performed on the following three benchmark Knowledge Graph datasets:

- **FB15k** is a dataset introduced in [3]. It is a subset of the Freebase Knowledge Graph which models 592,213 triples in which occur 14,951 unique entities and 1,345 unique relations.
- **FB15k-237** is a dataset introduced in [19]. It is a subset of the FB15k dataset that excludes redundant relations and triples in the test set for which corresponding inverse triple exists in the training set. This Knowledge Graph models 310,116 triples in which occur 14,951 unique entities and 237 unique relations.
- **WN18** dataset, also introduced in [3], is created from the WordNet lexical database that models different relations that exists between the synsets (sets of synonyms) in the English language. WN18 Knowledge graph models 151,442 triples that describe 18 different relations between 40,943 synsets in the English language.

In our experiments, each of these Knowledge Graph datasets were splitted into corresponding training set, validation set and test set. The percentage of triples in each set is shown in table 5.1. In order to have more samples during model training phase for the smaller datasets, the datasets FB15k-237 and WN18 contain higher percentage of triples in the training set than the FB15k dataset.

Dataset	Triples	Training set	Validation set	Test set
FB15k	592,213	81%	9%	10%
FB15k-237	310,096	87%	6%	7%
WN18	151,442	93%	3.5%	3.5%

Table 5.1: Training set, validation set and test set split ratios for the datasets used in the experiments

5.1.2 Implementation Details

The experiments in this thesis were implemented in Python, release version 3.7.3.
The RESCAL factorizations and link prediction evaluations were performed with the OpenKE package based on PyTorch [6]. The used PyTorch release version was 1.1.0. Furthermore, we customized the OpenKE package by implementing: L1 regularization for RESCAL model training, initialization of the RESCAL model with pretrained embeddings and modification of the link prediction evaluation procedures explained in section 4.6.
Spectral clustering and Hierarchical clustering algorithms were executed with the scikit-learn package implementation [15], release version 0.21.2. The eigendecomposition for Spectral clustering was obtained with the numpy package [14], release version 1.16.4. The Laplacian matrices were computed with the scipy package [21].

5.2 RESCAL Factorization

5.2.1 Model Training

All RESCAL models in the performed experiments were trained with the **pairwise loss** training procedure explained in section 3.2.1.3. The **margin-based ranking loss function with L1 regularization** was optimized over the dataset training set with the **Adam** optimization algorithm [7]: At each epoch, *num_batches* times is sampled a *batch_size* of positive examples and corresponding *batch_size* of negative examples (one per each positive example), where

$$batch\_size = |training\_set|/num\_batches.$$

For each of the datasets, we performed grid search for the hyperparameters *learning_rate* and *num_batches*. The best convergence of the training loss function was achieved for $learning\_rate = 0.0001$ and $num\_batches = 100$.

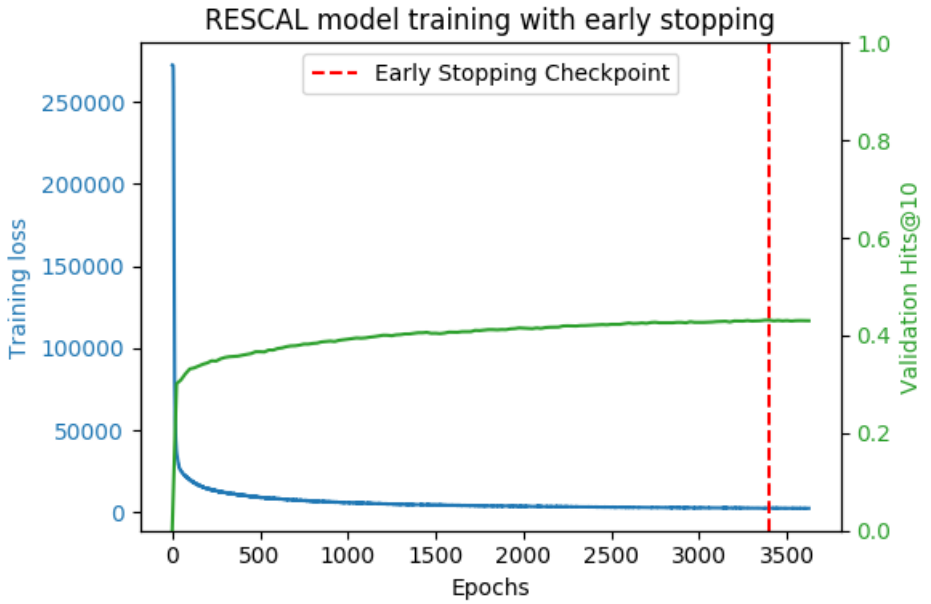

Figure 5.1: RESCAL model training with early stopping on FB15k-237 dataset

The initial number of training epochs was set to 10000. However, in order to prevent overfitting, for each dataset we implemented the following early stopping procedure: On each 25-th training epoch, we store the RESCAL model at the given epoch and evaluate the Hits@10 score on the validation set. If the Hits@10 score does not improve after five consecutive evaluations, the training procedure is stopped and the model computed at the epoch that achieved highest Hits@10 score on the validation set is returned.
The early stopping procedure for the FB15k-237 dataset is illustrated on figure 5.1. The blue line in the plot corresponds to the left y-axis and shows the training loss achieved in each epoch. The green line on the plot corresponds to the right y-axis and it shows the Hits@10 score on the validation set computed on every 25-th training epoch. As can be seen on this figure, the training loss decreases fast in the initial epochs and it starts to decrease very slowly after the 2000-th epoch. On the other hand, as expected, the Hits@10 score also increases fast in the initial epochs and after the 3400-th training epoch it begins to decrease. Since the decrease happens on five consecutive evaluations, the training procedure is stopped and the RESCAL model computed in the 3400-th training epoch is returned.

