Universität Münster

RDF-driven Entity Clustering of Unstructured Data

Bachelor Thesis

submitted by:
Nikolay Krasimirov Kazanliev

Matriculation number: 506179

Study programme: Computer Science

First reviewer:
JProf. Dr. Tanya Braun

Second reviewer:
Prof. Dr. Jan Vahrenhold

Münster, June 17, 2024

Abstract

Building groups of entities from unstructured data sources is beneficial for knowledge discovery and various other Natural Language Processing (NLP) applications. Determining which entities are semantically closer is a key research area and solving this challenge is an intermediate step to forming semantic entity clusters. In our work, we utilize an RDF (Resource Description Framework) model to represent entities and their properties found in unstructured text, aiming to create a foundation for well-defined similarity measurement. RDF triples illustrate the semantic relationship between two entities in the form of subject, predicate and object.

We combine NLP methods for direct triple extraction with querying the open knowledge base DBpedia to create an RDF representation of the data. This representation is then used as input to an RDF hierarchical clustering algorithm to determine groups of similar entities.

Our resulting pipeline has been evaluated to create a significant number of semantically reasonable clusters from natural language texts, primarily Wikipedia articles that contain many prominent entities. We assess how the resulting cluster sets change with different hyperparameters, such as various similarity measures and hierarchical linkage methods, and present an optimal hyperparameter subset. We also identify a notable correspondence between human judgment and the Silhouette coefficient evaluation metric, indicating that it can be used to unsupervisedly determine optimal cluster assignments.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

In nearly all data sources, we come across both physical entities like people and locations as well as abstract ones, such as concepts and ideologies. Organizing such entities into groups based on specific criteria can uncover new information about the entities themselves or the source data as a whole. Studies from different data science subdomains address entity clustering as a step towards solving problems, including explorative search and knowledge discovery (Gad-Elrab et al., 2020), topic exploration and recognizing latent connections among entities (Elbattah et al., 2017), organizing web search query results (Lee et al., 2013), and generating structural summaries over Linked Data sources (Christodoulou et al., 2015).

Another useful abstraction generated from unstructured data is a summarizing knowledge graph. In recent years, a large amount of data has been stored on the web in this format, following the standardization principles proposed by Berners-Lee (2006). Such graph knowledge bases use the Resource Description Framework (RDF), connecting web resources through relational triples (subject, predicate, object), as defined by Klyne and Carroll (2004). The aim is to create a "world-wide lingua franca" for automated web information processing.

Extracting RDF information from unstructured data can serve as an intermediary step of abstraction, which we later utilize for generating entity clusters. Additionally, querying open-source databases, such as DBpedia (Auer et al., 2007), can be used to complete the extracted data with facts that are not present in the original source but hold true.

1.1 Related Work

We categorize our references for topic-related studies into two groups based on the source type of the entities to be clustered. Text-based approaches address clustering entities found in text corpora, while graph-based methods use knowledge graphs as input and utilize their structural qualities to build semantic clusters. For the purpose of disambiguation, we address the Named Entity Recognition (NER) task in the last subsection to discuss its similarities and differences to our approach.

1.1.1 Text-based Approaches

Alsudais and Tchalian (2019) utilize an approach based on word embedding of named entities to identify entity clusters of people and organizations in text corpora. The evaluation relies on a human judge for labeling and measuring the accuracy of the found groups. The authors point out a significant difference in the cluster abstraction level ("Company executives involved in financial or investment scandal" vs. "Universities").

Lee et al. (2013) introduce a hybrid algorithm aiming to improve entity-level query result clustering in search engines. The study combines different similarity measures, including cooccurrence in a set of documents and entity type extraction from an ontology database.

1.1.2 Graph-based Approaches

Elbattah et al. (2017) employ the Louvain algorithm for community detection on a large graph knowledge base, operating on the premise that communities in knowledge graphs represent clusters of structurally similar entities. The study focuses on four categories of entities, and the implementation relies solely on "is-a" relations.

Gad-Elrab et al. (2020) suggest a clustering method that utilizes knowledge graph embeddings to generate clusters and rule learning to infer explanations that are "(approximately) mutually exclusive." Examples of rule-based labels derived from the clustering results would translate to comprehensible labels such as "created by a Grammy winner" and "is a novel."

Christodoulou et al. (2015) and Eddamiri et al. (2019) employ a hierarchical clustering algorithm on RDF datasets in order to organize entities into classes and extract a structural summary. The work of Eddamiri et al. (2019) builds up on Christodoulou et al. (2015) and utilizes more variations of the proposed algorithm in order to find an optimal combination of a similarity measure and a hierarchical linkage method.
In our research, we apply an adaptation of a graph-based approach for use on text corpora, utilizing the hierarchical clustering algorithm used in Christodoulou et al. (2015) and Eddamiri et al. (2019).

1.1.3 Named Entity Recognition

In order to situate our approach more specifically in the Natural Language Processing (NLP) domain and avoid misunderstandings, we want to address a related but different problem in NLP, Named Entity Recognition (NER). The CoNLL-2003 shared task (Sang and Meulder, 2003) proposes an early definition of the term "named entity" as "phrases that contain the names of persons, organizations, and locations." Later advancements in NER can be applied to various topic-specific corpora, e.g., labeling entities that represent genes, proteins, chemicals, and diseases in biomedical data (Yoon et al., 2019). The task of finding such entities and labeling them differs from our approach in two key aspects:

1. NER operates on a predefined set of labeling categories, while our approach does not put a boundary on the number of generated groups.

2. Our approach is adaptive regarding the input source. Depending on the domain scope, different entities can be clustered together or separated, while a NER method would map them consistently to specific predefined categories.

1.2 Research Problem and Objectives

To build groups of entities from a given unstructured data source, we aim to create a comprehensive pipeline that uses RDF data as an intermediate step between entity extraction and clustering. We adopt the RDF clustering algorithm proposed by Christodoulou et al. (2015) and Eddamiri et al. (2019). This algorithm requires measures of similarity between different entities as well as a measure of distance between clusters (linkage method) to define the order of hierarchical clustering. In this context, the specific objectives of our work are:

- Defining a theoretical model: Establishing similarity measures for comparing entities based on RDF triples, selecting various linkage methods for hierarchical clustering, and determining a clustering evaluation score to identify the best set of clusters for a given similarity measure and linkage method hyperparameters.

- Building an end-to-end pipeline: Creating a pipeline that uses RDF data to bridge the gap between entity extraction and clustering, employing NLP methods for text-specific RDF extraction and querying a knowledge base for additional RDF triples. An initial overview of the pipeline is depicted in Figure 1.1.

- Selecting evaluation criteria: Choosing appropriate metrics for assessing the produced clusters and evaluating the implementation on a given dataset to determine optimal combinations of hyperparameters for the clustering algorithm.

- Identifying further limitations: Highlighting additional limitations of the pipeline revealed during evaluation and finding the pipeline components responsible for them.

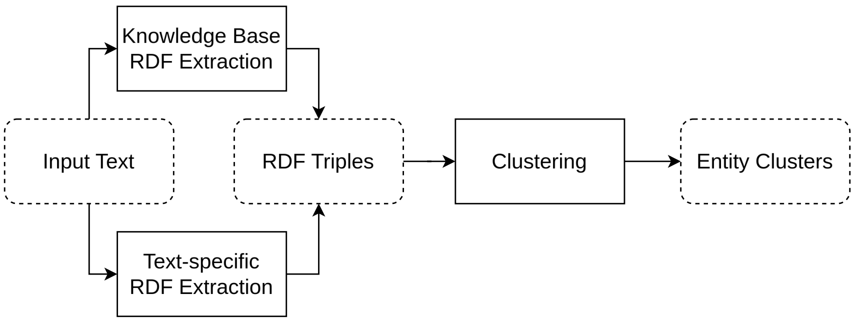

Figure 1.1: Initial Pipeline Structure.

1.3 Thesis Structure

The structure of our thesis corresponds with the research objectives outlined in Section 1.2. In Chapter 2, we give central definitions that provide a foundation for Chapter 3, which introduces the clustering algorithm and defines similarity measures, linkage methods, and an evaluation score used as hyperparameters. In Chapter 4, we refine the initial structure of our pipeline (Figure 1.1) and detail the implementation of different components. In our evaluation (Chapter 5), we present the selected evaluation criteria and dataset assessment. We further identify an optimal subset of hyperparameters for the algorithm and conclude that our pipeline is able to generate semantically reasonable clusters. Finally, in Chapter 6, we address further limitations of the pipeline and offer potential improvements in our conclusion (Chapter 7).

2 Preliminaries

In this chapter, we define concepts that are central to understanding the theoretical foundation of our approach. The following definitions are intended to specify the meaning of commonly used yet potentially ambiguous terms within the context of this work and are adapted to ensure consistency of notation.

In the following, we assume that we have a predefined set of unique identifiers I, which contains the canonical names of entities, and a set P of possible relations between them.

Definition 1 (Entity)**.** *An* entity *is a distinguishable object or concept that can be mapped to a unique identifier $i \in I$.*

Consider the sentence "LA is located on the West Coast." Here, we want to denote *LA* as the city *Los Angeles* and *West Coast* as the *American West Coast*. We assign these entities to their corresponding identifiers to facilitate further operations with them. For example, *LA* can be mapped to `Los_Angeles` $\in I$ and *West Coast* to `West_Coast_of_the_United_States` $\in I$.

Apart from identifying the two entities, we can also infer a relation between them: one is a geographical part of the other. Such a relation can also be mapped to a unique identifier, e.g., `part_of` $\in P$. We can now represent this semantic relation between the entities as an *RDF Triple*. We adapt the definition proposed by Arenas et al. (2009):

Definition 2 (RDF triple)**.**
An RDF triple *is a tuple $(subject, predicate, object) \in I \times P \times (I \cup \{\mathtt{b}\})$.*

For instance, in the triples (`Barack_Obama`, `leader_of_political_party`, `b`), (`Los_Angeles`, `part_of`, `b`) blank nodes represent missing or irrelevant objects. We note that our definition differs from the W3C Recommendation (Cyganiak et al., 2014). In our model, we utilize blank nodes as placeholders only for objects, not for subjects, since we aim to build clusters of subjects that have specific identifiers $i \in I$. Also, we only use a single blank node `b` as a universal placeholder rather than a set $B = \{\mathtt{b0}, \mathtt{b1}, \dots\}$, since we do not need to differentiate between missing objects (it is not important if two blank nodes point to the same entity or not). Furthermore, we do not make a distinction between literals and URIs, as in the W3C definition.

