Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTERTHESIS

Unsupervised Entity Disambiguation Using Pretrained Contextualized Word Embeddings

Frederik Wille

Field of Study: M.Sc. Computer Science
Matr.-Nr. 6533918

First Reviewer: Prof. Dr. Chris Biemann
Second Reviewer: Dr. Seid Muhie Yimam
Supervisor: Steffen Remus

Date of Submission: 25.04.2022

Abstract

Mentions of named entities are an integral part of natural language. They link pieces of text or speech to the entities of the real world. Named entities might be physical entities like persons and places, as well as abstract entities like organizations. Unfortunately, names can be ambiguous. For example, when mentioning “Michael Jordan”, most people will think of the former basketball player and name giver of the Air Jordan sneakers. However, at the time of writing, Wikipedia lists 13 additional famous persons with that name as well as media about the basketball player, e.g. movies. Consequently, disambiguation of entity mentions is essential to natural language processing.

In this thesis, we propose an unsupervised approach to entity disambiguation. We introduce a heuristic classification of unambiguous entity mentions limiting unambiguity to the scope of a knowledge base. Our approach clusters the unambiguous mentions based on their contextualized word embeddings. The clustering captures semantic similarities between the mentions through their embeddings. To disambiguate a mention, we calculate the most similar cluster. Ambiguous mentions that have the same surface form but refer to different entities are matched to different clusters. In contrast, mentions referring to the same entities are matched to the same cluster. Additionally, we describe the clusters with short texts based on the entities’ descriptions on a knowledge base and leverage those descriptions for disambiguation. For example, Michael Jordan might be described with “former basketball player”.

Our experiments show that the heuristic sufficiently classifies mentions as ambiguous or unambiguous. The results further indicate that we successfully created clusters of semantically related entities. Nonetheless, we can not yet compare our approach to others due to limitations of the currently available datasets.

Contents

1. Introduction

Named entities are real-world objects referred to by proper names. Their mentions, i.e. occurrences of their names, make up a significant part of natural languages. For example, the news articles that we use in this thesis mention entities over 35 000 times in about 22 000 sentences. That is more than one mention per sentence and it does not even include all co-references referring to entity mentions in other parts of an article. As a result, gaining an understanding of named entities gives an advantage in various natural language processing (NLP) problems. For example, limiting results to those semantically fitting can improve information retrieval applications (Tan et al., 2017). Thus, the NLP community has researched several problems around entity mentions for decades (Nadeau and Sekine, 2007). The research started with named entity recognition (NER), i.e. finding entity mentions in natural language. In the last few years, the focus has been shifted to linking mentions to entity items in knowledge bases (KBs) and categorizing mentions into more granular classes. Furthermore, there are special interests like domain transferability to adapt to specialized domains like medical research.

Another recent topic in NLP are contextualized word embeddings (CWE). Word embeddings are, in general, numerical representations of words. Compared to a one-hot encoding of the same dictionary, they are placed in a low-dimensional space. These representations are calculated in a way that incorporates the word's sense. The numerical form also allows machine learning techniques to process the incorporated sense. Recently, pretrained transformer models like BERT (Bidirectional Encoder Representations from Transformer; Devlin et al., 2019) provide CWE. In contrast to previous embedding approaches, CWEs capture the context in which a word occurs. Simplified, the previously used static word embeddings only train a mapping from strings to embeddings which is based on the contexts of the word in the embedding model's training data. Through the focus on a word's context, CWEs have a particular advantage in understanding named entities. Entity names consist of open-class words and there will always be unknown names no matter the size of the training data. Thus, disambiguating entity names should focus more on the context than the names themselves. For example, the name "Michael Jordan" by itself does not convey much about the mentioned person. However, having basketball established as the context gives a good notion of who might be mentioned.

In this thesis, we show an approach to entity disambiguation that uses CWEs of unambiguous entity mentions to detect semantic similarities between related entities. We introduce a heuristic on unambiguity that limits the scope of unambiguity to a KB during training. The heuristic considers the names and aliases of entities listed in the KB to

detect if there are multiple candidates for a mention. Otherwise, the mention is deemed unambiguous. Instead of addressing entity disambiguation as a classification problem with supervised learning, we tackle the issue using unsupervised learning. Ultimately, we build a model that describes an entity mention with a short text based on the similarity to mentions in the training data. We publish our code on Github[1].

1.1. Entity Mentions

This thesis is concerned with named entity mentions in natural language. Generally, named entities are referred to by proper names like those of persons, organizations and locations. We define the following terms for this thesis:

- entity: concept of a real world object (like persons, organizations, and locations), can be identified in a KB by a URI; we use Wikidata's items as entities;
- mention: occurrence in natural language referring to an entity by a proper name;
- surface form: the concrete string of a mention, may consist of multiple words.

While the exact definition of what constitutes a named entity mention might vary (Nadeau and Sekine, 2007), we limit it to those entities represented in the Wikidata[2] KB and mentioned by one of the names denoted in that Wikidata item. At least for the training part of our methodology, we omit mentions that are not part of the KB. Even so, during inference, our methodology works independently of a strict definition and can be used on any kind of sequence[3]. To apply our heuristic during training, we process the surface form of a mention as well as the possible surface forms of the entities in the KB. The surface form of a mention is the sequence, i.e. one or more words, that constitutes the mention. For example, in the following sentence, the surface forms of the mentions are "The Hitchhiker's Guide to the Galaxy" and "Douglas Adams".

The author of The Hitchhiker's Guide to the Galaxy is Douglas Adams.

For an entity, the possible surface forms are the names that might refer to the entity. The Wikidata entity Q42[4] can be referred to by the full name "Douglas Noël Adams" as well as shorter versions like in the mention above. Of course, entities can also have nicknames and aliases that are treated equally.

1.2. Motivation

Understanding of entities has been part of NLP research for years. Entity-focused tasks, like NER and entity linking (EL), have improved with other parts of NLP. The latest

[1] https://github.com/3wille/Unsupervised-Entity-Disambiguation
[2] https://www.wikidata.org/
[3] a sequence can be any length of text, e.g. a sentence, a paragraph or even one or more documents
[4] https://www.wikidata.org/wiki/Q42

advancement of transformers is already in use for a variety of NLP tasks, including entity linking. However, most of that work seems to be focused on supervised learning and thus annotated data. Although most-recent work tends to go in the direction of distant supervision (e.g. Choi et al., 2018), supervised learning does not take advantage of the vast knowledge available in unlabeled corpora and KBs.

By using heuristically unambiguous surface forms of entities, we ensure that a CWE is a proper representation of a specific entity in the KB. Thus, we do not need labeled data for training our model but work completely unsupervised. Without the heuristic, there is no trivial way of determining an entity for a mention. With it, any kind of corpus can serve as training data as long as the mentioned entities are present in the used KB.

1.3. Hypothesis

In this thesis, we present our research on a method of unsupervised learning for entity disambiguation. We detect unambiguous mentions in our training data and learn similarities between their CWEs. Our goal is to leverage the learned knowledge about the unambiguous entity mentions for entity disambiguation.

The unambiguity of a mention is a central part of our proposition. It is used to provide a set of entity links from which we gain further knowledge later on. We introduce a heuristic to detect whether a mention in our training data can be considered unambiguous or not. The heuristic considers the names and aliases procured on Wikidata. Using ontologies like Wikidata, we consider the open world assumption (OWA). The OWA states that a knowledge base may always be incomplete (Hitzler et al., 2010). For our case, this means that we are aware that unambiguity determination will never be perfect using the heuristic. Nevertheless, we hypothesize that we retrieve sufficiently unambiguous mentions based on the heuristic.

Once we have established a set of unambiguous mentions, we cluster them based on the similarity of their CWE. The ability of CWE to represent semantics is shown in previous work (e.g. Wiedemann et al., 2019). We expect that the clustering yields meaningful groups of related entities and that these groups can be used for entity disambiguation. For the clustering, we also leverage that all mentions are unambiguous. It allows us to treat all mentions with the same surface form as referring to the same entity. Thus, we can merge their representations before the clustering.

Then, we use the semantically related groups for disambiguation. Ultimately, we match ambiguous mentions to the clusters and produce a free-form description. These descriptions are pre-generated for each cluster and have similarities to ultra-fine entity typing (Choi et al., 2018).

For this thesis, we pose the following research questions:

1. Can we find correct mention-to-KB-item links using the unambiguity heuristic?
2. Can CWEs be used to produce clusters of semantically related entities?

3. How can we generate descriptions for clusters of entities?

4. Can the clusters be leveraged to disambiguate candidate entities?

1.4. Thesis Structure

The rest of this thesis is segmented into four chapters. Chapter 2 introduces past works relevant to this thesis, especially those around embeddings and entity disambiguation. In Chapter 3, we describe our novel methodology for working with entities. The experiments based on our methodology and their results are laid out in Chapter 4. Chapter 5 discusses our work and concludes this thesis.

2. Related Work

This chapter provides background information relevant to the thesis. First, we present Wikidata and Chinese Whispers (CW; Biemann, 2006) that are essential parts of our methodology. Then, we describe methods for calculating vector space representations (VSRs). Last, we place our work in the context of other entity-focused tasks and methodologies.

2.1. Background

Wikidata

Central parts of our methodology use the KB Wikidata[1] (Vrandečić and Krötzsch, 2014). Wikidata is a multilingual KB developed and offered by Wikimedia, the organization behind the encyclopedia Wikipedia. While people all over the world have amassed huge amounts of knowledge on Wikipedia since its inception in 2001, its unstructured content poses a challenge for programmatic access. Furthermore, the same information on different pages or languages on Wikipedia can be inconsistent.

> Population numbers for Rome, for example, can be found in English and Italian articles about Rome but also in the English article "Cities in Italy". The numbers are all different. (Vrandečić and Krötzsch, 2014)

To address the lack of structure and consistency, the Wikimedia Foundation launched Wikidata in October 2012. On Wikidata, some data values like the population number of Rome are stored independently of language, i.e. the value for "population number of Rome" is stored only once. Wikidata stores information like the population number in property-value pairs[2] attached to an item. In the example, Rome is the item and the population number is the property. Together with the value, they form a Semantic Web triple (Hitzler et al., 2008) consisting of subject, predicate, and object. Wikidata can also store language-dependent information, e.g. the property representing population numbers can be annotated with a translation for different languages. Further, items can have language-specific properties like the name, possible aliases, and a description that may be different in every language. Throughout this thesis, we use exactly these three properties: name and aliases as possible surface forms of the referenced Wikidata item, as well

[1] https://www.wikidata.org

[2] we use the terminology as used by Vrandečić and Krötzsch (2014) which partly collides with Semantic Web terminology, e.g. in (Hitzler et al., 2008)

as the descriptions (see Figure 2.1 for an example). However, as a KB, Wikidata is subject to the OWA (Hitzler et al., 2008), i.e. the KB has to be considered as incomplete at all times. Thus, the absence of content must be considered as unknown rather than being treated as the negation of the content (Arnaout et al., 2021). For example, the item of author Douglas Adams[3] states that he received the Inkpot Awards. However, the item does not state whether Adams received the Nobel Prize in Literature, so we cannot assume whether he did or did not, although the latter is true.