5.2.2 Hyperparameter Tuning and Model Selection

As mentioned in section 3.2, the key hyperparameter of the RESCAL model is the embedding dimension d. The different number of entities, relations and triples in the Knowledge Graph datasets used in the experiments require separate tuning of the embedding dimension for each dataset. By fixing the embedding dimension, we additionally search for the optimal L1 regularization constant for the given embedding dimension. Therefore, for each dataset, a grid search is performed over different pairs of embedding dimensions and regularization constants by training a separate RESCAL model for the given hyperparameter configuration with the procedure described in the previous section. An optimal pair of hyperparameters for a dataset is the pair for which the corresponding RESCAL model achieves highest Hits@10 score on the validation set. This RESCAL model is selected as the most representative model for the dataset and it is used as an input for the experiments performed in the next steps of the Relation Disambiguation framework.

The hyperparameter tuning procedure for FB15k-237 dataset is shown in figure 5.2. As can be seen in the plot, increasing the embedding dimension improves the link prediction performance on the validation set, up to embedding dimension of 50. Further increase in the embedding dimension results in worse link prediction performance. This implies that RESCAL models with embedding dimension greater than 50 are too complex for the FB15k-237 dataset and result in overfitting. The regularization constant of 0.00001 consistently produces best results across the different dimensions. Therefore, as a best model RESCAL model for the FB15k-237 dataset is chosen the model trained with the embedding dimension of 50 and regularization constant equal to 0.00001. The optimal embedding dimensions chosen by this procedure for the FB15k and WN18 datasets were 100 and 150, respectively. The higher embedding dimension for the FB15k can be justified by the fact that FB15k Knowledge Graph models almost twice as many triples and contains 1,108 more relations compared to the FB15k-237 Knowledge Graph. Although WN18 Knowledge Graph models much smaller number of triples than FB15k and FB15k-237, it contains almost three times more entities in its Knowledge Graph. Therefore, a higher embedding dimension is needed to properly model the entities and the interactions between their latent features in the WN18 Knowledge Graph.

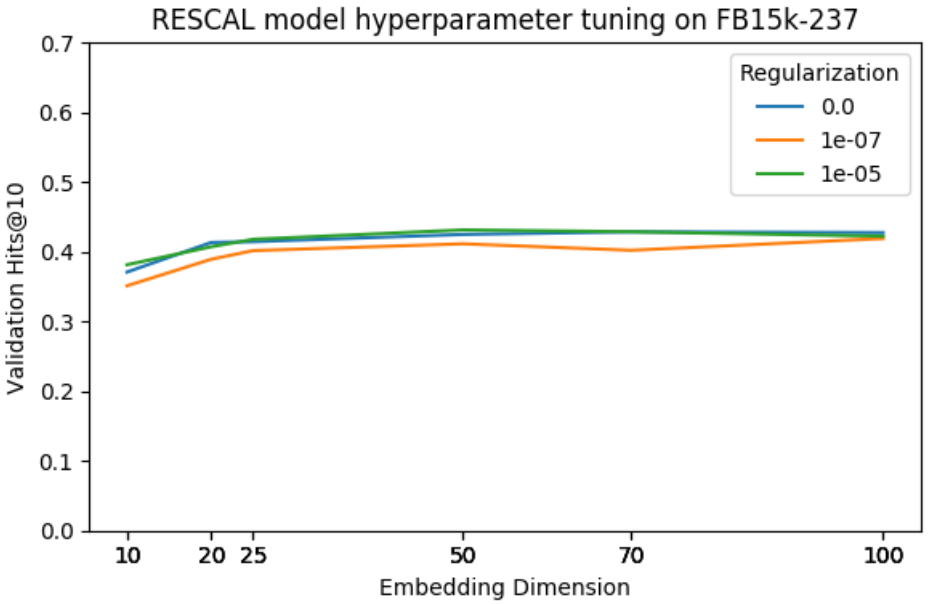

Figure 5.2: RESCAL model Hyperparameter tuning on FB15k-237 dataset

5.3 Clustering Relation-Specific Subject and Object Embeddings

The RESCAL factorization of the input Knowledge Graph dataset, obtained with the training and model selection procedures described in the previous section, is used to compute the relation-specific subject and object embeddings. These embeddings are calculated as described in section 4.2.

The clustering steps described in section 4.3 are executed with two clustering algorithms: Spectral clustering and Hierarchical clustering. Each experiment for a clustering algorithm and its corresponding hyperparameters outputs a clustering assignments for the subjects and clustering assignments for the objects in a relation. In order to evaluate the effect of the clustering algorithms and their hyperparameters, the next steps of the Relation Disambiguation algorithm are executed for each pair of clustering assignments for subjects and objects in a relation. Therefore, a new output Knowledge Graph with disambiguated relations is produced for each clustering experiment performed in this section.

In this thesis, the cosine similarity is used as a similarity measure between entities when evaluating the experiments for spectral clustering and hierarchical clustering, respectively. The motivation for using the cosine similarity comes from the L1 regularization that is added to the training loss function for the RESCAL model. L1 regularization tends to produce sparse entity embeddings and sparse relation interaction matrices for the RESCAL model. Therefore, when two entities are compared with cosine similarity by their relation-specific embeddings, only the latent features having non-zero values in both embeddings are relevant in order to determine the similarity between the entities.

In this section we describe the experiments performed with Spectral clustering algorithm and Hierarchical clustering algorithm. The effect of the clustering algorithms on the detected ambiguous relations and on the link prediction performance are then discussed in sections 5.4 and 5.5.

5.3.1 Spectral Clustering

In order to perform the clustering steps described in section 4.3 with the Spectral clustering algorithm, two similarity graphs are constructed: one for the subjects in a relation and the another one for the objects in a relation. The similarity graphs are constructed based on the relation-specific subject embeddings and relation-specific object embeddings, respectively. In this thesis, we experiment with two different similarity graphs: **k-nearest neighbor graph** and **ε-neighborhood similarity graph**.