RDF triples extracted from a given data source form an RDF graph. Following Cyganiak et al. (2014), we define:

Definition 3 (RDF graph)**.** *An RDF graph G is a set of RDF triples.*

Example 1. *Consider the following graph: G_0:*

$$\begin{aligned} G_0 = \{&(\texttt{Barack\_Obama}, \texttt{leader\_of\_political\_party}, \texttt{b}),\\ &(\texttt{Mahatma\_Gandhi}, \texttt{leader\_of\_political\_party}, \texttt{b}),\\ &(\texttt{Los\_Angeles}, \texttt{part\_of}, \texttt{b}),\\ &(\texttt{Los\_Angeles}, \texttt{postal\_code}, \texttt{b}),\\ &(\texttt{Berlin}, \texttt{capital\_of}, \texttt{Germany}),\\ &(\texttt{Berlin}, \texttt{postal\_code}, \texttt{b})\} \end{aligned}$$

Now that we have an RDF graph G, presumably extracted from an unstructured text, we also aim to further set up the foundation for clustering by inferring a description of each entity. Therefore, we adopt the term *candidate description* (Christodoulou et al., 2015), and define it as follows:

Definition 4 (Candidate description)**.** *Given an entity $i \in I$ and an RDF graph G, a* candidate description (CD) *of the entity i is the set*

$$CD = \{p \mid \exists o \in I \cup \{b\} : (i, p, o \in G)\}$$

Example 2. *Building up on our previous example, we can identify that G_0 corresponds to $\mathcal{CD}_0 = \{CD_1, CD_2, CD_3, CD_4\}$ with:*

$$\begin{aligned} CD_1 &= \{\texttt{leader\_of\_political\_party}\}\\ CD_2 &= \{\texttt{leader\_of\_political\_party}\}\\ CD_3 &= \{\texttt{part\_of}, \texttt{postal\_code}\}\\ CD_4 &= \{\texttt{capital\_of}, \texttt{postal\_code}\} \end{aligned}$$

For the rest of our work, we assume a one-to-one correspondence between the set of entities (subjects) contained in an RDF graph G and the set of candidate descriptions $\mathcal{CD}$. This would allow us to distinguish between equal sets, such as CD_1 and CD_2, and map them back to their corresponding subjects.

We also note that we include only subjects of RDF triples in the entity set considered for clustering. For instance, the entity `Germany` in G_0 would not be part of the clustering process, since it appears only as an object and its CD is therefore an empty set.

3 Method

As noted earlier in Section 1.1.2, we base our theoretical approach mainly on the RDF clustering algorithm, presented by Christodoulou et al. (2015) and Eddamiri et al. (2019). In this chapter, we assume that we have extracted an RDF graph G, containing subjects $E \subseteq I$, from an unstructured data source, as described in the Preliminaries Section 2.

3.1 Entity Clustering Algorithm

In the following, we present the adapted entity clustering algorithm, building up on the notation used by Christodoulou et al. (2015):

Algorithm 3.1 Entity Clustering on an RDF Graph.

```
Require: RDF graph $G$, $cd\text{-}sim$ (similarity measure), $linkage$ (linkage
    method), $cluster\text{-}score$ (clustering evaluation score)
 1: Extract set of candidate descriptions $\mathcal{CD} = \{CD_1, \ldots, CD_{|\mathcal{CD}|}\}$ from $G$
 2: $m \leftarrow 0$
 3: $U^m \leftarrow \{\{CD_1\}, \{CD_2\}, \ldots, \{CD_{|\mathcal{CD}|}\}\}$
 4: Build similarity matrix $M^m = |\mathcal{CD}| \times |\mathcal{CD}|$:
 5:         $M^m_{ij} = cd\text{-}sim(CD_i, CD_j)$
 6: Convert the similarity matrix to a distance matrix:
 7:         $M^m_{ij} = 1 - M^m_{ij}$
 8: while $m \leq |\mathcal{CD}| - 1$ do
 9:     Let $(U^m_i, U^m_j)$ be the most similar pair in $U^m$ for $i \neq j$:
10:             $\underset{(U^m_i, U^m_j) \in U^m}{\operatorname{argmin}} \; M^m_{ij}$
11:     $m \leftarrow m + 1$
12:     $U^m_l \leftarrow U^{m-1}_i \cup U^{m-1}_j$
13:     Update distance matrix:
14:             $M^m_{lk} = linkage(U^{m-1}_i \cup U^{m-1}_j, U^{m-1}_k)$ for all $k \neq i, j$
15:     $U^m \leftarrow U^{m-1} \setminus \{U^{m-1}_i, U^{m-1}_j\} \cup U^m_l$
16:     $C \leftarrow C \cup U^m$
17: Let $U^i$ be the clustering with the best score:
18:         $\underset{U^i \in C}{\operatorname{argmax}} \; cluster\text{-}score(U^i)$
19: Map each $CD$ in $U^i$ to its unique identifier $i \in E \subseteq I$
20: return mapped $U^i$
```

The algorithm requires an RDF graph G as input, along with three functions serving as hyperparameters: a similarity measure *cd-sim* to compute distances between different candidate descriptions (Section 3.2), a linkage method for hierarchical clustering *linkage* to compute distances between clusters (Section 3.3), and a clustering evaluation score *cluster-score* (Section 3.4) to rate all clusterings (sets of clusters) produced by the algorithm and select the one with the highest rating. Each combination of these hyperparameters, when applied to the same RDF graph G, will typically produce a different clustering result.

In the first step of the execution, we extract a set of candidate descriptions from an RDF graph G (Line 1). Then, we initialize a clustering $U^m = U^0$, where each candidate description (CD) lies in its own singleton cluster (Line 2). The pairwise similarities between these clusters are equal to the corresponding similarities of the candidate descriptions, computed using the given similarity measure *cd-sim*. The obtained values are stored in a similarity matrix M^0 (Lines 4-5). Before constructing larger clusters, we convert the similarity matrix into a distance matrix because hierarchical linkage methods operate on dissimilarity rather than similarity (Lines 6-7).

With the distance matrix computed, we identify the two most similar clusters U_i^m and U_j^m by finding the minimal non-diagonal value of M^m(Lines 9-10). We then merge the selected pair of clusters (Line 12) and update our distance matrix based on the old distances from the two merged clusters to all other clusters U^m, using the selected *linkage* (Lines 13-14). The newly formed cluster U_l^m is added to the updated set of clusters and the old subclusters U_i^{m-1} and U_j^{m-1} are removed (Line 15). Subsequently, the updated clustering is added to the set of obtained clusterings C (Line 16).

We determine the optimal set of clusters U^i by utilizing a given clustering evaluation score *cluster-score* (Lines 17-18) and map the best-rated clustering of candidate descriptions U^i back to a set of entity clusters (Line 19).

Note that if the selected cluster evaluation metric assigns lower values to better-defined clusters, then the best score would correspond to finding $\underset{U^i \in C}{\text{argmin}}$ rather than $\underset{U^i \in C}{\text{argmax}}$ (Line 18).

3.2 Similarity Measures

To cluster similar candidate descriptions, we utilize predefined measures to calculate similarities between each pair of CDs. The used set similarity measures are bounded within the interval [0, 1], with a value of 1 meaning that the two CDs are equal sets of predicates and 0 denoting that the pair has no common elements. The following metrics are commonly applied for preparing non-continuous data for clustering (Eddamiri et al., 2019; Finch, 2021):

3.2.1 Jaccard Similarity

The *Jaccard similarity* of two candidate descriptions CD_i and CD_j is equal to the cardinality of their intersection divided by the cardinality of their union:

$$\textit{cd-sim}(CD_i, CD_j) = \textit{Jaccard}(CD_i, CD_j) = \frac{|CD_i \cap CD_j|}{|CD_i \cup CD_j|} \in [0,1] \tag{3.1}$$

Considering our example for a set of CDs (Example 2), we can calculate:

$$\textit{Jaccard}(CD_3, CD_4) = \frac{|\{\texttt{postal\_code}\}|}{|\{\texttt{part\_of},\ \texttt{postal\_code},\ \texttt{capital\_of}\}|} = \frac{1}{3}$$

3.2.2 Sorensen Similarity

Similarly to the Jaccard measure, *Sorensen similarity* can be used on a set of candidate descriptions. Given CD_i and CD_j, we define:

$$\textit{cd-sim}(CD_i, CD_j) = \textit{Sorensen}(CD_i, CD_j) = \frac{2|CD_i \cap CD_j|}{|CD_i| + |CD_j|} \in [0,1] \tag{3.2}$$

3.2.3 Cosine Similarity

In contrast to Jaccard and Sorensen similarity measures, *cosine similarity* cannot be directly applied to a set of candidate descriptions. Each CD has to be mapped to a vector representing its predicates. To demonstrate this, we use Example 2 and define a list of all predicates for the given CDs, a 4-tuple:

$$(\texttt{leader\_of\_political\_party},\ \texttt{part\_of},\ \texttt{capital\_of},\ \texttt{postal\_code})$$

We can now compute a vector for each candidate description. For instance, $CD_3 = (0,1,0,1)$ and $CD_4 = (0,0,1,1)$.

Having these prerequisites, we can define cosine similarity between the vectors corresponding to CD_i and CD_j as their dot product divided by the product of their Euclidean norms:

$$\textit{cd-sim}(CD_i, CD_j) = \textit{cosine}(CD_i, CD_j) = \frac{CD_i \cdot CD_j}{\|CD_i\|\|CD_j\|} \in [0,1] \tag{3.3}$$

We can easily calculate that $\textit{cosine}(CD_3, CD_4) = \frac{1}{2}$, which differs from $\textit{Jaccard}(CD_3, CD_4) = \frac{1}{3}$.

3.3 Linkage Methods

Similarly to Eddamiri et al. (2019), we employ seven different linkage methods for hierarchical clustering. Our implementation utilizes the SciPy (Virtanen et al., 2020) package `scipy.cluster.hierarchy.linkage`, which uses matrix update formulas to compute the results for each linkage method. We included centroid, median and Ward linkage methods in our study to evaluate whether they yield meaningful results, although they are well-defined only for Euclidean distances. To ensure consistency, we rely on the update formulas and cluster dissimilarities as presented in the work Müllner (2011), which also serves as a foundation of the SciPy implementation.

In the following, we assume to be in iteration m of Algorithm 3.1, immediately after U_i^{m-1} and U_j^{m-1} have been merged. We compute the distances from the newly formed cluster $U_i^{m-1} \cup U_j^{m-1}$ to all other clusters U_k^{m-1}. To simplify our notation, we set $\mathit{linkage}(i \cup j, k) := \mathit{linkage}(U_i^{m-1} \cup U_j^{m-1}, U_k^{m-1})$. M_{ij} represents the distance between clusters i and j in the similarity matrix M^m, and n_i, n_j, and n_k denote the sizes of clusters i, j, and k, respectively.