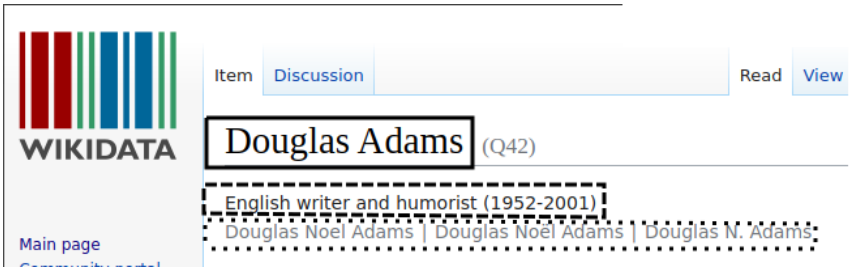


Figure 2.1.: Douglas Adams' item on Wikidata with name, description and aliases. Retrieved on 21.3.2022.

Chinese Whispers

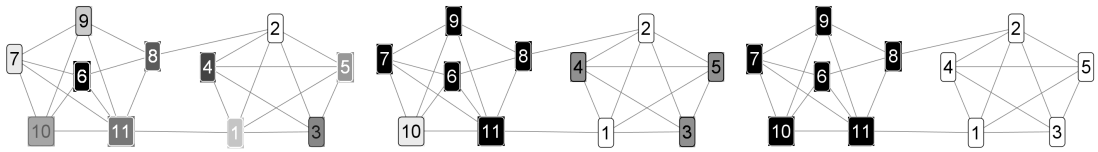
Chinese Whispers (CW; Biemann, 2006) is a graph clustering algorithm specifically designed with NLP in mind. The author's goal is to allow clustering of the large datasets that are common in NLP. The algorithm works in a bottom-up method, i.e. all nodes are initialized with a distinct class. Then, iterating over all nodes only a few times, each node changes its own class to the strongest class amongst its neighbors. The strongest class is the class with the maximal sum of all edge weights. The pseudocode of the algorithm can be seen in Listing 2.1. During the iterations, groups of close nodes will stabilize to the same class. These groups grow until they meet the border of another stable class. Figure 2.2 shows the algorithm clustering an example graph. CW does not converge in a formal sense, but Biemann shows that it reaches a state of almost-convergence quickly. In our experiments, we see that a few iterations are enough to yield promising clusters. To be used for large datasets, the clustering algorithm needs to be efficient. However, using a graph as the representation of the data also allows us to apply optimizations, like using edge pruning to reduce the complexity of the data. Other algorithms often work on a similarity matrix that cannot be pruned so easily. The performance of CW is especially good in small-world graphs. Small-world graphs (Watts, 2000) are characterized by a high clustering coefficient[4] and low average shortest path length between pairs of nodes. To determine whether a graph is a small-world, it is compared to equivalent graphs that are either completely random or regularly structured (see Figure 2.3). Anecdotally, small-world graphs represent the social small-world phenomenon (Milgram, 1967) which claims that

[3]https://www.wikidata.org/wiki/Q42, retrieved 21.3.22
[4]a high clustering coefficient means that the average of nodes have highly connected neighbors

```
initialize:
  forall v_i in V: class(v_i)=i;
while changes:
  forall v in V, randomized order:
    class(v)=highest ranked class in neigborhood of v;
```

Listing 2.1: Chinese Whispers in pseudocode as presented by Biemann (2006)

most people know each other by just a few intermediary acquaintances. Biemann (2006) lists a few examples of small-world occurrences in NLP research. We later show evidence that our graph has small-world characteristics as well, so the high performance in quality and speed of CW on these graphs is of interest to us.



(a) Initialization (b) First Iteration (c) Second Iteration

Figure 2.2.: An example of Chinese Whispers clustering an unweighted graph with 11 nodes in 2 iterations (from Biemann, 2006)

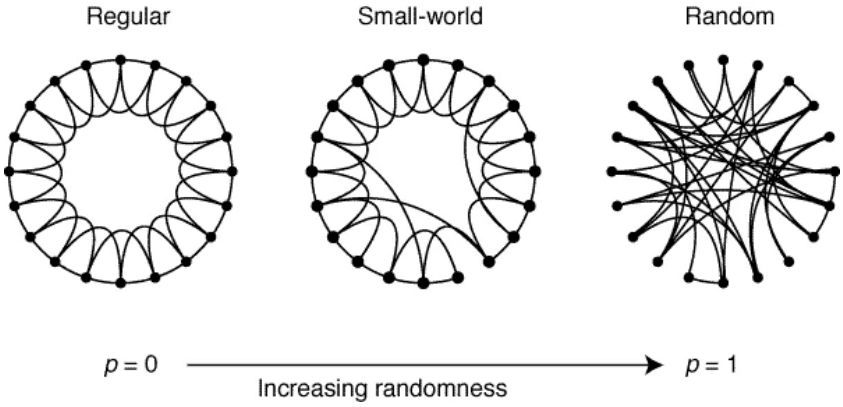


Figure 2.3.: Small-worlds graph are a middle ground between structured/regular and random graphs. All three graphs have the same number of nodes and edges. (from Watts and Strogatz, 1998)

2.2. Vector Space Representations

Vector space representations (VSRs) are an integral part of NLP. In general, VSR models place a piece of text in a multi-dimensional space of real numbers. As a numerical representation of text, they allow modern machine learning approaches, especially neural networks, to be used. Depending on the specific algorithm, VSR can represent any type of text: whole documents, paragraphs, sentences, words, or just parts of words. There is

also research on compositionality, i.e. how to aggregate multiple representations at one level to represent a higher level. For example, Reimers and Gurevych (2019) investigate averaging word embeddings to represent a sentence. In this section, we focus on word-level embeddings and later aggregate those to mention-level representations. The goal of embedding algorithms is to produce word embeddings that map semantically similar words into similar vectors. To achieve that goal, the algorithms presented here are based on the distributional hypothesis by Harris (1954) which states that words that occur in the same contexts tend to have similar senses. Here, we take a look at two different kinds of VSR:

- static embeddings;
- contextualized word embeddings (CWE).

Static word embeddings are static with regard to the surface form of a word. They produce the same vector for each surface form regardless of the word's sense. Thus, they cannot discern between polysemes, i.e. words with the same spelling but different senses. Mikolov et al. (2013) introduce the static word embeddings model called Word2Vec. Word2Vec was a novelty at the time for its simpler model compared to previous works that allowed it to be trained on significantly larger datasets. It is composed of two submodels that are also illustrated in Figure 2.4:

- continuous bag-of-words (CBOW): predict a word given its context, i.e. the surrounding words;
- continuous skip-gram: given a word, predict its context.

The usage of those two models allows Word2Vec to be trained more efficiently compared to previous language model (LM) approaches, meaning more training data can be used with the same effort in both computational resources and time. Building on the skip-gram model, Bojanowski et al. (2017) introduce FastText. FastText expands Word2Vec's capabilities by processing character n-grams rather than whole words. The character n-grams have two advantages. First, the model recognizes and thus learns morphological similarities between words. An example provided by the authors is the German compound noun *Autofahrer* (car driver), which consists of two components: *Auto* and *fahrer*. FastText returns the average of those components. Second, it produces embeddings for many words that are out-of-vocabulary (OOV) for Word2Vec on the same training data because a word's n-grams are more likely to have appeared in the training data. The latter advantage is especially of importance to our methodology because entity mentions are often OOV on a word level, even for extensive corpora.

Contextualized word embeddingss (CWEs) are, in contrast to static embeddings, dependent on the embedded token's context. Through the distributional hypothesis (Harris, 1954), they carry some notion of an ambiguous word's sense (as shown in Wiedemann et al. (2019) that we introduce later on). One of the most prominent models for CWE is

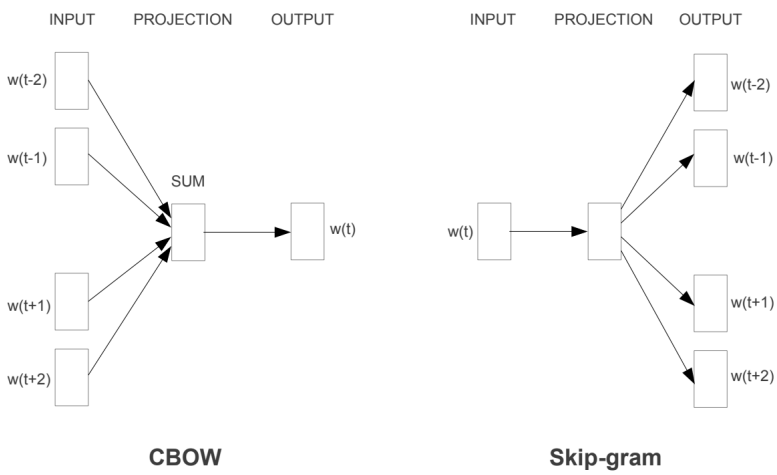


Figure 2.4.: Word2Vec models: CBOW predicts a word given its context; given a word, Skip-gram predicts its context (from Mikolov et al., 2013)

BERT (Bidirectional Encoder Representations from Transformer; Devlin et al., 2019). It is based on the transformer technology initially introduced by Vaswani et al. (2017) as a model for machine translation. The original transformer is composed of an encoder and a decoder. The encoder learns to produce a representation of the input that the decoder can turn into an output sequence in the target language. The novelty in Vaswani et al. (2017) is that the previously used recurrency in neural networks can be replaced with the self-attention mechanism. With attention, all parts of the model can focus on different values of the input or previous layers. Simplified, the attention is a factor applied to the output of a network layer that increases specific values dynamically. This allows cost-effective modeling of long-range dependencies between words. Before attention, learning such dependencies was ineffective because it required the value to traverse through multiple layers because the previous models could only consider a few words around the current word.

BERT needs a way to deal with OOV words as well. For this purpose, it employs WordPiece (Wu et al., 2016). WordPiece is a tokenizer originally built for languages like Korean or Japanese. These languages have vastly large vocabularies as well as fewer spaces between tokens than Latin-based languages (Schuster and Nakajima, 2012). This property requires more complex tokenization in addition to a method to avoid errors on OOV words. To train a WordPiece model, the vocabulary is initialized with the Unicode characters relevant to the respective language. Then, a greedy algorithm chooses the pair of vocabulary items that increases the likelihood of a language model the most when the model is trained on the training data using the new vocabulary. Training completes once the vocabulary reaches a defined target size or the increase in likelihood falls below a threshold. The result is a tokenizer model that splits words into subtokens but can also leave tokens as is when they are frequent enough in the training data.

In BERT, WordPiece is used to treat OOV words for latin-based languages as well. To keep parameter complexity within limits, BERT limits the vocabulary of WordPiece

to 30 000 tokens which is much less than the 200 000 used for Japanese and Korean by Schuster and Nakajima (2012). In BERT's WordPiece model, subtokens following the first are prefixed with `##` to denote which tokens are the result of a split. As an example, the word `Hitch` is OOV and therefore it is split into two known subtokens: `Hit` and `##ch`. Based on the WordPiece inventory, a mapping from tokens to integers is calculated. These integers are then used as the inputs for the BERT model during training and inference in place of the tokens themselves. Architecturally, BERT (Devlin et al., 2019) mostly replicates the model in Vaswani et al. (2017) but only includes the encoder leaving out the decoder. Now, the previously internal representations serve as the output of the model. Training of BERT itself is conducted targeting two tasks:

- Masked language model (MLM): during training, some of the tokens in a sentence are masked (i.e. hidden) from the model and the model is asked to predict what the token is;
- Next sentence prediction (NSP): the model is given two sentences and is trained to decide on whether those appear together in the training data.