5.3.1.1 k-Nearest Neighbor Graph

For the FB15k-237 dataset, we constructed multiple different subjects and objects k-nearest neighbor similarity graphs for values of k in the range from 3 to 10. For a fixed value of k, the similarity graphs for the subjects and the similarity graphs for the objects are constructed with the chosen value of k for each of the 237 relations that exist in the FB15k-237 dataset. However, all of the constructed similarity graphs were either sparse or dense. Sparse similarity graphs have high number of connected components and each connected component contains only a small number of entities. Spectral clustering on such sparse similarity graphs produces clustering assignments such that each entity is assigned to a separate cluster. If both similarity graphs for the subjects and objects in a relation are sparse, then the relation disambiguation step explained in section 4.5 will produce a large number of new relations. Learning interaction matrices that properly capture the semantics for these newly-created relations is not feasible since each relation contains a small number of

edges in its adjacency matrix. On the other hand, the Spectral clustering on dense similarity graphs produces only one cluster since most of the entities are densely connected. Therefore, dense subject and object similarity graphs for a relation can lead to not detecting a potentially ambiguous relation as ambiguous. Because we were unable to determine a value of k that will produce balanced number of subject and object clusters for most of the relations in the FB15k-237 dataset, the next steps in the Relation Disambiguation algorithm based on Spectral clustering were performed only with ε-neighborhood similarity graphs.

5.3.1.2 ε-Neighborhood Similarity Graph

For the construction of the ε-neighborhood similarity graph, we experimented with ε values of 0.3, 0.5, 0.7 and 0.8 as a threshold values for the cosine similarity between the entities in the similarity graph. For example, if $\varepsilon = 0.3$, then the similarity graph contains only edges between pairs of entities having a cosine similarity greater or equal than 0.3. For a given ε value, ε-neighborhood similarity graphs are constructed for the subjects and the objects in each relation of the knowledge graph datasets used in the experiments.

The different values for the ε-threshold strongly influence the number of subject and object clusters obtained for a relation.
Smaller ε values result in connecting less similar entities which leads to an existence of more edges in the similarity graph. Hence, the resulting similarity graph contains only a few connected components with higher number of entities in each component. Consequently, Spectral clustering produces small number of clusters on such similarity graphs.
On the contrary, higher ε values result in connecting smaller number of entities in the similarity graph. Therefore, similarity graphs constructed with a high ε threshold are sparse and contain a high number of connected components with small number of entities in each component. Hence, higher number of clusters are obtained on sparse similarity graphs with Spectral clustering.

The influence of the ε value on the number of clusters is additionally illustrated with an example for the relation *award_ceremony* from the FB15k-237 Knowledge Graph. The edges in this relation specify the ceremonies on which different awards are honored.
Figure 5.3 shows the subject and object Spectral clustering assignments for this relation computed by constructing the ε-neighborhood similarity graphs with $\varepsilon = 0.7$. The similarity graph for the subjects leads to identifying three clusters of awards among subjects: grammy awards, emmy awards and academy

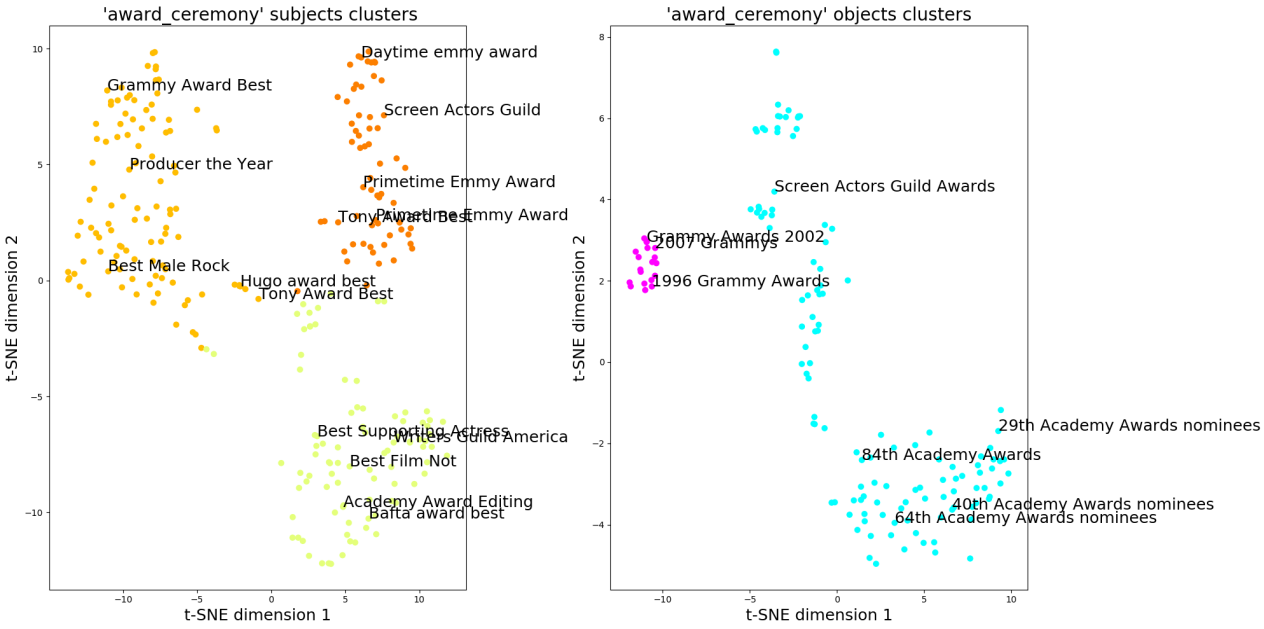

Figure 5.3: Subject clusters and object clusters in the relation *award_ceremony* - $\varepsilon = 0.7$

awards. The similarity graph for the objects leads to identifying two clusters of ceremonies among the objects: ceremonies for grammy awards and ceremonies for emmy/academy awards.
Figure 5.4 shows that constructing the similarity graph with the value of ε decreased to 0.3 for the relation *award_ceremony* leads to grouping all awards into one subject cluster and all ceremonies into one object cluster.