3.3.1 Linkage Methods for Predefined Distances

The following linkage methods are well-defined for both Euclidean distances and precomputed pairwise distances, such as $1 - \mathit{cd\text{-}sim}(CD_i, CD_j)$, where *cd-sim* represents Jaccard, Sorensen, or cosine similarity:

Average Linkage

For a given cluster k, *average linkage* computes the average of the distances from the merged clusters i and j to k:

$$\mathit{linkage}(i \cup j, k) = \frac{n_i M_{ik} + n_j M_{jk}}{n_i + n_j} \tag{3.4}$$

Complete Linkage

Complete linkage calculates the new distance to cluster k by taking the maximum of the distances from i to k and j to k:

$$\mathit{linkage}(i \cup j, k) = \max(M_{ik}, M_{jk}) \tag{3.5}$$

Single Linkage

In contrast to complete linkage, *single linkage* uses the minimum of the distances between each of the merged clusters and k:

$$\mathit{linkage}(i \cup j, k) = \min(M_{ik}, M_{jk}) \tag{3.6}$$

Weighted Linkage

Weighted linkage takes the average of the old distances from clusters i and j to k:

$$linkage(i \cup j, k) = 0.5(M_{ik} + M_{jk}) \tag{3.7}$$

3.3.2 Linkage Methods for Euclidean Distances

We present the linkage methods in this section using their cluster dissimilarity formulas $cluster\text{-}dist(i, j)$ for clusters U_i^m and U_j^m. Despite this presumption of Euclidean distances between CDs being false, the matrix update formulas, presented and proved in the work of Kaufman and Rousseeuw (1990), allow us to utilize these linkage methods using another precomputed distance, e.g., Jaccard similarity. The following definitions are not well-founded because CDs are sets of identifiers $p \in P$ and not Euclidean points. However, we prefer to present only the cluster dissimilarity definitions for each method, as they are more comprehensible and represent the intuition behind the method more clearly. For the update formulas refer to Appendix A.1.

Centroid Linkage

Centroid linkage considers the Euclidean distance between the centroids of the clusters:

$$cluster\text{-}dist(i, j) = ||c_i - c_j|| \tag{3.8}$$

When clusters i and j are combined, the new centroid is calculated based on all elements.

Median Linkage

Median linkage uses an iteratively defined point w, initially set equal to the data point itself for a singleton cluster. When clusters i and j merge to form cluster l, we compute $w_l = \frac{1}{2}(w_i + w_j)$ (Müllner, 2011). The cluster dissimilarity between clusters i and j is defined as:

$$cluster\text{-}dist(i, j) = ||w_i - w_j|| \tag{3.9}$$

Ward Linkage

Similarly to centroid linkage, *Ward linkage* uses the Euclidean distances of the centroids for measuring distance but also adds a specific factor:

$$cluster\text{-}dist(i, j) = \sqrt{\frac{2n_i n_j}{n_i + n_j}} ||c_i - c_j|| \tag{3.10}$$

Kaufman and Rousseeuw (1990) prove that minimizing this distance measure for choosing the next clusters to be merged is equivalent to minimizing the total error sum of squares (ESS) for the new cluster set. The ESS of a cluster is defined as the sum of squared Euclidean distances from each cluster element to the centroid.

3.4 Clustering Evaluation Score

After Algorithm 3.1 has produced a set of clusterings $C = \{U^0, \dots, U^{|\mathcal{CD}|-1}\}$ (Line 16), we aim to select U^i that would represent an optimal grouping of the elements, based on the distances computed by the selected similarity measure (Section 3.2). Although we use a single evaluation score in the scope of our work, we implement *cluster-score* as a hyperparameter, since our model can be easily extended to work with other metrics for determining an optimal clustering.

Christodoulou et al. (2015) use the *Silhouette coefficient* (SC) to empirically determine a threshold for maximal cluster dissimilarity which terminates the algorithm if exceeded. In contrast to datasets used in their study, our input data for the clustering algorithm can be separated into a potentially large number of categories, and the candidate descriptions are generally less similar to each other. Therefore, we did not elaborate on finding such a threshold. Instead, we apply Silhouette coefficient on each clustering U^i and choose the highest-rated one as the final result of the algorithm. In the following, we adopt the definition given by Kaufman and Rousseeuw (1990).

Given a clustering $U^l \in C$ and a candidate description $CD_i \in U_k \in U^l$, we define the *silhouette* of a point as:

$$Sil(CD_i) = \frac{b(CD_i) - a(CD_i)}{\max\{a(CD_i), b(CD_i)\}} \in [-1, 1] \tag{3.11}$$

In this definition $a(CD_i)$ denotes the average dissimilarity of CD_i to each other element in its assigned cluster U_k:

$$a(CD_i) = d(CD_i, U_k) \coloneqq \sum_{CD_j \in U_k} \frac{1 - \textit{cd-sim}(CD_i, CD_j)}{|U^k|} \in [0, 1] \tag{3.12}$$

Using the distance measure d introduced in Equation 3.12, we define $b(CD_i)$ as the average dissimilarity between CD_i and every element of its closest cluster: $b(CD_i) = \min_{U_j \in C, U_j \neq U_k} d(CD_i, U_j)$. If CD_i is the only element in its cluster, we set $Sil(CD_i) = 0$.

With these prerequisites, the *Silhouette coefficient* for a clustering $U^l \in C$ and a set of candidate descriptions $\mathcal{CD}$ with $n = |\mathcal{CD}|$ elements, also referred to as the *average silhouette width* (ASW), is defined as:

$$\textit{cluster-score}(U^l) = ASW(U^l) = \sum_{i=1}^{n} \frac{Sil(CD_i)}{n} \tag{3.13}$$

4 Implementation

Our implementation can be considered a hybrid approach, combining the extraction of RDF triples using Natural Language Processing libraries with extraction from an open database for all entities found in the text (also using NLP). Once the RDF data is obtained, clustering is performed as detailed in Section 3. We also tried to implement a method for labeling each cluster. Although this labeling approach was not successful, we kept it in the implementation because it can be useful for illustrative purposes.

The main components of our pipeline are shown in Figure 4.1. In the following sections, we will provide more detail about key implementation aspects, including the two aforementioned types of RDF extraction, clustering, and labeling.

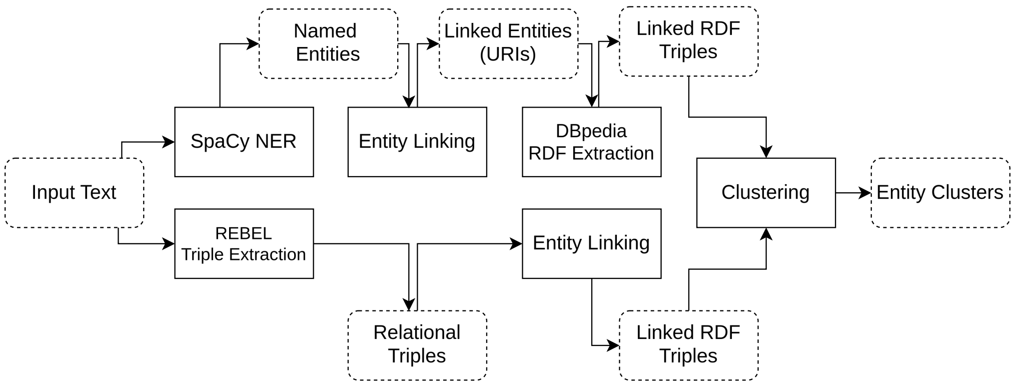

Figure 4.1: Pipeline Overview.

4.1 RDF Extraction using NLP (REBEL)

Extracting RDF triples from unstructured text is a component that is further subdivided into two tasks in our pipeline: relation extraction and entity linking. *Relation extraction* involves identifying semantically related entities in the text and their relationships, represented as relational triples. *Entity linking* refers to linking the triple subjects to a DBpedia URI. For relation extraction, we use the REBEL model (Cabot and Navigli, 2021), and for linking the subjects, we utilize the API provided by DBpedia Spotlight (Mendes et al., 2011). To demonstrate how this two-step approach works, consider the following sentence taken from the Wikipedia article for Germany[1]:

[1]`https://en.wikipedia.org/wiki/Germany`

The Academy Award for Best Foreign Language Film ("Oscar") went to the German production The Tin Drum (Die Blechtrommel) in 1979.

REBEL recognizes the following relational triple:

```
<The Tin Drum> <award received>
               <Academy Award for Best Foreign Language Film> .
```

DBpedia Spotlight would map the subject of the relation to a DBpedia URI:

```
<http://dbpedia.org/resource/The_Tin_Drum> <award received>
               <Academy Award for Best Foreign Language Film> .
```

Entity linking is needed for two reasons: first, we want to disambiguate subjects, e.g., `Barack Hussein Obama` and `Barack Obama` would be linked to `http://dbpedia.org/resource/Barack_Obama`. Entity linking allow us to later combine the triples with those extracted from DBpedia (Section 4.2).

We do not link predicates to URLs because REBEL uses a list of predefined unique predicates[2]. It is important to note that these conditions for normalizing triples conform to our definition of an RDF triple (Definition 2), but not the official definition given by the W3C Recommendation (Cyganiak et al., 2014).

4.2 RDF Extraction from DBpedia

As depicted in Figure 4.1, extracting DBpedia RDF triples is performed in three steps: First, we use the pretrained pipeline `en_core_web_sm` provided by spaCy (Honnibal et al., 2020) for recognizing various types of named entities in the input text, including `PERSON`, `LOC`, `EVENT`, `WORK_OF_ART`, and others[3]. After entities have been identified, we use the DBpedia Spotlight API to link them to DBpedia URIs. Lastly, for each linked entity, we extract all RDF triples from DBpedia in which it is a subject in the relationship.

Referring to our previous example in Section 4.1, spaCy also recognizes the entity `The Tin Drum`, DBpedia Spotlight links it to the same URI, and after extraction, we have all corresponding triples from DBpedia. As discussed further in Chapter 6, our pipeline is prone to error propagation, and the given example also demonstrates this issue. The found entity `The Tin Drum` is linked to the DBpedia resource for the book[4], rather than its film adaptation[5]. To illustrate how this component of the pipeline works, we consider the following triple subset:

[2]For the complete list of relations that REBEL is trained on, visit `https://github.com/Babelscape/rebel/blob/main/data/relations_count.tsv`

[3]`https://spacy.io/models/en#en_core_web_sm` (see Label Scheme/NER)

[4]`https://dbpedia.org/page/The_Tin_Drum`

[5]`https://dbpedia.org/page/The_Tin_Drum_(film)`

```
<http://dbpedia.org/resource/The_Tin_Drum>
    <http://dbpedia.org/property/genre> _:b0 .
<http://dbpedia.org/resource/The_Tin_Drum>
    <http://dbpedia.org/ontology/author> _:b0 .
<http://dbpedia.org/resource/The_Tin_Drum>
    <publication date> <1979> .
```

Instead of saving the objects of the DBpedia triples, we model them as blank nodes. This is because our hierarchical clustering algorithm (Algorithm 3.1) uses only entities and their corresponding candidate descriptions, i.e., subjects of RDF triples and their set of predicates. The last triple in the example is extracted from REBEL and included in the list of all triples.