The masked language model task contributes especially to the model being trained in a way that allows us to retrieve CWEs (Wiedemann et al., 2019) while the NSP task is more helpful to some downstream tasks like question answering (Devlin et al., 2019). BERT is pre-trained on a large dataset. The authors publish the learned parameters for use in downstream tasks. The pre-trained model is then intended to be used in a fine-tuning strategy for which it is meant to receive further training on a dataset-specific to the downstream task. However, the authors also experiment with a feature-based approach, i.e. using BERT's output as features for another model rather than fine-tuning the BERT model itself. We build upon this way of using BERT by leveraging the output as CWEs.

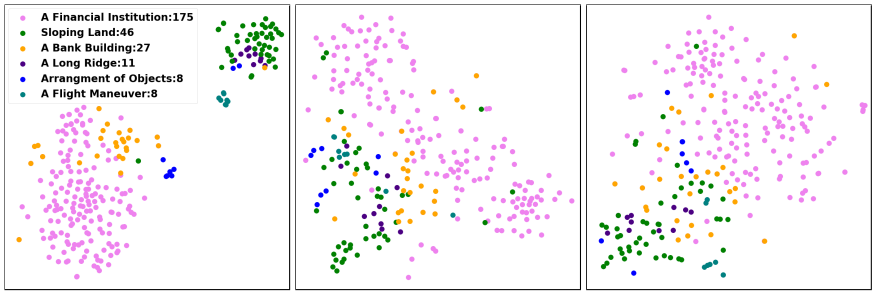
RoBERTa (Robustly optimized BERT Pretraining Approach; Liu et al., 2019) is a replication-study of BERT (Devlin et al., 2019). To show further capabilities of the BERT architecture, Liu et al. make the following adjustments to it:

- increasing training length, data, and batch size;
- removing the next sentence prediction (NSP) task;
- training on longer sequences, i.e. always making the best possible use of the 512 tokens limit per sequence;
- dynamically changing the masking pattern in MLM to diversify the task.

Increasing the training data is an obvious choice as the gains of larger datasets for LMs were known even before Word2Vec (Mikolov et al., 2013). In contrast to some other works that increased the size of BERT's training data, the authors publish their dataset, which aggregates five different corpora, to be used in future work. Larger batches are easier to parallelize and show better downstream task performance in the paper. Based on

previous works, the authors argue that the training loss generated by the NSP task does not yield advantages and might even hinder the LM learning. They further find that it is more important to use longer sequences and not individual sentences to learn long-distance dependencies adequately. These adjustments together contribute to increased performance over BERT that the authors show in their experiments.

To assess the semantic capabilities of CWEs, especially those produced by BERT, Wiedemann et al. (2019) propose a k-nearest neighbor (KNN) classifier on CWEs. The classifier is used to tackle the word sense disambiguation (WSD) task comparing the performance of three CWE producing models on different datasets. The authors chose KNN to be able to gain insight into classifier decisions that support vector machines or neural networks would not allow. This insight is displayed in the example of six senses of the word "bank" in comparison with the previous embedding models Flair (Akbik et al., 2018) and ELMo (Peters et al., 2018). The contextualized embeddings for "bank" are shown in a t-distributed stochastic neighbor embedding (t-SNE; Van der Maaten and Hinton, 2008)[5] plot color-coded by their sense that we show in Figure 2.5. From the recognizable clusters in the BERT example, the authors conclude that BERT can encode some form of knowledge about word senses in its CWEs while Flair and ELMo seem to lack this kind of information. We expect this knowledge of sense disambiguation to work similarly for entity disambiguation due to the similarity of the tasks. Example usage of this similarity can be seen in Moro et al. (2014) where a unified approach to both entity and word sense disambiguation is shown.



(a) BERT (b) Flair (c) ELMo

Figure 2.5.: Distribution of the embeddings from three models for the word 'bank' color-coded by their sense; from Wiedemann et al. (2019)

2.3. Entity Linking & Typing

The purpose of this thesis is to gain an understanding of entity mentions. Computational understanding of entity mentions has been part of NLP research since at least the early 1990s (Nadeau and Sekine, 2007). Historically, while starting with rule-based systems (e.g. Rau, 1991), most techniques have relied on machine learning. Along with the rest

[5] technique to visualize data with more dimensions than displayable

of NLP research, it took important steps forward with the growth of neural network systems (e.g. Lample et al., 2016). Most recent research in the area of named entities is further profiting from the adoption of the transformer architectures like BERT. There are three entity mention focused tasks that we want to mention here:

- Named entity recognition (NER): detect mentions of named entities;
- Entity linking (EL): link a mention to the referred entity in a KB;
- Entity typing (ET): classify/describe a mention with one or more labels.

All three are related to this thesis which we describe as entity disambiguation (ED). ED is often treated as one step in entity linking (EL) deciding which of a number of candidates is referred to by a mention. An example of this is CHOLAN (Kannan Ravi et al., 2021) that we describe in more detail later on. However, we take ED more into the direction of ET by describing mentioned entities rather than producing an explicit link to an entity in a KB while not fulfilling the ET task directly either. Nonetheless, EL is an important task in NLP and we show how this thesis fits into EL, as well as ET, research.

One example of a downstream task for our approach can be zero-shot entity linking. This task is introduced by Logeswaran et al. (2019) along with a dataset and a novel approach called domain-adaptive pre-training (DAP). The authors define the new task as an expansion of the scope of the previous entity linking task. They target a higher generalization, especially to new domains that are possibly unknown at training time. For that, they exclude a few assumptions that are allowed in previous EL tasks:

- a single entity set shared between training and test examples;
- an alias table that maps all names to their possible entities;
- frequency statistics, i.e. probabilities for the alias table;
- structured data, e.g. knowledge graph triples defining relations between entities.

Instead, they only assume access to an entity dictionary for targeting that incorporates entities along with a text description each. For training and testing, they need labeled mention and entity pairs that may come from multiple domains. The dataset published alongside the paper is built on eight Wikias[6] for training and four Wikias each for validation and testing. Each Wikia is an encyclopedia on its own, focusing on a separate topic, e.g. a specific sport like wrestling, lego, or different fictional universes. Thus, the dataset targets the generalization and transferability that the novel task demands. The descriptions in the dataset are different from those used in our methodology. Their descriptions are full sentences, e.g. the *Imperial Armored Transport* from the Star Wars Wikia is described as:

[6] https://www.wikia.com/

> The Kuat Drive Yards Imperial Armored Transport was fifty meters long and carried ten crewmen and twenty soldiers. (Logeswaran et al., 2019)

Meanwhile, our descriptions are shorter and use only a few descriptive words, e.g. the *AT-AT*[7], from the Star Wars franchise as well, is described on Wikidata as "four-legged walker in Star Wars". As for the author's own implementation, they implement a two-stage pipeline. First, as candidate generation, a variant of tf-idf (term frequency-inverse document frequency)[8] is used to produce 64 candidates. Then, they rank the candidates with a transformer architecture based on BERT. As in BERT's NSP task, they use special tokens to separate two sequences that are the mention context and the candidate entity's description. The transformer produces a joint representation for the pairs of context and description, which is passed to a last neural layer that learns a function to a final score for the candidate. In evaluation, Logeswaran et al. (2019) use a similar approach to ours in considering the top-k results. All in all, the zero-shot entity linking task works on similar problems as this thesis, the understanding and disambiguation of entity mentions. The research around entity linking has the potential to evolve further in the direction of higher generalization and domain independence and zero-shot is a step in this direction.

Kannan Ravi et al. (2021) describe their approach to end-to-end entity linking named CHOLAN. Though trained end-to-end, CHOLAN is modular and employs two independent transformer models for mention detection (MD) and ED. For MD, BERT is chosen as a token classifier. Each of the detected mentions is assigned a list of candidate entities for which the authors compare the performance of two different generation methods from previous works. For ED, a mention with its sentence as context and a candidate entity with its context, e.g. a Wikidata description, are fed into a second BERT model. The ED model is trained for binary classification deciding whether a mention belongs to a certain entity. To demonstrate CHOLAN's ability to work across KBs, the authors base their experiments on the T-REx (Elsahar et al., 2018) and AIDA datasets for Wikidata and Wikipedia respectively. At their time of writing, they outperform the previous state-of-the-art model (Kolitsas et al., 2018) in end-to-end EL in Micro-F1 score. They argue that this advancement comes from the modularity of their approach and the high performance of BERT on the MD subtask. The modularity allows for choosing the best system for each subtask. Thus, modularity is of particular interest to our work as it would allow our approach to be integrated into the CHOLAN architecture in future work.

Another task domain that is similar to our approach is entity typing (ET). In earlier years, NER systems often included a classification into a small number of what are essentially types. In this case, the task is also more specifically referred to as Named Entity Recognition and Classification (Nadeau and Sekine, 2007). Methods targeting wider sets of types are called fine-grained entity typing (e.g. Ling and Weld, 2012; Yogatama et al., 2015; Yaghoobzadeh and Schütze, 2015). Ling and Weld (2012) set a fine-grained entity

[7] https://www.wikidata.org/wiki/Q19879, retrieved 22.3.22

[8] initially used in information retrieval as a measure of how important a term is for a document

linking task based on 112 curated tags accompanied by a dataset that is automatically labeled through Wikipedia anchor links and associated Freebase[9] types. The tags are curated to trade-off granularity with increasingly noisy type labels. Examples from the selected tags are professions of famous persons like actors or artists and different types of locations like cities, countries, or mountains. Comparing those tags with our description, they are definitely coarser than ours and also more concise, consisting primarily of a single, short phrase.

A more recent work in entity typing is LUKE (Language Understanding with Knowledge-based Embeddings; Yamada et al., 2020). LUKE is based on the BERT architecture and extends RoBERTa's pretrained model. The focus of LUKE is on entity mentions and treats mentions as a single token each in addition to the usual representations of the input sequence. This allows the model to generate a single representation for each mention rather than several representations for each subtoken. These mention tokens are then trained with an entity-specific masked language model task. Furthermore, LUKE has an entity-aware self-attention mechanism in addition to the usual self-attention of BERT to deal with the new token type for entity mentions. For evaluation, the authors compare the embeddings produced by LUKE with those produced by RoBERTa as a baseline, as well as the scores reported by previous works. Yamada et al. report a state of the art performance on entity typing but for only nine entity types. While we did not consider LUKE during this thesis due to time constraints, it is definitely an interesting addition for future work as a substitute embedding model.

Going even finer in entity typing is "Ultra-Fine Entity Typing" (Choi et al., 2018). In contrast to the closed class of types the previously described fine grained entity typing works used, the goal of Choi et al. is to predict free-form noun phrases in an open class. Furthermore, multiple of those free-form types can be assigned to a single entity mention, e.g. in the following sentence the mention "Sam" might be labeled with types such as *person*, *accused*, *suspect* and *defendant*.

> Sam has been asked to appear in court to face the charge. (Choi et al., 2018)

Both differences to fine-grained ET together allow for easier crowdsourcing to construct a dataset because annotators do not have to agree on a single label and have more freedom. Implementation-wise, the authors extend the fine-grained typing model by Shimaoka et al. (2016) that builds a mention representation along with a context representation using a long short-term memory (LSTM) network with an attention mechanism. The authors improve the model's representations as well as its training through their own dataset. Additionally, they add learning of context-sensitive types to the model for cases where an entity can have different types in different contexts, e.g. famous persons can be known for different things. For example, Arnold Schwarzenegger is both an actor and former governor of California, and depending on the context, only the respective types should be

[9] a former knowledge graph whose data was integrated into Wikidata

produced. Ultra-fine grained entity typing as a task is the most similar to our approach. Both methods produce a free-form description given an entity in its context. The main differences lie in the supervision and thus required labeled dataset of Choi et al. (2018) while our approach is unsupervised and can be trained on any kind of corpus and KB.