5.3.1.3 Eigengap Heuristic

As already mentioned in sections 4.3 and 4.4, the number of clusters in the steps of the proposed Relation Disambiguation algorithm that involve Spectral clustering is chosen with the eigengap heuristic explained in section 3.3.2. In order to implement this heuristic, the eigengap score needs to be calculated for each possible choice of the number of clusters. The computation of the eigengap scores requires choosing a value for the weighting hyperparameter w in equation 3.9.
A value of w close to zero neglects the cumulative eigenvalues sum in the equation 3.9 and favorites selection of the eigenvalue gap that occurs within the later indices in the sorted eigenvalues array, even if the selected eigengap is not the first larger eigengap. Hence, a small value for w can produce higher

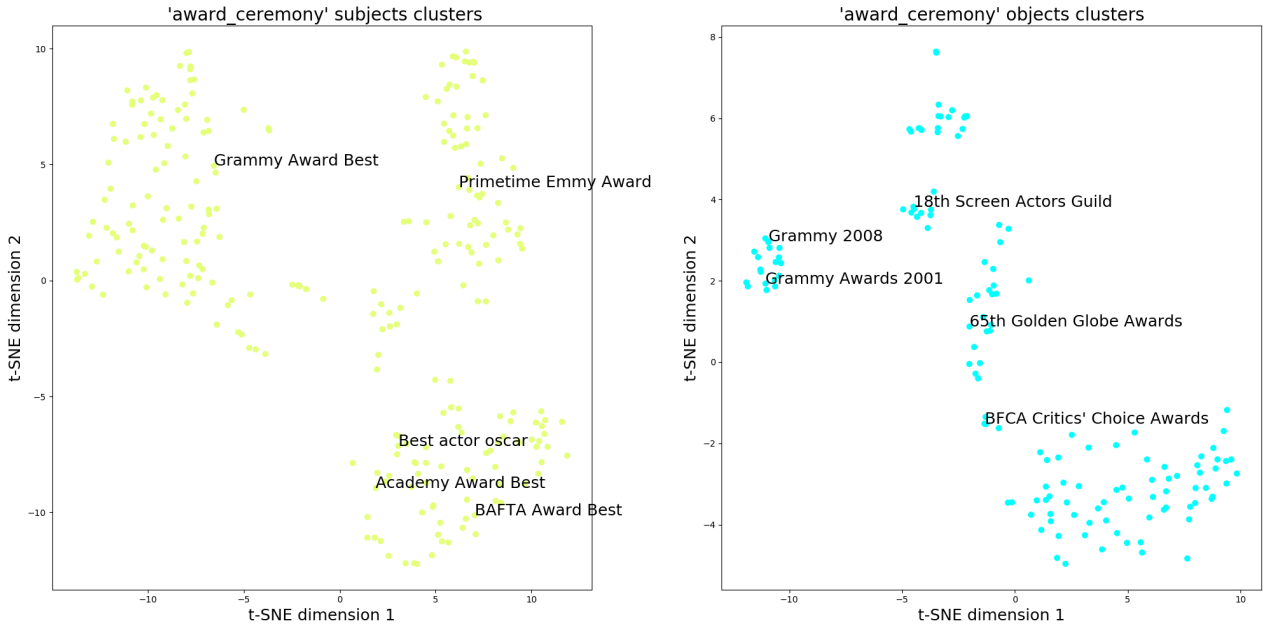

Figure 5.4: Subject clusters and object clusters in the relation *award_ceremony* - $\varepsilon = 0.3$

number of clusters. The issue with choosing a small value for w is illustrated with the example shown in figure 5.5. This figure shows the eigenvalues of the normalized Laplacian matrix obtained by clustering the objects in the relation *award_nomination* with the value of 0.5 for the ε-neighborhood graph. As can be seen in this figure, the first large eigengap occurs between the 1st and the 2n eigenvalue. However, there is another large eigengap that occurs between the 3rd and the 4th eigenvalue. If $w = 0.1$, then the eigengap heuristic selects the optimal number of clusters to 3. This means that the selection is based on the second largest eigengap which contradicts with the definition of the eigengap heuristic.
On the other hand, a value of w close to 1 favorites selection of the first small non-zero eigengap which can be problematic if there exists a higher eigengap that occurs within the next few indices in the sorted eigenvalues array. Such example is presented in figure 5.6. This figure shows the eigenvalues of the normalized Laplacian matrix obtained by clustering the objects in the relation *place_of_birth*. The first gap in this plot occurs between the 1st and the 2nd eigenvalue. However, this gap is very small and is close to zero. The first larger gap in this plot occurs between the 2nd and the 3rd eigenvalue. By setting w to 0.9 the eigengap heuristic outputs 1 as the optimal number of clusters.

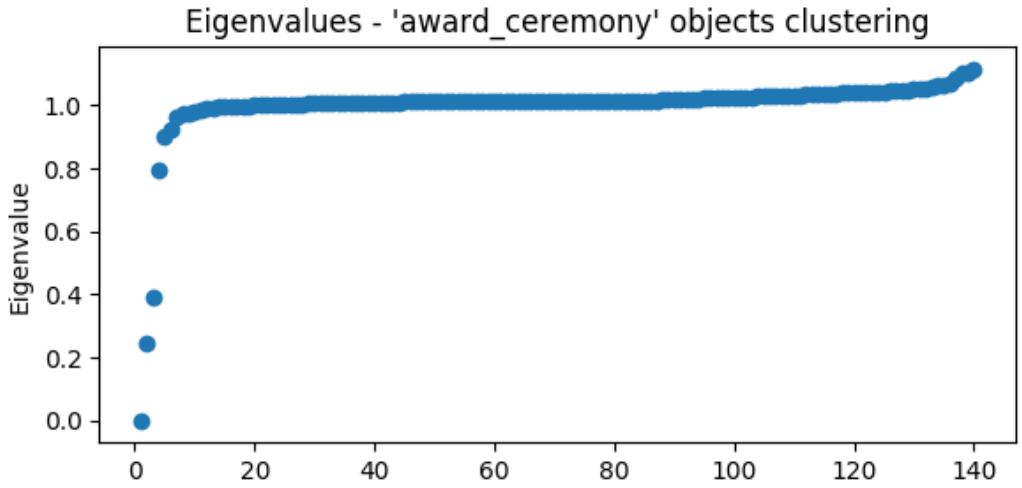

Figure 5.5: Normalized Laplacian eigenvalues - objects clustering for the relation *award_ceremony*