In our research, we discovered that including the values of two DBpedia property types to the RDF triples, `dcterms:subject`[6] and `rdf:type`[7], can significantly improve the accuracy of our clusterings. The `rdf:type` property indicates the classes to which the subject belongs and the `dcterms:subject` property refers to different topics related to the subject. To integrate these properties into our model and utilize them in the clustering process, we save their values for the corresponding entities as predicates, e.g., for the DBpedia triple (`dbr:The_Tin_Drum`, `rdf:type`, `schema:Book`), we add the following triple to the set for clustering:

```
<http://dbpedia.org/resource/The_Tin_Drum>
    <http://schema.org/Book> _:b0 .
```

Lastly, to further improve accuracy, we filter out triples with predicates that are very common but do not provide significant information about a given entity, such as `http://dbpedia.org/ontology/wikiPageID`.

4.3 Clustering

We implemented our clustering component as detailed in Chapter 3. For a given input text, we extract triples, as discussed in Sections 4.1 and 4.2, and use them to construct an RDF graph G as input for Algorithm 3.1. For each input text, we automatically run the algorithm for all combinations of three similarity measures, seven linkage methods, and one clustering evaluation score, resulting in 21 different clusterings.

4.4 Labeling

For each entity, we extract the value of the `gold:hypernym` property and then select a label for the cluster based on majority voting. In case of a tie, we

[6]`https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#subject`
[7]`https://www.w3.org/TR/rdf-schema/#ch_type`

resolve it by selecting the value that is more commonly assigned across all DBpedia resources.

The `gold:hypernym` property is extracted from the text of Wikipedia articles using lexico-syntactic patterns. If no accurate type that can be linked to a DBpedia Ontology class is found, further methods are applied to determine an appropriate class (Kliegr and Zamazal, 2016). For examples of entities and clusters, labeled based on DBpedia hypernyms, we refer to Appendix C.

We have also attempted to implement a labeling component utilizing the `dcterms:subject` and `rdf:type` property values. The approaches based on identifying a common value among the entities and if no common value was found, we then searched for superclasses of the values to determine the most common ancestor. The method using `dcterms:subject` did not produce labels for larger clusters mostly due to cycles in the hierarchy, as noted by Hulpus et al. (2013), and, similarly, we found cycles in the hierarchy of `rdf:type`. However, the `rdf:type` approach mostly failed because it returned very broad labels for larger clusters, such as `owl:Thing` or `dbo:Person`. Using majority voting instead of traversing the hierarchy for the values of both discussed properties produced less accurate labels than our initial approach using the property `gold:hypernym`.

Although we do not consider the implementation of labeling to be successful and have not evaluated its overall accuracy as part of our work, we decided to keep it, since it can be useful as a visual aid when interpreting the clustering results.

5 Evaluation

Although we utilize the clustering methods proposed by Eddamiri et al. (2019) and Christodoulou et al. (2015), our input data and, consequently, our performance evaluation metrics differ. Both studies assess their methods on RDF datasets with a number of predefined classes, allowing for ground truth performance evaluation. In contrast, our pipeline processes natural language text without a priori information about predefined categories. Similarly to the results observed by Alsudais and Tchalian (2019), our resulting clusters vary in terms of broadness, e.g., "politicians" vs. "classical composers of the 19th century."

5.1 Evaluation Metrics

We adopt three performance measures from Alsudais and Tchalian (2019), namely *coherence measure*, *precision measure*, and *coherent clusters measure*, to evaluate the quality of generated clusterings. Furthermore, we apply a weighting factor precision measure to better represent the specifics of our data.

For each combination of hyperparameters in Algorithm 3.1, we perform a manual evaluation of the resulting clusters to determine if the majority of elements fit into an appropriate category. A single cluster can be evaluated as *accurate*, *partly accurate*, or *inaccurate*. If the majority of the cluster elements fit into a specific category, we label the cluster as *accurate* when all elements fit, or *partly accurate* if some of the elements do not match the category. Otherwise, if no specific label can be found, the cluster is evaluated as *inaccurate*. We note that the appropriate specificity of categories would vary depending on the input text. In texts where more specific clusters can be generated, we also consider more concrete categories as a basis for evaluation. We refer to Appendix C for examples of evaluated clusters.

5.1.1 Coherence Measure

The *coherence measure* (CM) of a cluster U_i relies on the number of accurately placed (relevant) entities and is defined as follows:

$$\mathrm{CM}_{cluster}(U_i) = \frac{\text{Number of relevant entities in } U_i}{\text{Number of entities in } U_i} \in [0,1] \qquad (5.1)$$

We set $\mathrm{CM}_{cluster}(U_i) = 0$ for a cluster that is evaluated as inaccurate. Subsequently, we define coherence measure for a clustering U^k, with $|U^k| = n$, as:

$$\mathrm{CM}_{overall}(U^k) = \frac{1}{n} \sum_{j=1}^{n} CM_{cluster}(U_j) \in [0, 1] \tag{5.2}$$

5.1.2 Precision Measure

Similarly to coherence measure, the *precision measure* (PM) calculates the ratio of accurately placed entities to all entities, but it is applied to the entire clustering instead of averaging computations for each cluster:

$$\mathrm{PM}(U^k) = \frac{\sum_{j=1}^{n} \text{Number of relevant entities in } U_j}{\text{Total number of entities in } U^k} \in [0, 1] \tag{5.3}$$

5.1.3 Coherent Clusters Measure

The *coherent clusters measure* (CCM) of a clustering U^k indicates the proportion of clusters that were evaluated as accurate, or partly accurate:

$$\mathrm{CCM}(U^k) = \frac{\text{Number of (partly) accurate clusters in } U^k}{n} \in [0, 1] \tag{5.4}$$

5.1.4 Weighted Precision Measure

As discussed later in Section 5.2, some linkage methods typically produce a large number of one-element clusters. Because this tendency can drastically change the values of the previously introduced performance measures (Sections 5.1.1, 5.1.2, 5.1.3), we decided to exclude single-element clusters when using the aforementioned metrics. To compensate for the higher accuracy of methods that produce many one-element clusters, we include the percentage of evaluated clusters as a weighting factor for precision measure (PM) in an additional measure. We define the *weighted precision measure* (WPM) as:

$$\mathrm{WPM}(U^k) = \mathrm{PM}(U^k) \left(\frac{n - \text{ Number of one-element clusters in } U^k}{n} \right) \tag{5.5}$$

5.2 Dataset: Germany Wikipedia Article

To present the results of our pipeline, we selected the Germany Wikipedia article[1] as an input text. We observed similar tendencies regarding the generated

[1] `https://en.wikipedia.org/wiki/Germany`

results across other inputs (listed in Appendix B.2) and therefore selected this article as a representative example of our findings.

5.2.1 Overall Dataset Information

Before evaluating the generated clusterings, we want to present some overall statistics about the article as well as the extracted entities and triples used as an input for our clustering algorithm (Table 5.1):

Number of parsed sentences	421
Number of parsed words	10097
Total number of entities	333
Found only by REBEL	9
Found only by spaCy	283
Found by both	41
Total number of triples	28528
DBpedia triples	28457
REBEL triples	71

Table 5.1: Summary of Extracted Triples and Entities for the Germany Article.

Table 5.1 shows that the number of triples extracted by the relation extraction model is significantly smaller compared to those extracted from DBpedia for named entities recognized by spaCy. Another notable observation is that the total number of entities is equal to approximately 3% of the text's words. We discuss the reasons for these discrepancies further in Chapter 6.

5.3 Dataset Evaluation

5.3.1 One-element Clusters

As we noted earlier (Section 5.1.4), the percent of one-element clusters is very high for some linkage methods. In Table 5.2 we present the ratio

$$\frac{\text{Number of one-element clusters in } U^k}{|U^k|}$$

for each best-rated clustering U^k, resulting from a combination of specific linkage method and similarity measure hyperparameters in Algorithm 3.1.

Table 5.2: Percent of One-Element Clusters.

	Linkage Method for Clustering						
Similarity Measure	Average	Complete	Single	Weighted	Centroid	Median	Ward
Jaccard	0.47	0.34	0.80	0.44	**0.88**	0.86	**0.00**
Cosine	0.44	0.37	0.73	0.40	0.81	0.82	0.09
Sorensen	0.44	0.34	0.73	0.41	0.80	0.80	0.07

We want to point out that even in combinations with a very high percentage of one-element clusters, a significant proportion of the elements are still clustered in larger groups. For example, Jaccard similarity paired with centroid linkage has a ratio of 0.88 in Table 5.2. While 88% of all clusters consist of only one element, these clusters represent approximately 60% of the total number of entities (203 out of 333) and the rest of the entities are grouped into clusters containing more than one element.

Conversely, another notable observation from Table 5.2 is that Ward linkage tends to produce few to no single-element clusters. The significant difference in the results of centroid and median linkage compared to Ward linkage can be explained as follows: centroid linkage tends to reduce the distances between clusters and non-clustered entities in each iteration, while Ward linkage does the opposite—it increases the distances between clusters containing multiple elements and one-element clusters. We reference to Appendix A.2 for an example of the discussed tendencies. In the case of centroid and median linkage, this behavior often leads to elements being added to an already formed cluster instead of being clustered together with their most similar entities according to the initial similarity matrix. Conversely, Ward linkage's increasing distances between existing clusters leaves space for unclustered entities to create clusters among themselves in later iterations.

The actual difference in the number of one-element clusters is a result of the following: centroid linkage forms clusters early in the algorithm with elements that are not highly similar, leading to the Silhouette coefficient assigning lower scores to clusterings produced by higher iterations of the algorithm. Therefore, an early clustering is chosen, which contains many single-element clusters. On the other hand, the opposite behavior of Ward linkage results in clusterings produced by later iterations being selected by the Silhouette coefficient.

5.3.2 Evaluation Measures: Results

In this section, we present an evaluation of the generated results using our evaluation metrics (CM, PM, CCM, and WPM), as defined in Section 5.1. We want to highlight specific structural differences between clusterings produced by different combinations of similarity measures and linkage method hyperparameters, as indicated by the values of the evaluation metrics. Our objective is to identify combinations that offer the best trade-off between accuracy and reducing one-element clusters, such that the result cluster sets are closer to human intuition, i.e., would be rated higher by a human judge.

Coherence and Precision Measures

Table 5.3: Coherence Measure.

	Linkage method for clustering						
Similarity Measure	Average	Complete	Single	Weighted	Centroid	Median	Ward
Jaccard	0.89	0.87	0.93	0.90	**0.97**	0.91	**0.78**
Cosine	0.88	0.88	0.93	0.90	0.93	0.95	0.80
Sorensen	0.90	0.88	0.94	0.90	0.94	0.95	0.79

Table 5.4: Precision Measure.