3. Methodology

We propose an unsupervised approach to entity disambiguation. It builds upon a heuristic on the unambiguity of entity mentions and clustering of VSRs. Throughout this chapter, we will introduce our approach, starting with an overview. First, we introduce our heuristic on unambiguity, applying it to extract unambiguous mentions from a corpus. Then, we build representations of these unambiguous entity mentions. Based on those representations, the entity mentions are clustered. As the last step of our training, we generate descriptions for each cluster, leveraging the semantic similarity of the clustered mentions. For inference, we assign unknown entity mentions to the clusters and describe the mention with its cluster's description.

3.1. General Methodology

We base our methodology on a set of unambiguous entity mentions. The unambiguity is determined by querying the Wikidata KB with the mention's surface form. Only mentions with a single hit in the query's answer are considered unambiguous and used in subsequent methodology steps. We leverage the unambiguity later on so that there is only a single possible entity a mention can refer to. This allows us to train an unsupervised model on any dataset. On a plaintext corpus, we first find entity mentions, e.g. using an NER tagger. From the mentions found, we select those we deem unambiguous (more on unambiguity in Section 3.2). Then, we build semantically related groups of unambiguous mentions and the entities they refer to from which we draw knowledge. The semantic relatedness of mentions is calculated as the similarity of the respective mentions' VSR. The semantically related groups are then found by clustering the representations. Last, we generate descriptions for each group which are used during inference to describe ambiguous mentions. Our goal is that the description represents its group and generalizes from all entities in that group. To achieve that goal, we introduce and compare different strategies for both clustering and description generation. The training process can be summarized in five steps:

1. find mention candidates
2. select unambiguous mentions
3. build vector representations for unambiguous mentions
4. cluster representations

5. generate descriptions

With our model complete, we look at inference and testing. For an unknown and potentially ambiguous mention, we synthesize a text that describes the entity mentioned based on the entity descriptions learned during training. The goal, again, is to produce descriptions that incorporate important features of the described entity. Our model produces a set of descriptions based on the trained clusters that are most similar to the unknown mention. The similarity is based on vector representations of the unknown mention and of the mentions that constitute the clusters in the same way that the training mentions are clustered.

3.2. Unambiguous Entity Mentions

We base our work on a set of entity mentions that we consider to be unambiguous. The unambiguity allows us to link those mentions to entities in a KB without relying on a labeled dataset. Having established a link, we request additional information about the entity from the KB. Furthermore, we derive from the unambiguity that all mentions of the same surface form, i.e. a mention consisting of the same tokens, refer to the same entity and thus can be treated as such. In our experiments, we leverage this derivation to merge the representations of mentions with the intention of combining the knowledge inherent to the representations.

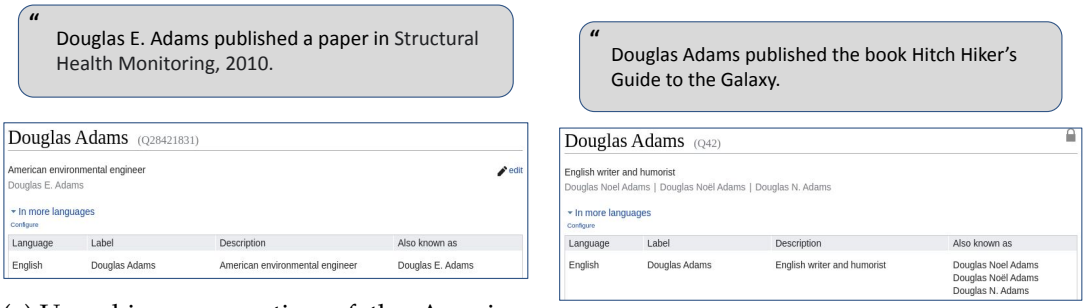
For this thesis, the unambiguity of a mention is determined by querying Wikidata for the mention's surface form. While the methodology can be transferred to any other KB, we opted for Wikidata because of its structured and queriable storage of entity names and aliases and thus potential surface forms of mentions. When querying for potential entities, we only consider exact matches of the surface form with an entity's name and aliases (shown as "Also known as" on Wikidata's website). Here, exact matches mean that we ignore results with similar names that search engines might return. Taking the Wikidata item of Douglas Adams[1] as an example, we consider the following names as surface forms:

- Douglas Adams
- Douglas Noel Adams
- Douglas Noël Adams
- Douglas N. Adams

We consider mentions as unambiguous for whose surface form Wikidata returns exactly one result. If there are multiple matches, the mention is ambiguous. If there are no matches at all, we ignore the mention as well because we cannot draw knowledge from

[1] https://www.wikidata.org/wiki/Q42, relevant part captured in Figure 3.1b

the mention without a Wikidata entity. In the example sentences in Figure 3.1, we recognize the mention "Douglas E. Adams" as unambiguous and that of "Douglas Adams" as ambiguous. While "Douglas Adams" is the name of the Wikidata entries for both Douglas N. Adams and Douglas E. Adams, "Douglas E. Adams" refers to only the one Wikidata entry. One issue arises for our unambiguity decision from the OWA. In the context of this thesis, the OWA states that a knowledge base may always be incomplete (Hitzler et al., 2010). We acknowledge that, following the OWA, there can be other entities that would match our queries but that are not (yet) part of the KB. However, we hypothesize that we find enough actually unambiguous entity mentions in our training data to be of value to our approach. Furthermore, we believe that the impact of mentions being assigned to wrong KB entities is negligible for the scope of this thesis. Thus, we heuristically consider Wikidata as a closed world for the purposes of this thesis.



(a) Unambiguous mention of the American environmental engineer Douglas Edward Adams

(b) Ambiguous mention of the English writer and humorist Douglas Noël Adams

Figure 3.1.: Two sentences showing the unambiguity heuristic using the example of two persons called Douglas Adams along with excerpts of the persons' Wikidata items. Screenshots retrieved on 21.3.2022.

One of the main selling points of our approach is its unsupervised training. To leverage this capability, it needs to handle plain text without annotations found on many datasets in NER or EL tasks. In this case, we first recognize entity mentions in a corpus. There have been numerous methodologies and implementations of NER for years (J Li et al., 2022). Newer implementations are based on the transformer technology. We use spaCy's[2] transformer pipeline that NER tagging is a part of. It recognizes entity mentions and labels the tokens accordingly. Those labels let us calculate which tokens belong to a mention and the mention's surface form. From the mentions found in a training corpus, we select those with an unambiguous surface form as described above and proceed with this set through the rest of our methodology.

[2]https://spacy.io/usage/linguistic-features#entity-linking

3.3. Entity Representations

After building a set of unambiguous entity mentions in Section 3.2, we continue by computing VSRs for those mentions. We introduced VSR and the three models in Section 2.2. Here, we reiterate the facts that are most important to our methodology. VSRs are a numerical representation of units of text like words, sentences and documents (Vajjala et al., 2020). The VSRs used in this thesis place words in a continuous, low dimensional space and are designed to convey a grasp of the words' semantics. While there are different kinds of vector space representations for words, we focus on CWEs provided by BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) and further include static embeddings for comparison.

Static embedding models like FastText (Bojanowski et al., 2017) build vector space representations that are static in regard to the surface form of a word. Generally, they cannot discern between polysemes, i.e. words with the same spelling but different senses. Instead, they produce the same vector for each surface form regardless of the sense of the word. Static embeddings for a word are trained on the contexts of the word in a training corpus but treat all occurrences of a surface form as belonging to the same sense. Furthermore, at inference, the prominent static embedding model Word2Vec (Mikolov et al., 2013), among others, cannot handle OOV words, i.e. words that do not occur in its training corpus. Hence, for building static embeddings, we choose FastText, which splits words into chunks similar to syllables. This allows FastText to build static embeddings for entity mentions whose surface form is not part of a training corpus in many cases. In these cases, the word's chunks can be used that are more likely to occur in the training data.

In contrast to static embeddings, CWE models like BERT produce a vector that is dependent on the context in which a word occurs. Thus, CWEs distinguish between different senses of a surface form (Wiedemann et al., 2019), because the sense of an ambiguous word depends on its context (Harris, 1954). The context-dependency also leads to different embeddings of multiple occurrences of a specific word, even with the same sense. For our work, we compare the CWEs from two models: BERT and RoBERTa. BERT, proposed by Devlin et al. (2019), is an end-to-end language model that is build on a self-attention based transformer architecture (Vaswani et al., 2017). To deal with OOV words, BERT employs WordPiece (Wu et al., 2016). WordPiece is a tokenizer originally built for languages like Korean or Japanese, which have large vocabularies and do not allow for tokenization as Latin-based languages do. When passing a sequence to BERT, it is tokenized by WordPiece, including OOV words being split into subtokens. Subtokens following the first are prefixed with `##` to denote which tokens are the result of a split. For example, the word `Hitchhiker` is OOV and therefore is split into the known subtokens: `Hit`, `##ch`, `##hi` and `##ker`. From the two tasks that BERT is trained on, mainly the masked language model task contributes to producing CWEs that allow distinguishing word senses (Wiedemann et al., 2019). Originally, BERT, being a pretrained model, was intended to be

used in a fine-tuning setup for downstream tasks. Instead, we directly use the pretrained model to build CWEs without further fine-tuning the model. RoBERTa (Liu et al., 2019) is introduced as a replication study of BERT with the goal to improve its performance through further hyperparameter tuning and a larger training dataset. By experimenting with a second CWE model, we investigate the influence of the embedding model on our approach.

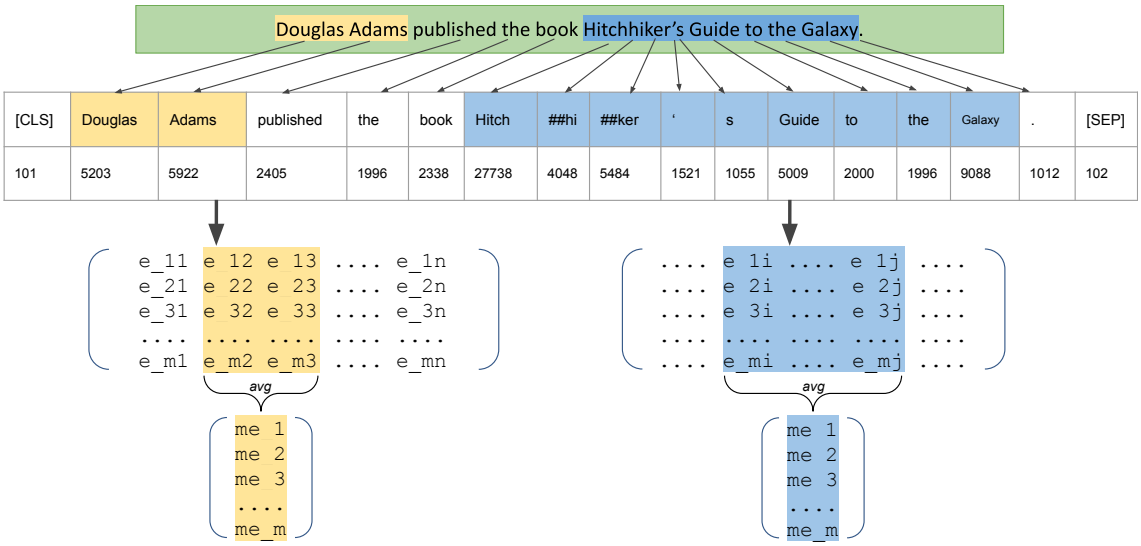


Figure 3.2.: Mention Embeddings: building a single vector for each mention. The sentence is tokenized and encoded. The model then generates embeddings that are averaged over all tokens of each mention.