This implicates that instead of finding the first larger eigengap, the eigengap heuristic incorrectly found the first small non-zero gap.
The value of $w = 0.4$ produces the optimal number of clusters in the both situations. Therefore, our Relation Disambiguation algorithm uses 0.4 as a value for the hyperparameter w in the eigengap heuristic.

For some Spectral clustering experiments, it was found that there is no well-defined gap between the eigenvalues of the normalized Laplacian matrix. The gap is not well-defined because the differences between all eigenvalues are very small and approximately the same. In [8] it is shown that this scenario occurs when there are many overlapping clusters in the data. Therefore, in order to prevent selecting more than one cluster in such cases, we introduced one more hyperparameter in the eigengap heuristic called **eigengap threshold**. We set its value to 0.05 for all datasets used in the experiments. This hyperparameter has the following meaning: An index corresponding to an eigengap is considered as a candidate for the optimal number of clusters if and only if the eigengap at the given index is greater than the eigengap threshold. If no eigengap exists being greater than the eigengap threshold, then one cluster is returned by default.

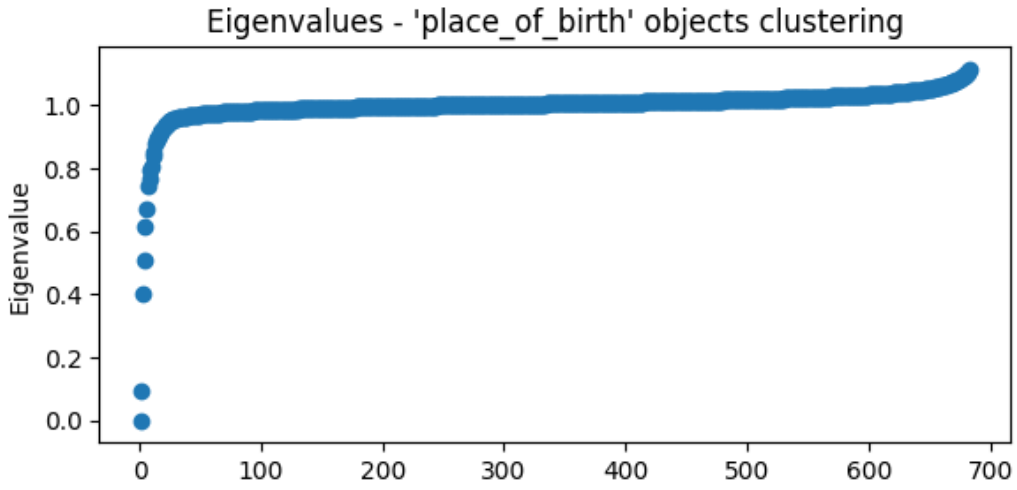

Figure 5.6: Normalized Laplacian eigenvalues - objects clustering for the relation *place_of_birth*

5.3.2 Hierarchical Clustering

In each Hierarchical clustering experiment performed over the input relation-specific embeddings, several clustering assignments were computed by iterating the value for the hyperparameter number of clusters in the range between 1 and 20. Each clustering assignment was evaluated with the Silhouette measure, described in section 3.4. The assignment which achieved the highest Silhouette score was chosen to be the optimal clustering assignment.

A major drawback of the above procedure for choosing the optimal clustering assignments with Hierarchical clustering is that it tends to produce higher number of clusters compared to the eigengap heuristic. This leads to having imbalanced clusters such that the majority of the entities are grouped only into one cluster. An example of such behavior is shown in figure 5.7 for the relation *gdp_nominal_currency* found in the FB15k-237 Knowledge graph. The left subplot illustrates that Hierarchical clustering identified two clusters among the subjects in this relation. However, the yellow cluster contains the majority of the subjects in this relation. The right subplot illustrates that the although each of the three objects are quite far from each other in the latent-feature space, two of them are grouped into the same cluster with Hierarchical clustering. The eigengap heuristic is able to avoid such clustering assignments because for a sufficiently high value of ε, the ε-neighborhood graph consists