	Linkage method for clustering						
Similarity Measure	Average	Complete	Single	Weighted	Centroid	Median	Ward
Jaccard	0.85	0.83	0.90	0.85	**0.94**	0.90	**0.74**
Cosine	0.84	0.84	0.80	0.83	0.86	0.93	0.80
Sorensen	0.85	0.84	0.85	0.83	0.87	0.93	0.78

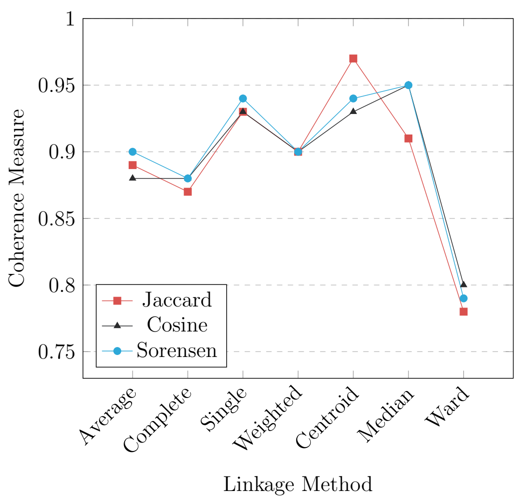

Figure 5.1: Coherence Measure.

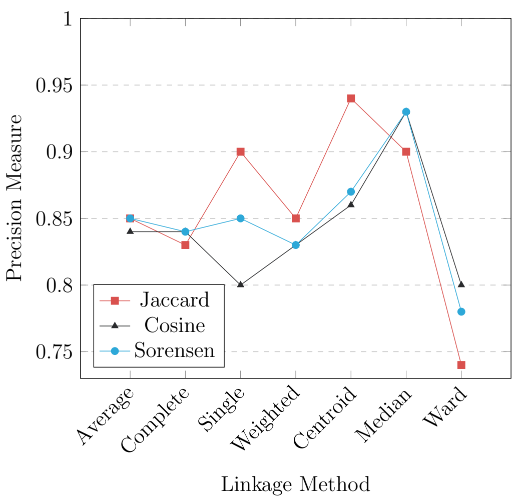

Figure 5.2: Precision Measure.

By jointly examining the results for coherence measure (Table 5.3, Figure 5.1) and precision measure (Table 5.4, Figure 5.2) we can explain some expected tendencies in our data. Clusters with a lower number of entities are much more common than larger clusters in our results across all combinations of linkage methods and similarity measures. Smaller clusters are more likely to be accurate compared to larger ones, which tends to shift the coherence measure results higher on average. Additionally, we observe a notable difference in the precision measure values for Jaccard similarity compared to those for cosine and Sorensen similarity. A possible reason for this is that Jaccard similarity tends to compute higher distances between similar entities (further discussed in Appendix B.1). The opposite direction of the average induced change for centroid and median linkage compared to Ward linkage can be attributed to the contrasting behavior of the methods, as discussed in Appendix A.2.

In our results, a higher number of one-element clusters is closely linked to both higher coherence and precision. As discussed earlier in Section 5.3.1, when using centroid and median linkage, the Silhouette coefficient tends to rate earlier iterations of the clustering algorithm higher, resulting in a clustering that contains many unclustered entities (one-element clusters). Conversely, multi-element clusters formed in earlier iterations consist of the most similar entities in the dataset, making them more accurate. For this reason, centroid and median linkage have high coherence and precision values. The third-best performer in terms of both measures is single linkage, which, by keeping the minimum pairwise distance between elements of different clusters, can also result in larger clusters with highly dissimilar entities on average in later iterations.

Coherence Clusters Measure

The results of the coherence clusters measure (Table 5.5) align closely with our observations for the coherence and precision values. As expected, centroid, median, and single linkage are the best performers. We note that centroid and median linkage produced smaller and more accurate clusters than single linkage. However, even in clusterings that used single linkage, there was no single cluster evaluated where we could not find a suitable category for the majority of the elements, although smaller subsets within clusters could be completely unrelated to the relevant majority. Predictably, Ward linkage performed the worst because it generated some clusters with dissimilar entities that usually remain unclustered when using other methods. The examples listed in Appendix C.3 all result from Ward linkage.

Table 5.5: Coherent Clusters Measure

	Linkage method for clustering						
Similarity Measure	Average	Complete	Single	Weighted	Centroid	Median	Ward
Jaccard	0.95	0.95	**1**	0.98	**1**	**1**	**0.90**
Cosine	0.93	0.95	**1**	0.96	**1**	**1**	**0.88**
Sorensen	0.98	0.97	**1**	0.98	**1**	**1**	**0.88**

Weighted Precision Measure

The best performers in the previous evaluations using PM, CM, and CCM had better results primarily due to the high accuracy of a few generated multiple-element clusters. Rather than achieving extreme accuracy in a few clusters, we aim to find a trade-off between accuracy and having more elements clustered into semantically similar groups. Using the weighted precision measure acts as a counterbalance and constitutes an evaluation which is much closer to our perception of quality clustering.

The evaluation results for WPM (Table 5.6 and Figure 5.3) nearly reverse the rankings of the best and worst performing linkage methods compared to the previous metrics. Single, centroid, and median linkage now rank lower, while the other linkage methods, particularly Ward linkage with a very small number of unclustered elements, achieve higher evaluation scores.

Table 5.6: Weighted Precision Measure.

	Linkage method for clustering						
Similarity Measure	Average	Complete	Single	Weighted	Centroid	Median	Ward
Jaccard	0.45	0.54	0.18	0.48	0.12	0.13	**0.74**
Cosine	0.47	0.53	0.22	0.49	0.16	0.16	**0.73**
Sorensen	0.48	0.55	0.23	0.49	0.17	0.19	**0.72**

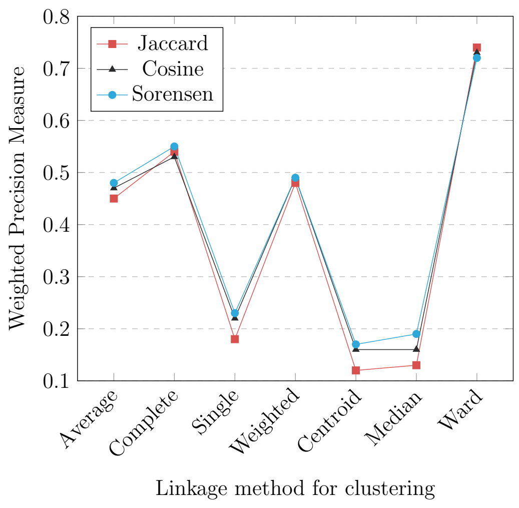

Figure 5.3: Weighted Pr. Measure.

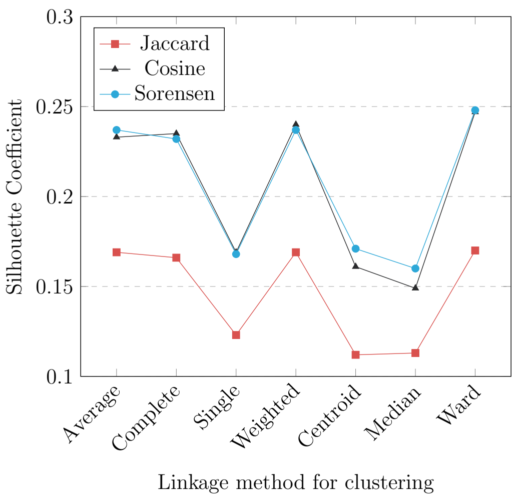

Figure 5.4: Silhouette Coefficient.

We point out the evident similarity between the results of the weighted precision measure (WPM) and the Silhouette coefficient (Figure 5.4). This similarity indicates that WPM is a balanced measure that accurately captures the trade-off between accuracy and the number of clustered elements. Conversely, because WPM aligns with our understanding of which clusterings are more semantically reasonable, we can hypothesize that the Silhouette coefficient can be used to unsupervisedly determine linkage methods and similarity measure combinations that produce meaningful results for a given dataset. We also note that the observed results for the Silhouette coefficient of the best-rated clusterings represent a tendency that is found in multiple other datasets (Appendix B.2) and statistically stabilizes with an increasing number of entities.

The lower values for Jaccard similarity in Figure 5.4 result from the measure computing higher distances between similar elements compared to Sorensen and cosine similarity (Appendix B.1). As demonstrated by our evaluation, this difference does not indicate a lower clustering quality, since Jaccard similarity did not show any significant deviation from other similarity measures.

Conclusion

Using the NLP tools described in our Implementation (Chapter 4), we were able to put together a basis of RDF triples for hierarchical clustering (Algorithm 3.1) and infer a significant proportion of clusters containing semantically similar entities. The key finding from our evaluation is that median, centroid, and single linkage methods achieve high accuracy but leave a significant number of elements unclustered. In contrast, although slightly less accurate, average, complete, weighted, and Ward linkage methods generate semantic reasonable clusters, with most entities being combined into larger clusters. Our results remained consistent across all similarity measures.

6 Discussion

Results from Chapter 5 demonstrate that our implementation effectively captures semantic similarities and produces a significant proportion of clusters that align well with human intuition. However, it is also important to address issues such as the accuracy of text-specific relation extraction and how errors in some pipeline components can affect the final results. In this chapter, we will outline two major limitations of our implementation related to these issues.

Text-specific Relations. The input text may emphasize information that is not available in DBpedia, and even if this information is recognized and recorded as a triple by REBEL, it will have minimal impact on the clustering process due to the much larger number of included DBpedia triples. Consider the following example:

Robert Wiene and Heinrich Brüning were born in the 19th century. Angela Merkel and Werner Herzog were born in the 20th century.

The text emphasizes when the four prominent people were born, and a human judge can easily identify two groups: those born in the 19th century and those born in the 20th century. However, REBEL is not able to recognize this relationship. Generally, semantics that extend beyond the sentence level are harder to capture. Because of the similarities based on DBpedia properties, the most common clustering result is the following:

```
"Chancellor 1": {
    "Heinrich_Brüning": "Chancellor",
    "Angela_Merkel": "Not found"},
"Director 1": {
    "Robert_Wiene": "Director",
    "Werner_Herzog": "Director"}
```

Due to the small context, REBEL was only able to identify one relation, while all the extracted DBpedia triples were 497.

Although the linked relational triples extracted with REBEL have a limited effect on the clustering process, we included a relation extraction module in our pipeline to allow for easy replacement with other models. Based on our experience, the most important quality of a model for generating triples suitable for clustering is having an extensive predefined set of possible triple predicates. We use REBEL, as it is the best model we have found so far in this regard. Fine-tuning a pretrained model for topic-specific corpora is also a potential area for future work, given the list of predicates is sufficiently long and descriptive to serve as disambiguation criteria for different classes of entities.