For each unambiguous mention, we parse its sentence from the training corpus and let the VSR model produce embeddings. While the FastText implementation outputs one vector per word in the text input, the implementations of BERT and RoBERTa do not automatically merge the subtoken vectors into word vectors. However, entity mentions often consist of multiple words. For these cases, we merge the embeddings of the words to obtain mention embeddings. Thus, we select the embeddings for all words of the mention and their subtokens. Those are averaged, yielding a single VSR for each mention. While averaging BERT embeddings generally does not yield very good results for building sentence representations (Reimers and Gurevych, 2019), we argue that averaging only a few tokens at a time reduces its impact on the representations. Further, there are simple improvements like those proposed by B Li et al. (2020) to be considered in future work. Our embedding process at the example of BERT is displayed in Figure 3.2. First, BERT's tokenizer splits the words into (subword) tokens and translates those into training vocabulary-based integers. In the next step, these are the inputs for the BERT model, which returns an embedding vector for each (subword) token. We then calculate the indices of the tokens that compose a mention and average the embedding vectors at those indexes. This yields a single vector that we use as the mention's representation in the following steps of our methodology.

3.4. Clustering of Contextual Word Embeddings

Ultimately, our goal is to draw knowledge from the similarity of entity mentions through their CWEs. To do so, we employ clustering algorithms yielding groups of mentions that we expect to be semantically related through the CWE similarity. We experiment with two clustering methods:

- graph clustering (`GC`)
- hierarchical clustering (`HC`)

The former requires us to think about how to construct a graph that allows the clustering algorithm to perform well. The latter's usage is more straightforward than the former, but its hyperparameters need testing and optimization.

3.4.1. Graph Construction

For graph clustering, we first build a graph that is a well-designed representation of the training data. It needs to properly incorporate the semantic relationships between the entity mentions that serve as the nodes. However, we cannot just construct a complete graph with an edge between all pairs of mentions. This would lead to computational problems, especially for larger datasets. Furthermore, a sparse graph can help the clustering algorithm to focus on the more important connections. In our experiments, we compare two modes for the graph construction:

- mention level (`M`): each mention is mapped to a seperate node;
- surface form merged (`SF`): each node represents a surface form and therefore might combine data from multiple mentions.

The graph clustering algorithm that we use, Chinese Whispers (CW; Biemann, 2006), requires us to build an undirected, weighted graph We introduce CW in Section 2.1 and describe its usage in this thesis in Section 3.4.2. The nodes of the graph comprise the unambiguous mentions (multiple with the same surface form in `SF`) and the edges represent the similarity between those mentions. The edge weights hold the similarity between the respective mentions. Thus, the numeric similarity can be taken into account by the clustering algorithm.

In both modes, `M` and `SF`, each mention contributes three edges to the most similar mentions. The number of edges needs to be limited to keep the graph sparse for the clustering to perform well. Building three edges is a heuristic that we do not delve deeper into because it worked well from the start and we leave the optimization for future work. In `SF`, we merge all mentions with the same surface form into a single node. This can add more than three edges to a node because each mention still generates three edges. However, this might also cause collisions when multiple mentions with the same surface

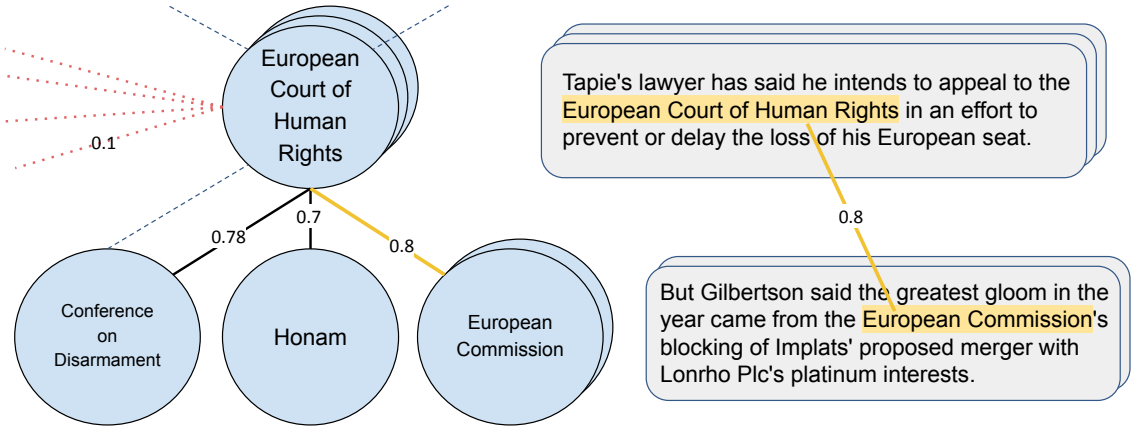


Figure 3.3.: Graph Construction: The mention "European Court of Human Rights" in the example sentence with edges to the three most similar mentions. One of the three is highlighted in yellow with the corresponding sentence on the right. Edges with lower similarity are indicated in red, and incoming edges in blue. Arrows indicate the direction for which mention generated the edge and not an edge direction

```
def auto_cosine_sim(embeddings):
    normalized_embeddings = normalize(embeddings)
    return mm(
        normalized_embeddings,
        normalized_embeddings.transpose(0,1)
    )
```

Listing 3.1: Auto cosine similarity for a list of vectors (based on implementation by Reimers and Gurevych, 2019)

form have the same other mention in their top similarities. We resolve collisions by always using the maximum similarity of the colliding mentions pairs as edge weight. Pairs of mentions with the same surface form are ignored in `SF`, i.e. we do not build reflexive edges. An example node of the graph constructed in `SF` can be seen in Figure 3.3. The surface form "European Court of Human Rights" (ECHR) appeared multiple times in the training data and for one of the appearances, the three most similar mentions are "Conference on Disarmament", "Honam" and "European Commission" (EC). The similarity between ECHR and EC in the two example sentences with 0.8 is the top similarity for ECHR, and thus, the edge is actually constructed in the graph. Edges with lower similarity than the three mentioned above are omitted in this example. Additional appearances of ECHR might lead to further edges being constructed or the shown edges being overwritten.

As the similarity measure we choose the cosine similarity defined as:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{3.1}$$

In practice, we use a simplified computation that uses matrix multiplication because we

calculate the similarities between all pairs of embedding vectors. This can be seen in Listing 3.1 using matrix multiplication of the normalized vectors with its transpose. For example, 100 mentions produce 100 vectors and for BERT each embedding vector is of length 768 resulting in a 100×768 matrix. Multiplying that matrix with its transpose yields a 100×100 matrix with element $[i, j]$ denoting the similarity between mention i and mention j. This similarity matrix is an easy way to compute all similarities needed to construct the graph in a single computation. However, it grows quadratically with the number of unambiguous mentions in the training data. This becomes a problem quickly when working with larger datasets such as text extracts from Wikipedia. The first problem is the memory consumption of the matrix, which we circumvented with batching. With batching, we reduce the size of the matrix for n mentions from $n \times n$ to $n \times batch\_size$. Even with batches, the computation is another problem. Once we have computed the matrix for a batch of the data, we add edges to the graph for the top three similarities of each mention. For that, we sort n vectors of n length, each sorting of one vector taking $O(n \cdot log(n))$ time. Then, we add an edge between the respective edges for each mention, completing the graph.

3.4.2. Graph Clustering

For the graph clustering method, we employ Chinese Whispers (CW; Biemann, 2006). CW is a graph clustering algorithm specifically designed with NLP in mind. It works in a bottom-up manner by initializing each node of the graph with a distinct class. The algorithm then iterates until no node changed its class during the previous iteration. Each iteration updates all nodes in a random order with the predominant class among the node's neighbors. The update calculation also considers the edge weight for the class of the respective neighbor. We describe CW in more depth in Section 2.1.

To solve clustering specifically in NLP, CW has a low computational complexity that is needed to cluster large graphs within a reasonable time. Other work (Di Marco and Navigli, 2013) shows that it performs well in tasks that deal with word senses which relates to entity disambiguation. Furthermore, the algorithm is, in general, parameter-free, meaning that we do not need to supply a target number of clusters or similarity thresholds. However, we set an upper limit on the iterations performed, which showed a more promising distribution of cluster sizes in preliminary experiments. Clustering the graph with CW yields a partitional clustering directly that we use as semantically related groups of entity mentions.

3.4.3. Hierarchical Clustering

Hierarchical clustering (HC) is another category of clustering algorithms (Maimon and Rokach, 2005). In contrast to partitioning clustering algorithms like CW, HC algorithms produce a structure of nested partitions. We show an example structure from our experiments in Figure 3.4. There are two variants:

- agglomerative: bottom-up, each mention starts in its own cluster and clusters are merged iteratively;
- divisive: top-down, all mentions start in one cluster that is split consecutively.

We use the agglomerative variant, which works in a bottom-up manner. It starts with all data points in a singleton cluster, i.e. clusters containing only a single element. Then, the most similar clusters are iteratively merged until a single cluster is left. The structure can be broken at any point to yield the final clusters. This point can be defined by a hyperparameter such as the desired cluster count or a similarity threshold. For our experiments, we use the implementation in scikit-learn[3]. This implementation offers a number of affinity metrics, linkage algorithms, and the option to either supply a distance threshold or a number of clusters to produce. We explore and optimize these hyperparameters in Table A.1 and use the best scoring methods for our experiments. The affinity defines the similarity measure that is used. Examples that are implemented by scikit-learn are euclidean, cosine, or manhattan distances. The linkage algorithms (Jain and Dubes, 1988) differ in the way they compute the distances for non-singleton clusters based on all data points in the clusters. The *single-link* linkage uses the minimum distance of all possible pairs of points from each cluster. In contrast, *complete-link* uses that pair of points with the maximum distance yielding more compact clusters and dealing better with noisy data than single-link (Jain and Dubes, 1988). Further, scikit-learn's implementation offers *average* linkage calculating the average of all distances from two clusters and *ward* which minimizes the variance of the merged clusters. The method of how to separate the hierarchical structure into partitions for comparability is of special interest. The structure can either be split by some similarity threshold or by a defined number of clusters. The similarity threshold defines a lower bound under which nodes are not merged. Alternatively, the algorithm tries to satisfy the defined number of clusters.