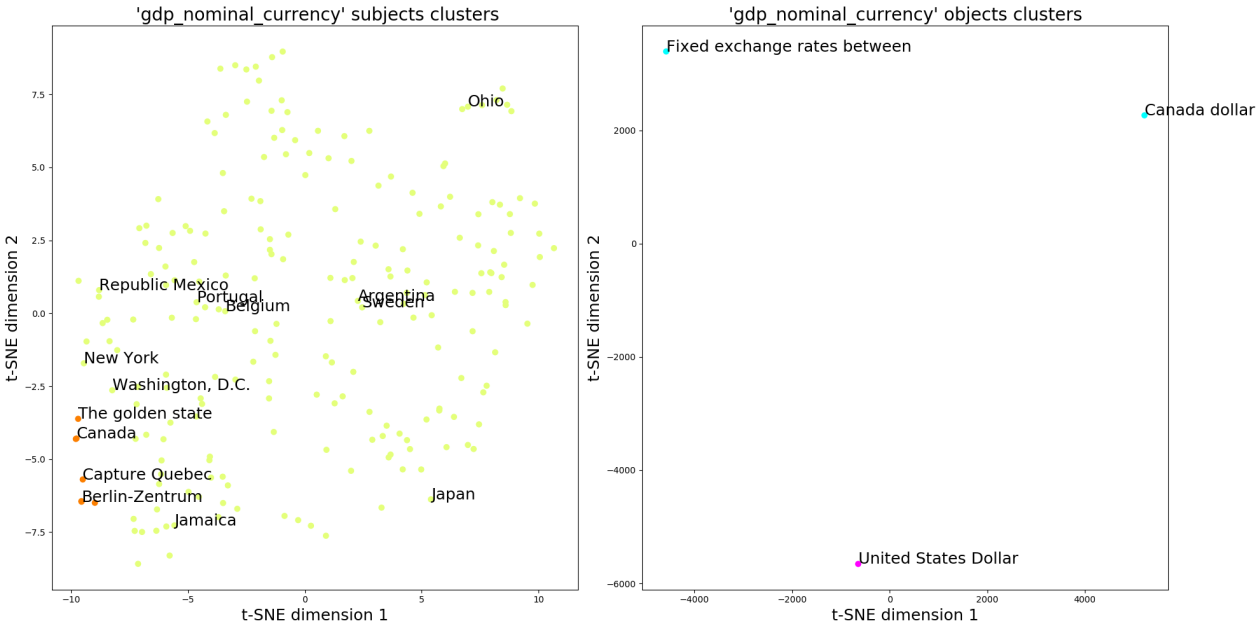

Figure 5.7: Hierarchical clustering assignments in the subjects and the objects in the relation *gdp_nominal_currency*

only of zeros when the entities are far from each other in the latent feature space. As a consequence, the eigenvalues of the corresponding Laplacian matrix are all zeros and the eigengap heuristic will output one cluster as optimal number of clusters.

5.4 Ambiguous relations

Table 5.2 summarizes the percentage of detected ambiguous relations in the FB15k and FB15k-237 datasets obtained with the different Spectral clustering and Hierarhical clustering experiments applied on the subject and on the objects in the relations found in these Knowledge Graphs. The shortcut SC in the table stands for the Spectral clustering algorithm and the shortcut HC stands for the Hierarchical clustering algorithm.
As can be seen in this table, increasing the ε value for the ε-neighborhood similarity graphs used in Spectral clustering increases the percentage of detected ambiguous relations in both Knowledge Graphs. Based on the algorithm for detecting ambiguous relations in our framework, described in section 4.4, a relation is detected as ambiguous if its bipartite graph contains more than one community. Consequently, a relation that has only one subject cluster and

only one object cluster is not ambiguous. Section 5.3.1.2 shows that Spectral clustering for similarity graphs constructed with small ε values results in a small number of clusters. Therefore, the smallest percentage of ambiguous relations is detected for a value of ε equal to 0.3 because the Spectral clustering algorithm applied on the similarity graphs constructed with this ε value results in one subject cluster and one object cluster for most of the relations in both graphs. Section 5.3.1.2 also illustrates that the increase of the value of ε for the construction of the ε-neighborhood similarity graph produces higher number of clusters with Spectral clustering. Consequently, when both similarity graphs for the subjects and the objects in a relation are constructed with a high value of ε, a high number of subject clusters and object clusters is obtained. At the same time, the high number of subjects clusters and objects clusters for a relation increases the number of nodes and edges in the bipartite graph constructed for a relation. This increases the chances for identifying more than one community in the bipartite graph for the relation. Therefore, highest percentage of ambiguous relations with Spectral clustering is detected for value of ε equal to 0.8 which is the highest value used in our experiments. The Hierarchical clustering produced highest percentage of detected ambiguous relations in both Freebase Knowledge Graphs. This is strongly correlated with the imbalanced number of clusters obtained for the subjects and the objects in the relations with Hierarchical clustering being explained in section 5.3.2.

The WN18 Knowledge Graph contains almost three times more entities than the Freebase Knowledge Graph. This fact causes the eigendecomposition of the Laplacian matrices for some relations in this dataset to require high amount of numerical operations. We were able to compute eigendecomposition of the Laplacians for all relations only for ε values of 0.5 and 0.8 because these values produced sparse similarity graphs. Therefore, the results on the WN18 dataset are shown only for Spectral clustering experiments performed with ε values of 0.5, 0.8 and Hierarchical clustering.
Only 1 ambiguous relation (out of 18) was identified in the two experiments performed with Spectral clustering. On the other hand, 11 relations were identified as ambiguous with Hierarchical clustering.

5.4.1 Semantic Evaluation

In this section we would like to give a notion about the detected ambiguous relations and the disambiguation results from a semantically point of view. We explain the semantics of 5 out of 11 relations which were identified as ambiguous in the FB15k-237 dataset with the following configuration: The

	SC, eps = 0.3	**SC, eps = 0.5**	**SC, eps = 0.7**	**SC, eps = 0.8**	**HC**
FB15k	2.5%	6%	10.5%	12.5%	40%
FB15k-237	2.5%	4.7%	12%	16%	46%

Table 5.2: Percentage of detected ambiguous relations in FB15k and FB15k-237 datasets with the different clustering experiments

subjects and objects in the relations were clustered with Spectral clustering and ε value of 0.5 was used for the construction of the subjects and objects similarity graphs in all of the 237 relations in this dataset. We evaluate the semantics of the following relations:

- **location_contains**: This relation specifies the location of famous institutions such as universities and businesses. The majority of locations in the Freebase dataset are places in USA. Therefore, the subject entities in this relation are grouped into two clusters. The first subject cluster contains all the locations from USA and the second subject cluster contains locations from the rest of the world. Likewise, the object entities are also grouped into two clusters. The first object cluster contains all institutions in USA and the second object cluster contains the institutions from the rest of the world. The relation disambiguation step (section 4.5) splits this relation into two unambiguous relations:
  - **location_USA_contains**
  - **location_world_contains**
- **organization_membership**: This relation is explained in the introduction and the newly created unambiguous relations are stated in section 4.5.
- **educational_institution_campus**: This relation specifies the campuses of the different universities. Similar as for the relation location_contains, the universities from USA make the majority of the universities in the dataset. This relation is disambiguated by our framework into two unambiguous relations:
  - **educational_institution_USA_campus**
  - **educational_institution_world_campus**
- **award_nominated_for**: This relation specifies the award nominations for two categories of subject entities: musicians and movies. Therefore, it is disambiguated by our framework into two unambiguous relations:

  - **award_nominated_movie_for**
  - **award_nominated_musician_for**

- **sports_team_roster**: Subjects in this relation are different positions that exists in football, basketball and american football like midfielder and quarterback. The objects in this relation are different teams which have players for the subjects positions. This relation is disambiguated into two new relations:
  - **sports_football_team_roster**
  - **sports_american_football_and_basketball_team_roster**

5.5 Link Prediction

An execution of the Relation Disambiguation algorithm for an input Knowledge Graph and a choice of one of the supported clustering algorithms with its corresponding hyperparameters produces a new output Knowledge Graph with disambiguated relations. In order to evaluate the effect of the relation disambiguation on link prediction for the RESCAL method, each Knowledge Graph with disambiguated relations is factorized with the RESCAL method.

5.5.1 RESCAL factorization on a Knowledge Graph with disambiguated relations

The embedding dimension of the RESCAL factorization on the Knowledge Graph with disambiguated relations is set to the value of the optimal embedding dimension found for the corresponding input Knowledge Graph. For example, the embedding dimension is set to 50 in the RESCAL models for all FB15k-237 Knowledge Graphs with disambiguated relations, since this embedding dimension was found as optimal in section 5.2.2 for the FB15k-237 dataset. Additionally, the optimal RESCAL model for a dataset was used to perform the following initialization steps in all RESCAL models for Knowledge Graphs with disambiguated relations produced for the same dataset:

- The entity embeddings were initialized with the values of the entity embeddings from the optimal RESCAL model for the dataset.
- The interaction matrices for relations that correspond to unambiguous relations in the input Knowledge Graph were initialized with the values of the corresponding unambiguous relations interaction matrices from the optimal RESCAL model for the dataset.

After completion of the initialization steps, the RESCAL models for the Knowledge Graphs with disambiguated relations were trained with the early stopping procedure described in section 5.2.1.

5.5.2 Evaluation

The RESCAL factorization of a given Knowledge Graph was used to compute the link prediction metrics on a test set that was not used during the RESCAL training procedure. The test sets for the Knowledge Graphs with disambiguated relations were derived with the procedure explained in section 4.5.
In this section we examine the effect of the Relation Disambiguation algorithm on the link prediction results for the datasets used in our experiments. Additionally, in order to examine the effect of the Relation Disambiguation algorithm only on the detected ambiguous relations, for each execution of the Relation Disambiguation algorithm for an input Knowledge Graph we performed the following evaluation: A new test set was created for the input Knowledge graph which is a subset of the initial test set. This test set contains edges only from relations which were identified as ambiguous by the algorithm. Similarly, a new test set was created for the output Knowledge Graph which is a subset of the test set of the output Knowledge Graph that only contains edges from relations that were introduced in the relation disambiguation step (section 4.5). Then, the link prediction metrics for the new test of the input Knowledge Graph were computed by the corresponding RESCAL model for this Knowledge Graph. Similarly, the link prediction metrics for the new test of the output Knowledge Graph with disambiguated relations were computed by the corresponding RESCAL model for the output Knowledge Graph with disambiguated relations.

Table 5.3 presents the link prediction results for the FB15k Knowledge Graphs. The first row shows the scores for the input FB15k dataset. Each of the following rows shows scores for a FB15k Knowledge Graph with disambiguated relations that is obtained for a choice of clustering algorithm and its hyperparameters. Link prediction scores are higher in all Knowledge Graphs with disambiguated relations than in the input FB15k Knowledge graph. The best scores are obtained when subjects and objects in the relations are clustered with the Hierarchical clustering algorithm. These results are very close to the best results with the Spectral clustering algorithm, obtained for the value of ε equal to 0.5. This shows that for the experiments performed with Spectral clustering, the number of detected ambiguous relations is not correlated with the improvements in the link prediction results for FB15k since the ε values

of 0.7 and 0.8 lead to detecting a higher number of ambiguous relations (as shown in table 5.2), but result in a worse prediction.
Table 5.4 shows the link prediction results for the ambiguous relations detected with the different clustering algorithms on the FB15k dataset. The first cell in each row describes the clustering algorithm used in the corresponding experiment. Then, the next cells are splitted into two cells: the top cell represents the score of the RESCAL model computed only on the ambiguous relations detected with the corresponding clustering algorithm for the FB15k Knowledge Graph and the bottom cell represents the score of the corresponding RESCAL model for the output Knowledge Graph with disambiguated relations. The values in the top cell of each row can be interpreted as a confidence in the ambiguity of the detected ambiguous relations. Based on the RESCAL method, one can assume that link prediction scores should be smaller for ambiguous relations than for unambiguous relations. Therefore, small values in the top cells indicate a high confidence in the ambiguity of the detected ambiguous relations. An ε value of 0.3 for Spectral clustering produces clusters that contain many dissimilar entities. Therefore, the ambiguous relations detected based on such clusters are not trustworthy which is confirmed by the high values of the top cells in the first row in table 5.4. Moreover, the bottom cells show that the disambiguation of these relations leads to worse link prediction scores on these relations. These values justify our interpretation of the link prediction scores as a confidence in the ambiguity of the detected relations. When the value for ε is increased to 0.5, the link prediction scores for the FB15k dataset are smaller compared to the link prediction scores for 0.3 value of ε. Our higher confidence in the ambiguity of the relations produced with $\varepsilon = 0.5$ is justified by the increase in the link prediction scores after the disambiguation of these relations and by the fact that the best link prediction scores with Spectral clustering for the FB15k Knowledge Graph are obtained with this ε value.