Error Propagation. REBEL fails to recognize relations correctly when provided with insufficient information, e.g., the linked relation triple generated from the previous example text is:

```
<http://dbpedia.org/resource/Werner_Herzog>
    <spouse> <Angela Merkel> .
```

Although this error would not affect the result in this specific scenario, a large number of such errors can lead to incorrect clustering of entities.

In the following, we present another example of error propagation that shows how an error can be propagated impacting subsequent components of our pipeline:

```
<Bobby Rush> <member of political party> <Democratic> .
<http://dbpedia.org/resource/Rush_(band)>
    <member of political party> <Democratic> .
```

In this example, the entity `Bobby Rush` is incorrectly linked to `Rush_(band)`. Although we have increased the confidence parameter of the DBpedia Spotlight API to 0.9, which is quite strict, such errors can still occur. After the false linking, `Rush_(band)` gets clustered with another semantically distant entity:

```
"Band 1": {
    "entities": {
        "United_States_Armed_Forces": "Forces",
        "Rush_(band)": "Band"}
```

This cluster is an example of how noisy data can lead to clusters containing contextually dissimilar elements. The two entities share predicates that are either unrelated to the entities' meanings and were not filtered out (as explained in Section 4.2), such as `http://dbpedia.org/ontology/wikiPageWikiLink`. Additionally, they share values of `dcterms:subject` or `rdf:type` that are too broad in general but happen to be common to both entities, e.g., `http://www.wikidata.org/entity/Q24229398`, which denotes a "distinct and identifiable entity capable of performing actions."

Lastly, since `Band` is a more common hypernym than `Forces`, it is chosen as the cluster label, completing the propagation from erroneous entity linking to incorrect clustering and labeling.

7 Conclusion

In this chapter, we discuss the most notable insights gathered during the implementation of our end-to-end pipeline and the evaluation of its results.

Our findings indicate that RDF data describing entities serves as an effective intermediate semantic representation and basis for clustering, even when the data is noisy or the average similarity between entities is very low. The formal model we outlined in Chapters 2 and 3 provided a reliable foundation for our implementation (Chapter 4).

As discussed in our evaluation (Chapter 5), we found that the results remained stable across all similarity measures: Jaccard, Sorensen, and cosine similarity. We identified a subset of linkage methods—average, complete, weighted, and Ward linkage—that produce clusterings with a good balance between accuracy and the inclusion of most elements in larger clusters. Additionally, we observed that the Silhouette coefficient selected clusterings that were also likely to receive high evaluations from a human judge.

The most significant limitation of our implementation, as outlined in Chapter 6, is that many text-specific relations can remain unrecognized due to the relatively small number of triples directly extracted from the input text. This leads us to several potential areas for future work to further enhance our pipeline:

- REBEL can be replaced with another NLP relation extraction model with higher accuracy, a long predefined set of predicates, and a high rate of extracting relevant triples from the text.
- For a highly accurate NLP relation extraction model, consider giving higher weight to NLP-extracted triples to emphasize text-specific relations.
- Research incorporating other open knowledge bases, such as YAGO[1] and Wikidata[2].
- Instead of filtering out triples with non-relevant predicates, as discussed in Chapter 4.2, define a set of predicates that are suitable for disambiguation and allow only triples with these predicates.

[1]`https://yago-knowledge.org`
[2]`https://www.wikidata.org`

A Additional Notes on Linkage Methods

A.1 Linkage Methods for Euclidean Distances: Update Formulas

In this appendix, we present the linkage matrix update formulas as implemented in the `scipy.cluster.hierarchy.linkage` package. After generating the initial similarity matrix with one of the discussed similarity measures (Section 3.2), the following update formulas allow us to perform centroid, median, and Ward linkage. These formulas use distances only from the matrix itself and do not require Euclidean coordinates, unlike the definitions from Section 3.3.2. Using the notation and sources from Section 3.3, we define:

A.1.1 Centroid Linkage: Update Formula

$$linkage(i \cup j, k) = \sqrt{\frac{n_i M_{ik}^2 + n_j M_{jk}^2 - n_i n_j M_{ij}^2/(n_i + n_j)}{n_i + n_j}} \tag{A.1}$$

A.1.2 Median Linkage: Update Formula

$$linkage(i \cup j, k) = \sqrt{0.5(M_{ik}^2 + M_{jk}^2) - 0.25 M_{ij}^2} \tag{A.2}$$

A.1.3 Ward Linkage: Update Formula

$$\begin{aligned} a_{ik} &= \frac{n_k + n_i}{n_i + n_j + n_k} \\ a_{jk} &= \frac{n_k + n_j}{n_i + n_j + n_k} \\ a_{ij} &= \frac{n_k}{n_i + n_j + n_k} \\ linkage(i \cup j, k) &= \sqrt{a_{ik} \cdot M_{ik}^2 + a_{jk} \cdot M_{jk}^2 - a_{ij} \cdot M_{ij}^2} \end{aligned} \tag{A.3}$$

A.2 Linkage Methods: Examples

In this section we aim to demonstrate the different behaviors of median and Ward linkage, as discussed in Section 5.3.1, using a subset of our dataset, consisting of the following DBpedia entities (omitting the base URI): `Munich_Airport`, `Ludwig_van_Beethoven`, `Frankfurt_Airport`, `Felix_Mendelssohn`, `Berlin_Brandenburg_Airport`, `Michelin_Guide`, `The_World_Factbook`. For representing these entities in the distance matrix we define shorter keys in the same order: `MUC`, `Beet`, `FRA`, `Mend`, `BER`, `Guide`, `Book`. To illustrate the difference between the linkage methods more effectively, we have increased the distance between `Guide` and `Book` from 0.814 to 0.875.

A.2.1 Median Linkage: Example using Jaccard Similarity

	MUC	Beet	FRA	Mend	BER	Guide	Book
MUC	0	0.969	**0.298**	0.970	0.360	0.950	0.954
Beet	0.969	0	0.971	0.742	0.969	0.974	0.975
FRA	**0.298**	0.971	0	0.972	0.436	0.955	0.958
Mend	0.970	0.742	0.972	0	0.970	0.968	0.970
BER	0.360	0.969	0.436	0.970	0	0.949	0.953
Guide	0.950	0.974	0.955	0.968	0.949	0	0.875
Book	0.954	0.975	0.958	0.970	0.953	0.875	0

Consider the distance matrix (1 - similarity matrix M) for our entity subset.

	FRA,MUC	Beet	Mend	BER	Guide	Book
FRA,MUC	0	0.958	0.960	**0.371**	0.941	0.944
Beet	0.958	0	0.742	0.969	0.974	0.975
Mend	0.960	0.742	0	0.970	0.968	0.970
BER	**0.371**	0.969	0.970	0	0.949	0.953
Guide	0.941	0.974	0.968	0.949	0	0.875
Book	0.944	0.975	0.970	0.953	0.875	0

After the first iteration of the clustering algorithm, we can notice that distances between the formed cluster {`FRA`, `MUC`} and other entities have been reduced. Examining the first row of the updated distance matrix shows that distances from the cluster to any given entity, except for `BER`, are lower than any of the distances from the elements `FRA` and `MUC` themselves to the same entities.

	`FRA,MUC,BER`	`Beet`	`Mend`	`Guide`	`Book`
`FRA,MUC,BER`	0	0.945	0.947	0.927	0.930
`Beet`	0.945	0	**0.742**	0.974	0.975
`Mend`	0.947	**0.742**	0	0.968	0.970
`Guide`	0.927	0.974	0.968	0	0.875
`Book`	0.930	0.975	0.970	0.875	0

	`FRA,MUC,BER`	`Beet, Mend`	`Guide`	`Book`
`FRA,MUC,BER`	0	**0.870**	0.927	0.930
`Beet, Mend`	**0.870**	0	0.897	0.899
`Guide`	0.927	0.897	0	0.875
`Book`	0.930	0.899	0.875	0

	`FRA,MUC,BER,Beet,Mend`	`Guide`	`Book`
`FRA,MUC,BER,Beet,Mend`	0	**0.802**	0.805
`Guide`	**0.802**	0	0.875
`Book`	0.805	0.875	0

	`FRA,MUC,BER,Beet,Mend,Guide`	`Book`
`FRA,MUC,BER,Beet,Mend,Guide`	0	**0.739**
`Book`	**0.739**	0

We observe similar behavior throughout the iterations. In particular, we can see how `Guide` and `Book` do not get clustered together until the last merge, even though the pairwise distance between the two entities is lower than any other distance from `Guide` and `Book` to any other entity. Reducing distances leads to *inversions* in the resulting dendrogram (Figure A.1)

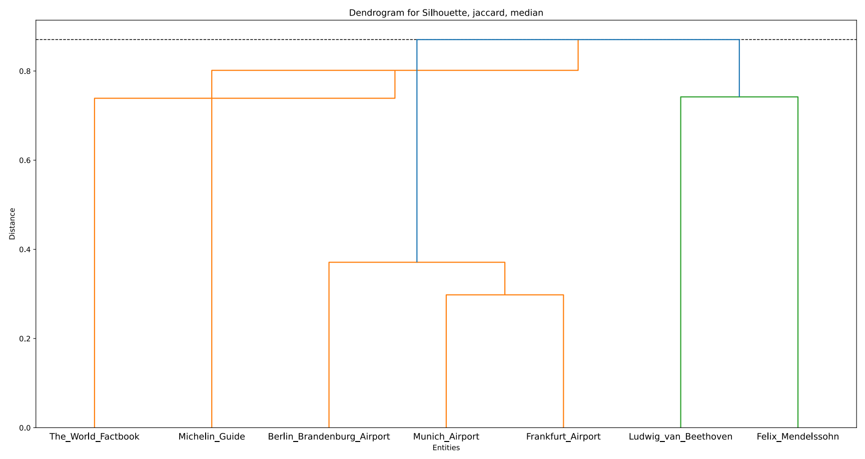

Figure A.1: Median Linkage Example: Dendrogram.

A.2.2 Ward Linkage: Example using Jaccard Similarity

	MUC	Beet	FRA	Mend	BER	Guide	Book
MUC	0	0.969	**0.298**	0.970	0.360	0.950	0.954
Beet	0.969	0	0.971	0.742	0.969	0.974	0.975
FRA	**0.298**	0.971	0	0.972	0.436	0.955	0.958
Mend	0.970	0.742	0.972	0	0.970	0.968	0.970
BER	0.360	0.969	0.436	0.970	0	0.949	0.953
Guide	0.950	0.974	0.955	0.968	0.949	0	0.875
Book	0.954	0.975	0.958	0.970	0.953	0.875	0

Once again, we have the same initial distance matrix, and consequently, the same pair of elements get clustered first: MUC and FRA.