3.5. Describing Mention Clusters

We now delve into generating descriptions for each cluster. Later, these descriptions are used for inference on unknown, possible ambiguous mentions when those are assigned to clusters. First, we generate descriptions for the clustered training mentions. Each cluster consists of a set of unambiguous mentions to which we already assigned a Wikidata item. From those Wikidata items, we now retrieve their respective descriptions that are often just a few words briefly outlining their subject. For example, the Douglas N. Adams[4] mentioned above is described with "English writer and humorist" which focuses on the person's profession and further mentions the nationality. This combination is also a recurring pattern for other persons on Wikidata. In the following, we introduce

[3]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering
[4]https://www.wikidata.org/wiki/Q42, viewed 15.12.2021

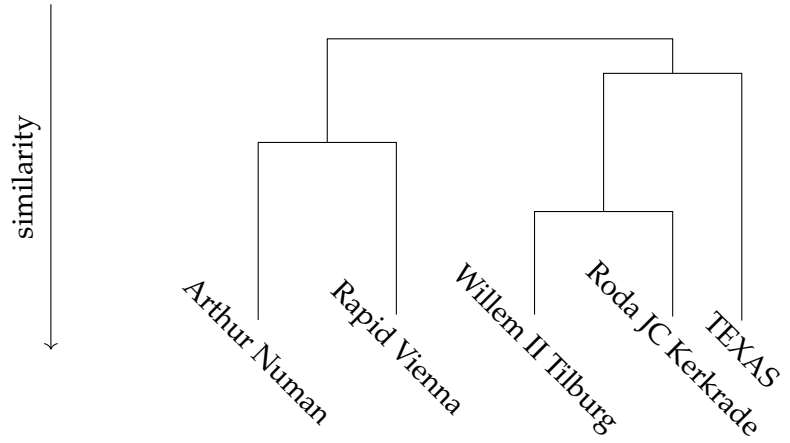


Figure 3.4.: Dendogram of hierarchical clustered mentions revolving around association football with the exception of "TEXAS" which refers to an Major League Baseball (MLB) team

three approaches to generating descriptions for clusters which we use for disambiguation in the last section of this chapter.

3.5.1. Statistical Language Model

The first technique generates descriptions using statistical LMs. In general, LMs represent a language understanding that can be used to either generate new sequences in the learned language or recognize whether a given string belongs to a language (Manning et al., 2010). A statistical LM is trained by calculating the probabilities of word occurrences. The probability of a word can be just the count of the word's occurrences over the total count of all words, but there are numerous smoothing approaches to address the inherently sparse distribution of word frequencies. The probability calculation can involve any number of preceding words, e.g.

- a unigram model computes the probability of a word occurring, i.e. how often a word occurs in the training data regardless of the context;
- a bigram model calculates the probability of a word occurring depending on the previous word;
- a trigram model includes the two previous words in the calculation.

In our application, we train a separate language model for each cluster based on the cluster's Wikidata descriptions. Therefor, the Wikidata descriptions are tokenized and transformed into uni-, bi- and trigrams with padding tokens at the beginnings and endings. The probabilities are then calculated from the n-gram counts while applying Kneser-Ney (Kneser and Ney, 1995) smoothing. Once the probabilities are calculated, they can be used to generate new sequences based on the seen training data and thus generalize over descriptions of a cluster's mentions.

3.5.2. Tf-idf

The second technique generates descriptions using tf-ids scored n-grams. Tf-idf is a scoring method developed for information retrieval (Manning et al., 2010). The score for a given term in a given document is high for terms that frequently occur in the given document but rarely across the dataset. In contrast, words that appear in many documents are scored lower. Calculation-wise, it is the product of two parts: tf and idf.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

The term frequency (tf) describes a relevance signal by how often a term t occurs in a document d. The document frequency (df) describes a discriminating power by how many documents a term t occurs in. To penalize terms that offer no discrimination between documents, an inverse of df is used and defined as follows with N being the total number of documents:

$$\text{idf}_t = \log \frac{N}{\text{df}_t}$$

In our case, we consider the clusters as documents and n-grams as the terms and use the top scored n-grams as the cluster description. Again, from each cluster, we use the Wikidata descriptions to produce uni-, bi-, and trigrams. We argue that calculating the tf-idf scores yields the n-grams that are most relevant to the cluster in the context of all clusters, similar to how tf-idf works in information retrieval. Based on that assumption, each cluster's top scored n-grams describe the cluster as a whole.

3.5.3. Log Likelihood

The third technique works similarly to tf-idf but uses the log likelihood ratio (LLR; Dunning, 1993) measure for scoring. LLR is a statistical significance measure that used in some NLP research (e.g. Moore, 2004; Remus, 2012). Originally, it is used to measure the significance of two words co-occurring in bi-grams and for its ability to work with rare events. This rarity stems from the very low frequency with which most words occur in English, as described by Zipf's Law (Zipf, 1936). However, LLR can be used for the occurrence of any two events. We measure the co-occurrence of an n-gram X with a cluster Y. More specifically, X is n-gram in the description of some entity that is part of cluster Y. Table 3.1 shows a contingency table of the co-occurrences that we count in the clustered training data. Based on the event occurrence counts, we use the LLR score formula shown in Equation 1 to calculate scores for the n-grams in each cluster. As in the tf-idf technique in Section 3.5.2, we now calculate the top-scored n-grams for each cluster that serves as a description of the cluster as a whole.

3.6. Entity Assignment

Once we have generated descriptions for the clusters with the previous methods, we describe new entities. While the previous sections described the training procedure of our

$$
\begin{aligned}
LLR &= -2\log(\lambda) \\
&= 2 * \big[n\log(n) - n_a\log(n_a) - n_b\log(n_b) + n_{ab}\log(n_{ab}) + n_{\overline{A}\overline{B}}\log(n_{\overline{A}\overline{B}}) + n_{\overline{A}B}\log(n_{\overline{A}B}) \\
&\qquad + n_{A\overline{B}}\log(n_{A\overline{B}}) - n_{\overline{A}}\log(n_{\overline{A}}) - n_{\overline{B}}\log(n_{\overline{B}})\big]
\end{aligned}
$$

Equation 1: LLR formula (Dunning, 1993) (as written by Remus (2012))

	A: occurrence of n-gram X	$\overline{A}$: non-occurrence of n-gram X	$\sum$
B: occurrence in cluster Y	n_{AB}: n-gram X occurs in cluster Y	$n_{\overline{A}B}$: n-grams in cluster Y but n-gram X	n_B: all occurrences in cluster Y
$\overline{B}$: occurrence not in cluster Y	$n_{A\overline{B}}$: n-gram X occurs in clusters but Y	$n_{\overline{A}\overline{B}}$: n-grams but X that occur in clusters but Y	$n_{\overline{B}}$: all occurrences in all clusters but Y
$\sum$	n_A: all occurrences of n-gram X	$n_{\overline{A}}$: all occurrences of all n-grams but X	n: all occurrences

Table 3.1.: LLR contingency table: co-occurrence of an n-gram X with a cluster Y. n_{mn} corresponds to the n_{mn} in Equation 1.

methodology, we now introduce the method for inference. In an input sequence, e.g. one or more sentences, we recognize any entity mentions and produce descriptions for all of them. The sequence and the mentioned entities can be entirely unknown from the training. While unknown and known mentions are treated the same during inference, for this thesis, a further heuristic might be to match mentions to the surface form encountered during training directly. However, we argue that the effect of this heuristic is negligible for the purpose of this thesis.

For finding entity mentions in the input sequence and building their representations, we reiterate the same methods as described in Section 3.2 and Section 3.3. To be more specific, we find entity mentions using an NER tool and then calculate VSR for them. After that, we select clusters that are similar to the mentions in regard to the mention's VSR. We continue to use cosine similarity as the similarity metric. There are two possibilities for the representation of a cluster when calculating the similarity to an unknown mention:

- a cluster center based on the VSRs of all mentions in a cluster;
- best similarity of all individual VSRs of a cluster.

In both cases, the cluster with the best score is assigned to the new mention. We implemented the cluster center variant because they can be precomputed at training time, while the other variant requires calculating more scores during inference. Furthermore,

we chose to compute the mean of each cluster's VSRs to serve as the respective cluster's center. This averaging follows our method of computing the entity representations (Section 3.3). The cosine similarity between a new mention and all clusters can be calculated using the same technique used in Section 3.4.1 and Listing 3.1. This yields a list of similarities from which we select the best.

With a cluster assigned to the new, unknown entity, we now draw knowledge from that cluster. We base our experiments on the descriptions generated for all clusters in Section 3.5. Below, we exemplify the results of assigning an entity mention to a cluster and using the cluster's description in the following sentence.

> The European Commission said on Thursday it disagreed with German advice to consumers to shun British lamb until scientists determine whether mad cow disease can be transmitted to sheep.

There are four mentions of named entities in the sentence: European Commission, Thursday[5], German and British. Our model, as introduced throughout this chapter, describes the mention "European Commission" with "executive branch of the European" because that is the highest scored n-gram for the most similar cluster.

[5] some definitions of named entities do not include weekdays

4. Evaluation

This chapter is concerned with the experiments around our methodology and their evaluation. First, we give an insight into the data on which the experiments are based. Second, we look at the unambiguous mentions we found based on the previously introduced heuristic. Then, we investigate the graph constructed and the clusters found during training, as well as the cluster assignments of the mentions during inference. Last, we evaluate experiments based on the descriptions introduced in Section 3.5. Unless otherwise stated, references to data in this chapter refer to the AIDA dataset presented in Section 4.1.1 or the models based on it.

4.1. Data Analysis

Throughout this thesis, we train our models on two datasets: AIDA CoNLL-YAGO and one based on Wikipedia pages. We introduce the AIDA corpus that we use for both training and evaluation of our models. We build a second corpus based on Wikipedia pages to show that our approach is trainable on plaintext corpora and that its performance scales with the size of the training data.

4.1.1. AIDA CoNLL-YAGO Corpus

The primary corpus used for this thesis is the AIDA CoNLL-YAGO dataset provided alongside Hoffart et al. (2011). Unless specified otherwise, the experiments and analytics following this section are based on the AIDA dataset. It is based on the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). The CoNLL task itself focuses on NER and classification into one of four categories: persons, organizations, locations, and miscellaneous names. The dataset for the task is built upon the Reuters Corpus for English and the ECI Multilingual Text Corpus for German. The English part, that we use, consists of 1393 news articles, over 22 000 sentences and 300 000 tokens and is annotated with 35 000 named entities. Each token is annotated with its part-of-speech (POS) tag, its syntax parsing chunk tag and an entity tag, consisting of an IOB (inside, outside, beginning) identifier and the category for tokens that are entity mentions. While tokenization, POS tagging and chunking are automated, the entity categories have been annotated manually. As another part of preprocessing, the data is also split sentence-wise into roughly 68% training set, 16% development set and 16% test set.

While the CoNLL dataset targets NER, AIDA extends the data for Entity Linking. It

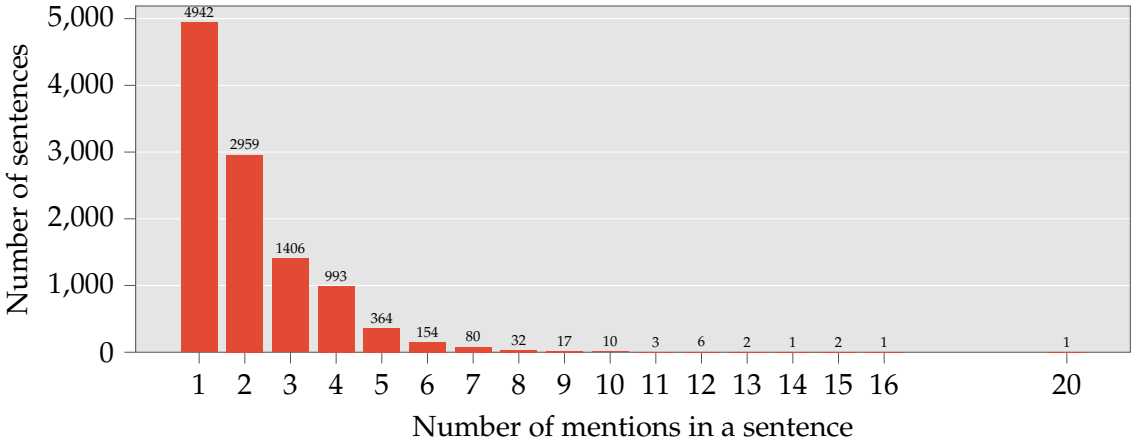


Figure 4.1.: Count of sentences by number of mentions in the AIDA training set

provides assignments for a portion of the entity mentions in the English CoNLL dataset to YAGO2, Wikipedia, and Freebase[1] entities. Of the 34 450 entity mentions in CoNLL, AIDA provides Wikipedia IDs for 27 608. For the training part of AIDA, there are about 23 000 detected entity mentions. The development set, that we use for evaluation, has 8500 mentions of which 6400 are labeled with a Wikipedia page. Sentences may, of course, mention multiple entities; the distribution of the count of mentions per sentence is shown in Figure 4.1. This insight leads to the realization that mere sentence embeddings are not specific enough because multiple mentions in a sentence would be represented by the same embeddings. Representing two semantically different entities by the same embeddings would increase the similarity between those two entities. While it is unlikely that completely different entities are mentioned in the same sentence, it would still distort their similarity. Because of this, we use mention-level embeddings as presented in Section 3.3.