Our Relation Disambiguation algorithm also improves the link prediction performance on the FB15k-237 and WN18 datasets. The results for the FB15k-237 dataset are shown on tables 5.5 and 5.6. The results for WN18 dataset are shown on tables 5.7 and 5.8.

Dataset	MRR	Hits@10	Hits@3	Hits@1
FB15k	0.414	0.664	0.482	0.285
FB15k-SC, eps=0.3	0.431	0.685	0.505	0.298
FB15k-SC. eps=0.5	0.438	0.691	0.514	0.304
FB15k-SC, eps=0.7	0.427	0.681	0.499	0.293
FB15k-SC, eps=0.8	0.431	0.685	0.504	0.297
FB15k-HC	0.441	0.691	0.517	0.309

Table 5.3: Link prediction results for the FB15k datasets

Dataset	MRR	Hits@10	Hits@3	Hits@1
FB15k-SC, eps=0.3	0.556	0.883	0.634	0.419
	0.527	0.776	0.603	0.392
FB15kSC, eps=0.5	0.442	0.734	0.546	0.28
	0.487	0.776	0.599	0.325
FB15k-SC, eps=0.7	0.536	0.729	0.586	0.436
	0.548	0.729	0.594	0.453
FB15k-SC, eps=0.8	0.579	0.763	0.631	0.482
	0.565	0.758	0.605	0.471
FB15k-HC	0.409	0.688	0.491	0.261
	0.449	0.725	0.54	0.299

Table 5.4: Link prediction results for the ambiguous relations in the FB15k datasets

Dataset	MRR	Hits@10	Hits@3	Hits@1
FB15k-237	0.263	0.424	0.29	0.183
FB15k-237-SC, eps=0.3	0.266	0.426	0.294	0.185
FB15k-237-SC. eps=0.5	0.268	0.431	0.293	0.186
FB15k-237-SC, eps=0.7 3	0.271	0.434	0.296	0.189
FB15k-237-SC, eps=0.8	0.272	0.434	0.298	0.191
FB15k-237-HC	0.282	0.448	0.307	0.2

Table 5.5: Link prediction results for the FB15k-237 datasets

Dataset	MRR	Hits@10	Hits@3	Hits@1
FB15k237-SC, eps=0.3	0.293	0.445	0.321	0.215
	0.302	0.465	0.335	0.219
FB15k237-SC, eps=0.5	0.309	0.47	0.339	0.226
	0.31	0.483	0.35	0.22
FB15k237-SC, eps=0.7	0.307	0.465	0.349	0.223
	0.345	0.5	0.383	0.265
FB15k237-SC, eps=0.8	0.254	0.377	0.297	0.184
	0.285	0.415	0.306	0.218
FB15k237-HC	0.249	0.41	0.273	0.168
	0.287	0.454	0.31	0.206

Table 5.6: Link prediction results for the ambiguous relations in the FB15k-237 datasets

Dataset	MRR	Hits@10	Hits@3	Hits@1
WN18	0.72	0.893	0.788	0.6248
WN18-SC, eps=0.5	0.74	0.9	0.8	0.6505
WN18-SC, eps=0.8	0.75	0.901	0.816	0.662
WN18-HC	0.78	0.9	0.837	0.706

Table 5.7: Link prediction results for the WN18 datasets

Dataset	MRR	Hits@10	Hits@3	Hits@1
WN18-SC, eps=0.5	0.44	1	0.66	0.16
	0.565	0.66	0.5	0.5
WN18-SC, eps=0.8	0.382	0.576	0.403	0.307
	0.421	0.5	0.442	0.365
WN18-HC	0.51	0.74	0.549	0.406
	0.564	0.771	0.613	0.464

Table 5.8: Link prediction results for the ambiguous relations in the WN18 datasets

Chapter 6

Conclusion

This thesis introduced a framework for relation disambiguation in Knowledge Graphs. A main component of the presented framework is the Relation Disambiguation algorithm which performs relation disambiguation based on relation-specific embeddings that are computed on a basis of RESCAL's tensor factorization for the input Knowledge Graph. The achieved results on different datasets showed that the Relation Disambiguation algorithm ennhances the Knowledge Graph semantics which leads to improving the link prediction results of the RESCAL method.

The relation disambiguation framework currently supports only a usage of bilinear tensor factorization methods like RESCAL. In order to compute relation-specific embeddings that are able to capture non-linear relationships between the entities in the Knowledge Graph, as part of the future work, we plan to extend our framework so that it can support computation of other tensor factorization and Knowledge Graph embeddings methods.

List of Figures

List of Tables

Bibliography

[1] Building The LinkedIn Knowledge Graph, howpublished = https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph, note = Accessed: 2019-11-13.

[2] Pytorch biggraph Evaluation, howpublished = https://torchbiggraph.readthedocs.io/en/latest/evaluation.html, note = Accessed: 2019-11-14.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, USA, 2013. Curran Associates Inc.

[4] Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. Unsupervised relation disambiguation with order identification capabilities. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 568–575, Sydney, Australia, July 2006. Association for Computational Linguistics.

[5] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, New York, NY, USA, 2014. ACM.

[6] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference

paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[8] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[9] Fionn Murtagh and Pedro Contreras. Methods of hierarchical clustering. *CoRR*, abs/1105.0121, 2011.

[10] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

[11] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs, 2015. cite arxiv:1503.00759Comment: To appear in Proceedings of the IEEE.

[12] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961. AAAI Press, 2016.

[13] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, USA, 2011. Omnipress.

[14] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, November 1987.

[17] Sebastian Ruder. An overview of gradient descent optimization algorithms., 2016. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam.

[18] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 926–934. Curran Associates, Inc., 2013.

[19] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[20] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[21] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv e-prints*, page arXiv:1907.10121, Jul 2019.

[22] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119. AAAI Press, 2014.