	FRA,MUC	Beet	Mend	BER	Guide	Book
FRA,MUC	0	1.107	1.108	**0.428**	1.090	1.090
Beet	1.107	0	0.742	0.969	0.974	0.975
Mend	1.108	0.742	0	0.970	0.968	0.970
BER	**0.428**	0.969	0.970	0	0.949	0.953
Guide	1.090	0.974	0.968	0.949	0	0.875
Book	1.090	0.975	0.970	0.953	0.875	0

As in the example about median linkage (Section A.2.1), the general tendency to shift distances outside the min-max range of initial distances for the clustered entities is evident after the first iteration. We can again observe the first row to highlight the opposite behavior of Ward linkage compared to median linkage: distances from {`FRA`, `MUC`} to all other elements (except for `BER`) are greater than the initial distances.

	`FRA,MUC,BER`	`Beet`	`Mend`	`Guide`	`Book`
`FRA,MUC,BER`	0	1.159	1.160	1.138	1.140
`Beet`	1.159	0	**0.742**	0.974	0.975
`Mend`	1.160	**0.742**	0	0.968	0.970
`Guide`	1.138	0.974	0.968	0	0.875
`Book`	1.140	0.975	0.970	0.875	0

	`FRA,MUC,BER`	`Beet,Mend`	`Guide`	`Book`
`FRA,MUC,BER`	0	1.349	1.138	1.140
`Beet,Mend`	1.349	0	1.036	1.038
`Guide`	1.138	1.036	0	**0.875**
`Book`	1.140	1.038	**0.875**	0

As noted in Section 5.3.1, we can see in practice how unclustered entities tend to form clusters among themselves, rather than being separately merged into a larger cluster, as in the case of median linkage. `Guide` and `Book` would be merged even if their pairwise distance were closer to 1.

	`FRA,MUC,BER`	`Beet,Mend`	`Guide, Book`
`FRA,MUC,BER`	0	1.349	1.271
`Beet,Mend`	1.349	0	**1.109**
`Guide, Book`	1.271	**1.109**	0

	`FRA,MUC,BER`	`Beet, Mend, Guide, Book`
`FRA,MUC,BER`	0	**1.388**
`Beet, Mend, Guide, Book`	**1.388**	0

The dendrogram for this example clearly shows the formed clusters (Figure A.2) with pairwise distances between elements scaled down due to the larger (than 1) distances between clusters.

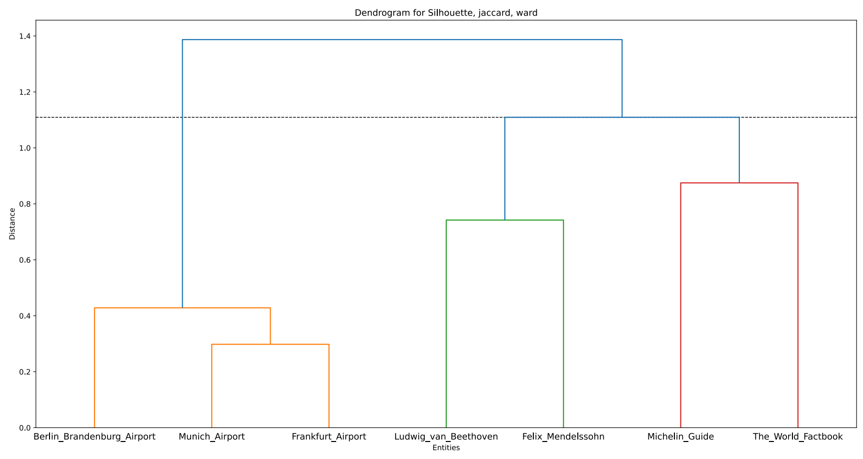

Figure A.2: Ward Linkage Example: Dendrogram.

B Additional Notes on the Silhouette Coefficient

B.1 Lower Maximum Values of SC for Jaccard Similarity

To illustrate how Jaccard similarity can compute similarities differently from Sorensen and cosine similarity, we present the following simplified case: consider two candidate descriptions CD_i and CD_j with $|CD_i| = |CD_j| = n$ and $|CD_i \cap CD_j| = k$.

For Jaccard similarity, we calculate:

$$Jaccard(CD_i, CD_j) = \frac{|CD_i \cap CD_j|}{|CD_i \cup CD_j|} = \frac{k}{2n - k} = \frac{\frac{k}{n}}{2 - \frac{k}{n}} \tag{B.1}$$

For Sorensen and cosine similarity:

$$\begin{aligned} Sorensen(CD_i, CD_j) &= \frac{2|CD_i \cap CD_j|}{|CD_i| + |CD_j|} = \frac{k}{n} \\ &= \frac{CD_i \cdot CD_j}{\|CD_i\|\|CD_j\|} = cosine(CD_i, CD_j) \end{aligned} \tag{B.2}$$

Setting $x = \frac{k}{n}$, and converting to distance by $1 - \textit{cd-sim}(CD_i, CD_j)$, we get:

$$distance_J(CD_i, CD_j) = 1 - Jaccard(CD_i, CD_j) = \frac{x}{2 - x} \tag{B.3}$$

$$distance_S(CD_i, CD_j) = distance_C(CD_i, CD_j) = 1 - x \tag{B.4}$$

where Jaccard distance is denoted by $distance_J$, while Sorensen and cosine distances are denoted by $distance_S$ and $distance_C$, respectively.

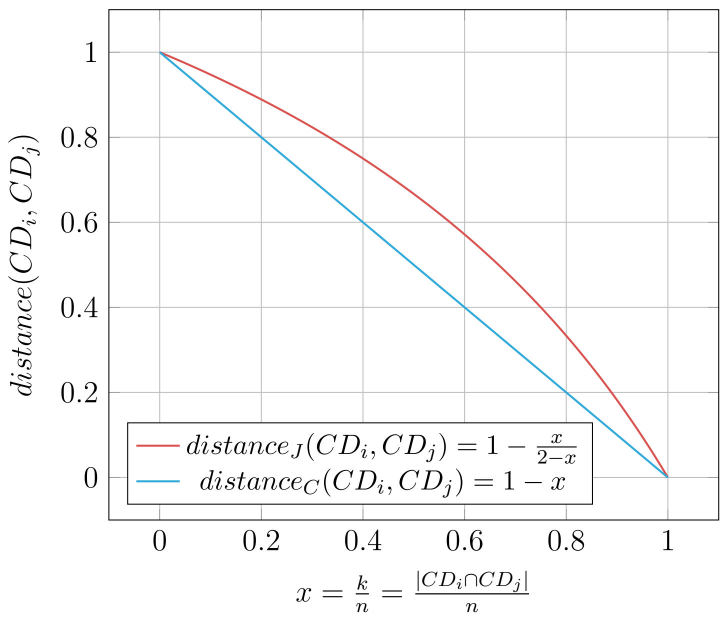

Figure B.1: Comparison: Jaccard vs. Sorensen and Cosine Distances.

In our datasets most of the computed distances are large. For example, in the dataset from Section 5.2, only 1% of the computed pairwise distances across all similarity measures were smaller than 0.5 and around 90% were larger than 0.8, we also assume that in our best-rated clustering $U^k = \text{argmax}_{U^i} U^i$ (as in Algorithm 3.1) almost every candidate description CD_i has been assigned to the correct cluster, i.e., $max(\{a(CD_i), b(CD_i)\} = b(CD_i))$ (with our notation from Equation 3.11). Having these prerequisites and applying the same subscripts as in Equations B.3 and B.4 to the definition of the silhouette for a candidate description $i := CD_i$ (Equation 3.11), we can state the following inequality for the vast majority of CDs i:

$$Sil_S(i) = Sil_C(i) = \frac{b_C(i) - a_C(i)}{b_C(i)} = \frac{c_1 b_J(i) - c_2 a_J(i)}{c_1 b_J(i)} \overset{c_2 < c_1}{>} Sil_J(i) \quad \text{(B.5)}$$

where c_1 and c_2 are constants. It is evident that $c_2 < c_1$ because, for any distance measure, the average distance to the neighbor cluster $b(i)$ would be higher than the average intra-cluster distance $a(i)$. Given that $b(i)$ tends closer to 1 than $a(i)$ (in terms of distance, y-axis in Figure B.1), and assuming $b(i) > a(i) \geq 0.5$, c_2 must be lower than c_1 to compensate for the larger difference between the distance measures at a lower average distance $a(i) \in [0.5, 1]$ (for $x \in [0, 0.5]$).

B.2 Silhouette Coefficient of Additional Datasets

In the following, we present the Silhouette coefficient of the best-rated clusterings U^k for different input texts. $|I|$ denotes the number of entities which have been found in the text and included in the clustering process. The figures below use the same legend as those in Section 5.3.2.

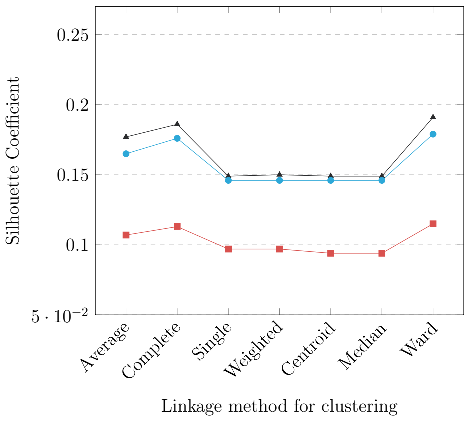

Figure (B.1) Linked Data[1], $|\mathcal{I}| = 17$.

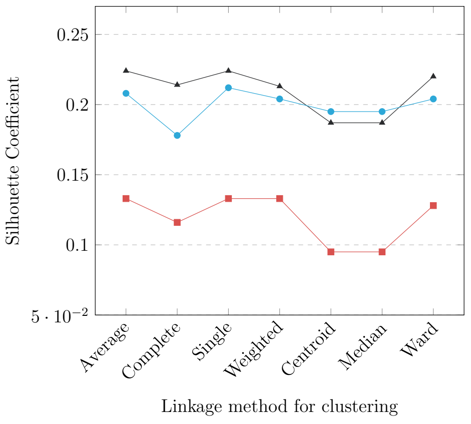

Figure (B.2) Data Science[2], $|\mathcal{I}| = 21$.

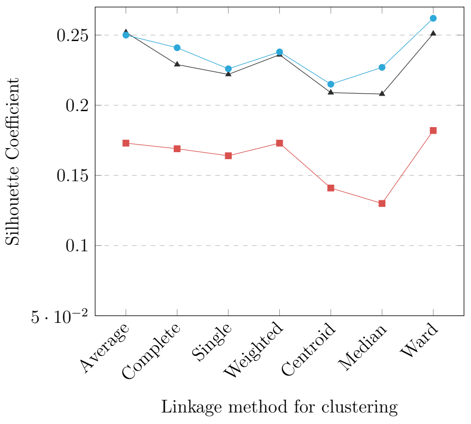

Figure (B.3) Evolution[3], $|\mathcal{I}| = 91$.