During the training of our model, we only use the tokens and NER tags. Instead of the Wikipedia or YAGO2 annotations, we directly learn from Wikidata, which allows us to extend the learning data by any available texts. We use the Wikipedia ID annotations for evaluation only.

As mentioned before, AIDA is built upon the Reuters Corpus. The Reuters corpus consists of news stories that were published from August 1996 to August 1977. Experimenting with our methodology gives us an insight into that corpus. One issue arises from sequences in AIDA that are not proper English sentences. For example, part of AIDA are end results of sport-events like “Ajax Amsterdam 1 AZ Alkmaar 0”. While there is enough context for transformers to produce embeddings, it is not trivial whether it affects the similarity of those embeddings to those of proper sentences with mentions of the same entity.

[1] a former knowledge graph whose data was integrated into Wikidata

4.1.2. Wikipedia Corpus

As a second corpus, we use the texts of Wikipedia pages. The corpus is based on a 2017 Wikipedia dump which consists of texts of 5 490 664 Wikipedia pages. While the AIDA corpus is already tagged with NER tags, the Wikipedia dump is only pure text. To compensate, we use basic NER tagging provided by spaCy[2]. The tagged mentions are then processed as described in Section 3.2 to select the unambiguous mentions. We build two sets of different sizes to investigate the performance gain on a larger training set. The first set consists of 20 000 unambiguous mentions occuring on random Wikipedia pages and the other adds another 10 000 resulting in 30 000, including the 20 000 of the first. Table 4.1 reports basic statistics of both datasets. The *Surface Forms* column describes how many unique surface forms occur in the respective dataset and Wikidata (*WD*) *Links* to how many unique entities they are linked to. The small difference means that only a few different surface forms are linked to the same entity.

unamb. Mentions	Surface Forms	WD Links	Source Pages	Sentences	Tokens
20 000	9256	9158	769	34 778	598 978
30 000	12 931	12 782	1058	59 514	977 760

Table 4.1.: Statistics of the Wikipedia datasets. Both are limited to 20k and 30k unambiguous mentions to keep the computational effort within limits. Surface forms and Wikidata (WD) links count the unique values each.

4.2. Analysis of Mention Unambiguity

In the AIDA dataset, we recognize the mention "St. Louis" as unambiguous and link it to the item for the city in Missouri, United States[3]. However, we demonstrate that it is ambiguous and thereby exemplifies a false-positive resulting from the heuristic. The surface form "St. Louis" appears seven times in AIDA's training set, one of those appears in this sentence:

> In **St. Louis**, Gary Sheffield and Devon White each drove in two runs and Mark Hutton scattered four hits over six innings to lead the Florida Marlins past the St. Louis Cardinals.

The mention above and one other are labeled in AIDA with St. Louis, Missouri[4] while five of the seven mentions are not labeled. The sentence already shows an additional entity that could be referred to by the surface form St. Louis: the baseball team St. Louis Cardinals. Referring to teams just by the city they are based in is common in sports contexts. This phenomenon is frequent in European football, where many clubs are abbreviated

[2]https://spacy.io/usage/linguistic-features#entity-linking
[3]https://www.wikidata.org/wiki/Q38022
[4]http://en.wikipedia.org/wiki/St._Louis,_Missouri

through their city names. There are even some cases, like Hamburg, where two clubs play on the same level (FC St. Pauli[5] and HSV[6]) but only one of them is usually meant by mentioning the city name (HSV in the case of Hamburg). In the St. Louis example, there are even more entities that are possible candidates. The city's county[7] is also called St. Louis, as is a county in the US state Minnesota[8], cities in Michigan[9] and Oklahoma[10], teams in other sports[11] than the mentioned baseball team and more[12]. According to the OWA, there can be even more entities that are not yet documented in both Wikidata and Wikipedia. The effect of OWA is also shown by this example because we recognize St. Louis as unambiguous based on an outdated Wikidata dump, while based on the online Wikidata the heuristic deems it ambiguous. Thus, additional St. Louis entities were added to Wikidata in the meantime. However, we linked to the correct entity in this instance, although we were working on incomplete data. Intuitively, a more famous entity is both more likely to be present on a KB as well as more likely to be mentioned in texts. While this is just a heuristic, it supports our hypothesis on unambiguity.

Another mention that we recognize as unambiguous is "O'Neal" in the sentence:

> "The good news is, if you're highly skilled and have many abilities, you'll be paid more", said **O'Neal**.

From this sentence alone, humans cannot easily guess the correct entity referred to by the mention. However, it is clear to a human that O'Neal must be a person. Looking a few sentences behind in the corpus, we see that the last name O'Neal is used as an abbreviation after introducing the person in:

> Presently, for example, if an accountant's job involves doing five specific tasks, he or she can expect a certain salary, said Sandra **O'Neal**, a Towers Perrin principal.

Unfortunately, we link the mention to a settlement[13] in Virginia, USA. The mention also should not be considered unambiguous because it is a common last name in the USA and there are multiple famous persons with that name.

However, there are also mentions of persons whose names might be ambiguous considering the whole world population but unambiguous when only considering famous persons. We limit the scope for our unambiguity heuristic to famous persons because we work on a publicly accessible KB and thus, the scope of persons described by the KB is limited. This might be different in other applications, e.g. when working on private KBs.

[5] https://en.wikipedia.org/wiki/FC_St._Pauli
[6] https://en.wikipedia.org/wiki/Hamburger_SV
[7] https://www.wikidata.org/wiki/Q498034
[8] https://www.wikidata.org/wiki/Q111549
[9] https://www.wikidata.org/wiki/Q3046589
[10] https://www.wikidata.org/wiki/Q3135108
[11] https://www.wikidata.org/wiki/Q207735
[12] https://en.wikipedia.org/wiki/Saint_Louis
[13] https://www.wikidata.org/wiki/Q7071880

An example in AIDA of such famous persons is Steffen Freund, a German football player, in the sentence:

> He will, however, have to do without the Dortmund trio of libero Matthias Sammer, midfielder **Steffen Freund** and defender Rene Schneider, who were all formally nominated despite being injured.

Both his first and last names are somewhat common in Germany, so the combination should be ambiguous in general. However, taking the above consideration into account, it is unlikely that there are other famous people called Steffen Freund aside from the football player that are not yet part of the KB. Thus, we consider the mention of Steffen Freund as unambiguous within the scope of the KB.

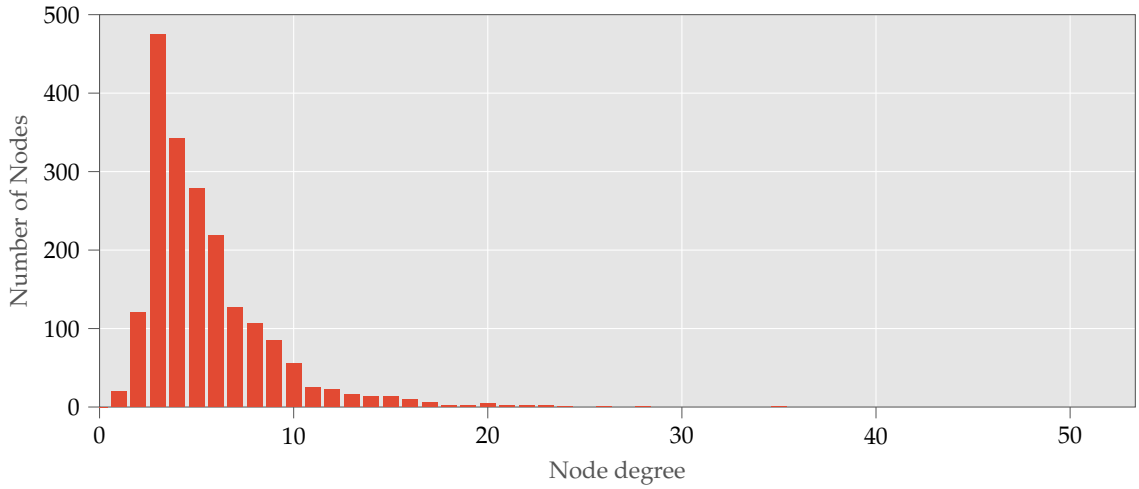
4.3. Graph & Cluster Analysis

This section analyzes the graph and the clusters produced by our approach as described in Section 3.4. We aim for a semantically consistent clustering that allows us to draw conclusions when assigning unknown mentions to the clusters. Furthermore, we look at the assignments of clusters when applying the AIDA dev-set in inference. For the duration of this section, we focus on the `SFGC-bert` (surface form merged, graph clustered, using BERT embeddings) mode, i.e. the mode for which we merge mentions on their surface form into a single node and use BERT as the VSR model. Additionally, we occasionally make comparisons to the other models as well.

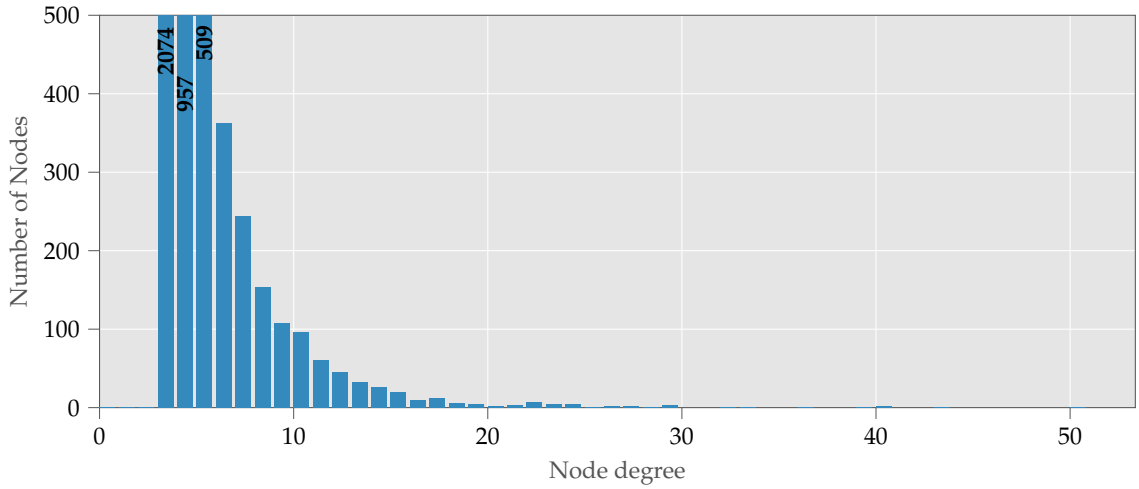
4.3.1. Graph Properties

We construct a graph as presented in Section 3.4.1 based on the AIDA dataset that we introduced in Section 4.1.1. As required by the employed graph clustering algorithm, Chinese Whispers (CW; Biemann, 2006), the graph is undirected and weighted. Furthermore, we keep the graph sparse so that CW performs well. Table 4.2 shows that the `SF` mode constructs a smaller graph than the `M` mode. The `SF` graph has less than half the nodes as well as edges of the `M` graph. Despite that, the average node degree suggests that `M` is slightly sparser. We display the distribution of node degrees in Figure 4.2 showing that most nodes have only a few edges in both graphs. For each mention, we construct edges to the three mentions with the highest similarity so that the peaks of both graphs in Figure 4.2 are at three. Nodes with a lower degree appear only in the `SF` mode, whereas in the `M` mode, all nodes have at least degree three. Nodes with a lower degree than three have multiple mentions with the same surface form in their top three. In this case, we use the highest similarity as edge weight. An example of a node with only two edges is *Ernie Els* which is connected to the nodes *Frankie Fredericks* and *Wayne Westner* because the first and third best similarities are to *Frankie Fredericks*. In `SF`, surface forms that occur

multiple times in the training data lead to nodes with a higher degree than three. Additionally, in both modes, mentions can get additional edges when they happen to be in the top three similarities of other mentions. For example, *John Veldman* occurs four times and the respective node has degree 23. The four mentions contribute 12 edges and the rest are constructed by other mentions having *John Veldman* in their top three.