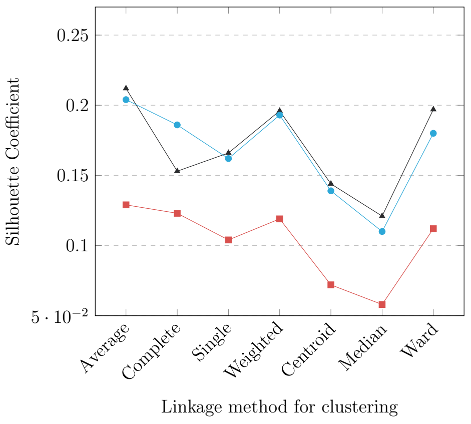

Figure (B.4) Frida Kahlo[4], $|\mathcal{I}| = 167$.

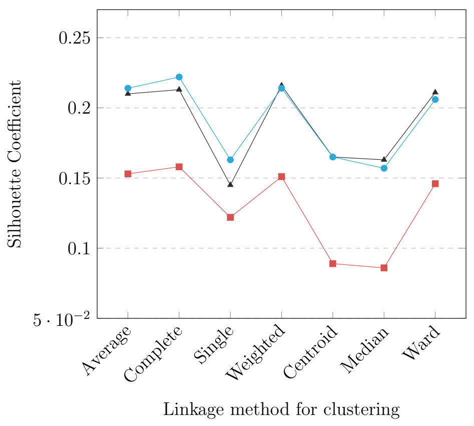

Figure (B.5) Barack Obama[5], $|\mathcal{I}| = 271$.

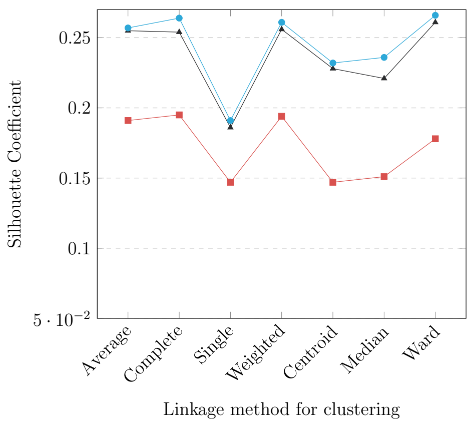

Figure (B.6) Sweden[6], $|\mathcal{I}| = 319$.

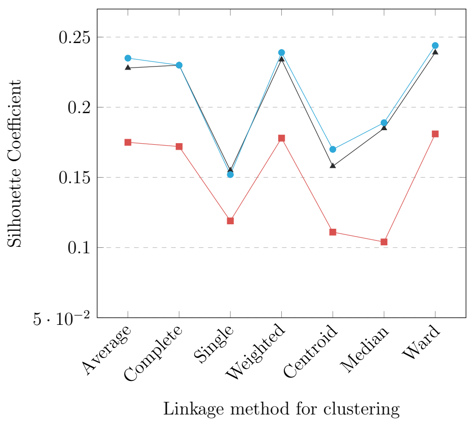

Figure (B.7) Poland[7], $|\mathcal{I}| = 390$.

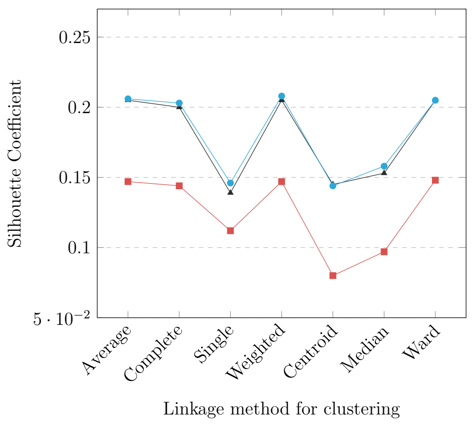

Figure (B.8) France[8], $|\mathcal{I}| = 489$.

[1] `https://en.wikipedia.org/wiki/Linked_data`
[2] `https://en.wikipedia.org/wiki/Data_science`
[3] `https://en.wikipedia.org/wiki/Evolution`
[4] `https://en.wikipedia.org/wiki/Frida_Kahlo`
[5] `https://en.wikipedia.org/wiki/Barack_Obama`
[6] `https://en.wikipedia.org/wiki/Sweden`
[7] `https://en.wikipedia.org/wiki/Poland`
[8] `https://en.wikipedia.org/wiki/France`

C Result Clusters: Examples

In this appendix, we present a small subset of different clusters that have resulted from our pipeline. We note that the labels of the clusters, based on values of the DBpedia property `gold:hypernym`, are shown only for illustrative purposes and were not used as a basis for evaluation in our work.

C.1 Accurate Clusters

```
"Composer 2": {
    "Felix_Mendelssohn": "Composer",
    "Ludwig_van_Beethoven": "Composer",
    "Richard_Wagner": "Composer",
    "Karlheinz_Stockhausen": "Composer",
    "Richard_Strauss": "Composer",
    "Robert_Schumann": "Composer",
    "Carl_Maria_von_Weber": "Composer"
}

"Ceremony 1": {
    "Academy_Awards": "Ceremony",
    "Pritzker_Architecture_Prize": "Not found",
    "European_Film_Awards": "Not found",
    "Deutscher_Filmpreis": "Ceremony"
}

"Reconstruction 1": {
    "Proto-Germanic_language": "Completion",
    "Proto-Indo-European_language": "Reconstruction"
}
```

C.2 Partly Accurate Clusters

```
"Dynasty 1": {
    "Hohenstaufen": "Dynasty",
    "House_of_Habsburg": "Not found",
    "Bundesliga": "League",
    "Salian_dynasty": "Dynasty"
}
```

```
"Body 1": {
    "Babelsberg_Studio": "Studio",
    "Eurostat": "General",
    "World_Bank": "Institution",
    "European_Commission": "Body",
    "Cambridge_University_Press": "Business",
    "Frankfurt_Stock_Exchange": "World",
    "European_Central_Bank": "Bank",
    "European_Space_Agency": "Organisation"
}
```

C.3 Inaccurate Clusters

```
"System 1": {
    "LGBT": "Initialism",
    "Autobahn": "System",
    "Reformation": "Not found",
    "Evangelicalism": "Worldwide",
    "Art_Deco": "Style"
}

"Term 1": {
    "Peaceful_Revolution": "Process",
    "Golden_Twenties": "Term",
    "Wirtschaftswunder": "Not found",
    "Great_Depression": "Depression",
    "German_Revolution_of_1918\u201319": "Conflict",
    "UEFA_European_Championship": "Competition",
    "Olympic_Games": "Event",
    "Berlin_Fashion_Week": "Week"
}

"Band 1": {
    "Kraftwerk": "Band",
    "Deutsche_Welle": "Germany",
    "Neanderthal": "Species",
    "Venus": "Planet",
    "Brick_Gothic": "Style"
}
```

Bibliography

Abdulkareem Alsudais and Hovig Tchalian. Clustering prominent named entities in topic-specific text corpora. In *25th Americas Conference on Information Systems, AMCIS 2019, Cancún, Mexico, August 15-17, 2019*. Association for Information Systems, 2019.

Marcelo Arenas, Claudio Gutierrez, and Jorge Pérez. Foundations of RDF databases. In *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*, pages 158–204. Springer, 2009. doi: 10.1007/978-3-642-03754-2_4.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007. doi: 10.1007/978-3-540-76298-0_52.

Tim Berners-Lee. Linked data. World Wide Web Consortium (W3C), 2006. URL `https://www.w3.org/DesignIssues/LinkedData`.

Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2370–2381. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-EMNLP.204.

Klitos Christodoulou, Norman W. Paton, and Alvaro A. A. Fernandes. Structure inference for linked data sources using clustering. *Trans. Large Scale Data Knowl. Centered Syst.*, 19:1–25, 2015. doi: 10.1007/978-3-662-46562-2_1.

Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 2014. URL `https://www.w3.org/TR/rdf11-concepts/#dfn-literal`.

Siham Eddamiri, El Moukhtar Zemmouri, and Asmaa Benghabrit. An improved rdf data clustering algorithm. In *Procedia Computer Science*, volume 148, 2019. doi: 10.1016/j.procs.2019.01.038.

Mahmoud Elbattah, Mohamed Roushdy, Mostafa Aref, and Abdel Badeeh M. Salem. *Big Data Analytics: Tools and Technology for Effective Planning*, chapter Large-scale entity clustering based on structural similarities within knowledge graphs. 2017. doi: 10.1201/b21822.

Holmes Finch. Comparison of distance measures in cluster analysis with dichotomous data. *Journal of Data Science*, 3, 2021. ISSN 1680-743X. doi: 10.6339/jds.2005.03(1).192.

Mohamed H. Gad-Elrab, Daria Stepanova, Trung-Kien Tran, Heike Adel, and Gerhard Weikum. Excut: Explainable embedding-based clustering over knowledge graphs. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, volume 12506 of *Lecture Notes in Computer Science*, pages 218–237. Springer, 2020. doi: 10.1007/978-3-030-62419-4_13.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020. doi: 10.5281/zenodo.1212303.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. Unsupervised graph-based topic labelling using dbpedia. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 465–474. ACM, 2013. doi: 10.1145/2433396.2433454.

Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, pages 224–238. John Wiley, 1990. ISBN 978-0-47187876-6. doi: 10.1002/9780470316801.

Tomás Kliegr and Ondrej Zamazal. LHD 2.0: A text mining approach to typing entities in knowledge graphs. *J. Web Semant.*, 39:47–61, 2016. doi: 10.1016/J.WEBSEM.2016.05.001.

G Klyne and J.J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 2004. URL `https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

Jongwuk Lee, Hyunsouk Cho, Jin-Woo Park, Young-rok Cha, Seung-won Hwang, Zaiqing Nie, and Ji-Rong Wen. Hybrid entity clustering using crowds and data. *VLDB J.*, 22(5):711–726, 2013. doi: 10.1007/S00778-013-0328-8.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM International Conference Proceeding Series, pages 1–8. ACM, 2011. doi: 10.1145/2063518.2063519.

Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *CoRR*, abs/1109.2378, 2011.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL, 2003. doi: 10.3115/1119176.1119195.

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, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, 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. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. Collabonet: collaboration of deep neural networks for biomedical named entity recognition. *BMC Bioinform.*, 20-S(10):55–65, 2019. doi: 10.1186/S12859-019-2813-6.

Declaration of Academic Integrity

I hereby confirm that this thesis, entitled *RDF-driven Entity Clustering of Unstructured Data*, is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited. I am aware that plagiarism is considered an act of deception which can result in sanction in accordance with the examination regulations.

Nikolay Krasimirov Kazanliev, Münster, June 17, 2024

I consent to having my thesis cross-checked with other texts to identify possible similarities and to having it stored in a database for this purpose.

I confirm that I have not submitted the following thesis in part or whole as an examination paper before.

Nikolay Krasimirov Kazanliev, Münster, June 17, 2024