(a) `SFGC-bert` mode



(b) `MGC-bert` mode

Figure 4.2.: Number of nodes with a certain degree in comparison for `SF` and `M` modes.

Incidentally, the `SF` graph consists of two components. The smaller only contains three surface forms: Ginebra San Miguel, Philippine Basketball Association, and Philippine Basketball Association. In the other component, all the other surface forms' nodes are connected. The small component's surface forms have multiple mentions that take up the top three similarities for each other so that no other edges are constructed. In retrospect, we should have made sure that always at least three edges are constructed. This would have led to a connected graph with a single component, which is preferable because it integrates the otherwise disconnected nodes in the clustering. However, the component also shows that these surface forms' embeddings are similar across their mentions. Fur-

	Nodes	Edges	Avg. Node Degree	ω	S
SF	1952	5304	5.434	-0.2542	38.8671
M	4757	11742	4.936	-0.4345	

Table 4.2.: Comparison of graphs constructed by different modes. Small-world metrics omega (Telesford et al., 2011) and S (Humphries and Gurney, 2008)

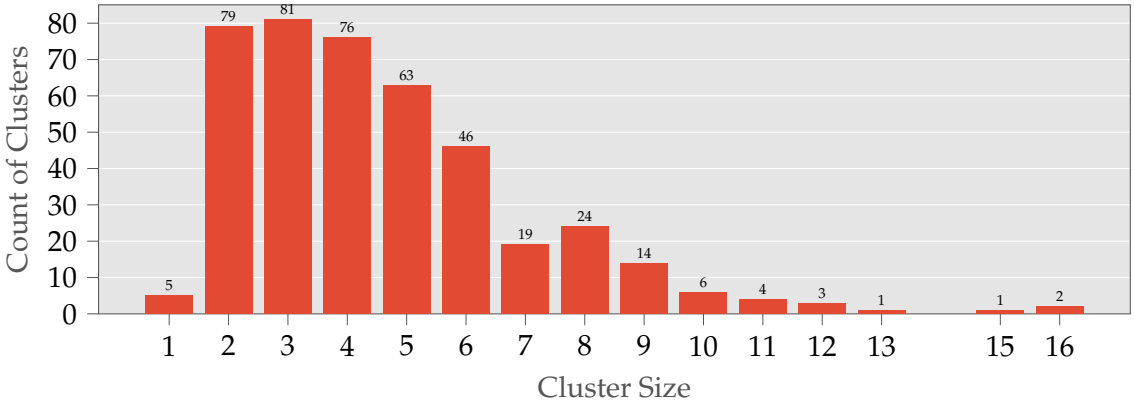
thermore, there are 20 bridges in the larger component, i.e. edges that are the only edges connecting two otherwise disconnected components. The bridges should be considered in further experiments when tuning the top-k or testing a similarity threshold for edge pruning. A higher number of bridges shows a risk that the graph may break into components when pruned further. In contrast, fewer bridges may indicate a less sparse graph.

Some problems in NLP are known to be representable by small-world graphs. Small-world graphs are characterized by a high clustering coefficient and low average shortest path length between pairs of nodes. This combination allows message passing systems to perform well (Telesford et al., 2011) which CW essentially is. Two coefficients indicate whether a graph is a small-world: S (Humphries and Gurney, 2008) and ω (Telesford et al., 2011). Both metrics are based on comparisons to generated equivalent graphs that are either fully random or regularly structured. Humphries and Gurney (2008) deem a graph as a small-world if $S > 1$. Additionally to just determining small-worldness, ω places a graph in the range of -1, completely structured, to 1, purely random. In their experiments, Humphries and Gurney find that small-world graphs usually fulfill $-0.5 \leq \omega \leq 0.5$. However, they state that the exact range varies with the size of the graph. We report small-world measures for SF and M modes in Table 4.2. Unfortunately, the computation for S of the M graph did not complete in time. The generation of equivalent graphs is computationally expensive and was not finished in a week. Still, the ω measures show that the M graph follows more structured patterns and the SF graph has more indications of small-world properties. The small-worldedness of SF is further supported by its S measure. Although we have no point for comparison, it falls in the restriction for small-worlds of $S > 1$.

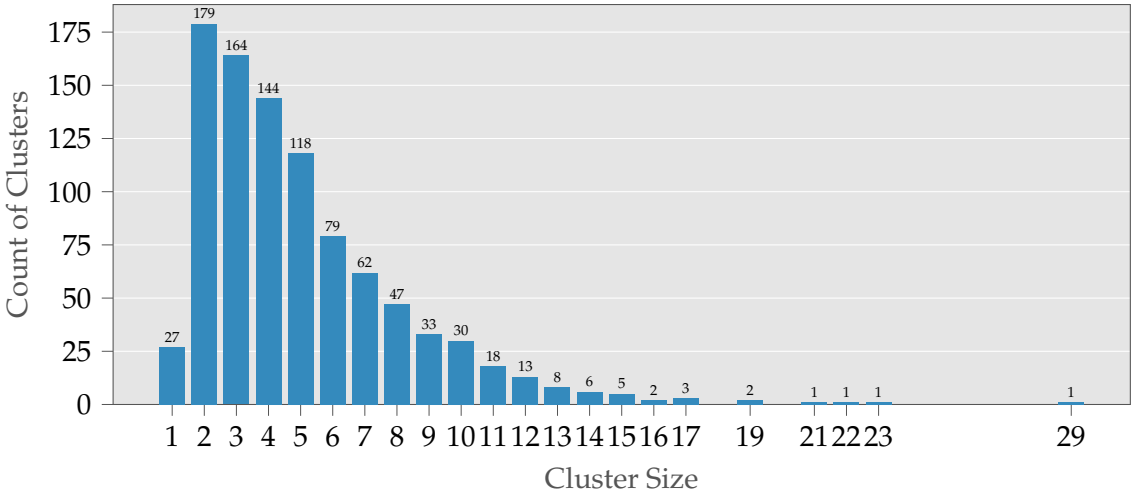
4.3.2. Cluster Properties

We cluster the entity mentions to draw knowledge from their semantic similarity. To achieve that, we want the clusters to incorporate multiple mentions that can be generalized over. Arguably, the more mentions are comprised by a cluster, the more knowledge it accumulates. However, in a real-world application, this would increase the risk of incorporating unrelated mentions. To not add too much noise to the cluster, a middle ground is needed. A cluster of a single node does not give a knowledge gain and is thus to be averted. As a start, we look at the distribution of cluster sizes in Figure 4.3 comparing the SF and M modes with BERT embeddings. CW produced 430 clusters in SF and 944 in M. The plots for both modes are similar. Both have only a few clusters of size one.

Then, a peak directly follows at sizes two or three. From there, the count is declining. While the largest cluster in `SF` has 16 nodes, the `M` mode has nine clusters larger than that up to size 29. The similarity suggests that the differences in the modes do not affect the clustering in regard to the cluster sizes.



(a) `SFGC-bert` mode



(b) `MGC-bert` mode

Figure 4.3.: Count of clusters by size in comparison for `SF` amd `M` modes. Here, size is the number of nodes in a cluster.

Figure 4.4 shows an example cluster that is a semantic capture of Major League Baseball (MLB) teams. Most mentions refer to the teams by their names directly. Only "St. Louis" mentions the city's name, however all mentions of "St. Louis" are in the context of MLB matches. Some of those mentions refer to St. Louis, the city, and some refer to the St. Louis Cardinals[14]. Again, this example is not actually unambiguous as described in Section 4.2. Still, the cluster assignment shows that the CWE captured the baseball context. However, this also marks a vulnerability of `SF`. Imagining an additional mention in a non-MLB context, e.g. talking about St. Louis' city council, would mingle the contexts because the different mentions are merged on the surface form "St. Louis". The cluster still captures the similarity of 52 mentions with 12 surface forms. Anecdotally, the

[14]https://www.wikidata.org/wiki/Q504309

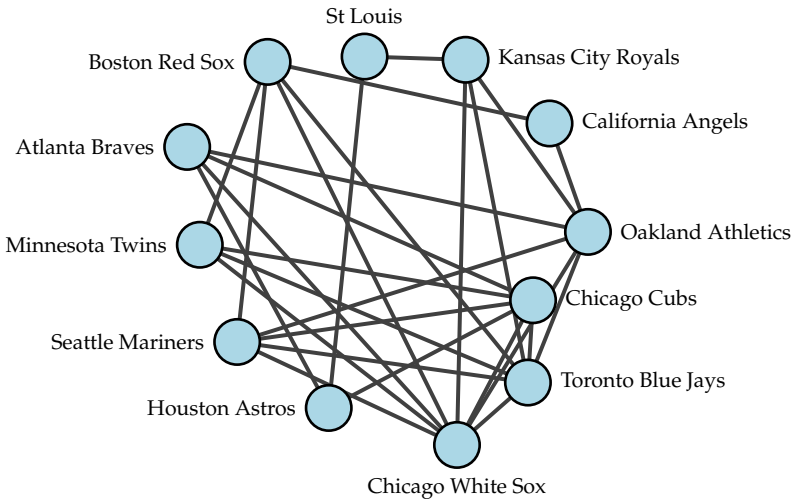


Figure 4.4.: Cluster consisting of mentions of Major League Baseball teams' names and of "St Louis". The nodes can have additional edges to nodes outside the cluster that are omitted for brevity.

surface form "St. Louis Cardinals" does not appear in the cluster because it is found to be ambiguous. There was an American football team[15] before the MLB team that currently uses the name.

In Figure 4.5, we show the unambiguous mentions as clustered in `SFGC-bert` and `SFHC-bert` (surface form merged, hierarchical clustering, using BERT embeddings). The position of the mentions is calculated by applying t-SNE (Van der Maaten and Hinton, 2008) to their embeddings. T-SNE reduces the dimensionality of the embeddings to fit into the 2D space. We use the same t-SNE mapping in both plots and denote the clusters through combinations of color and shape. Both plots show some discernable clusters, for example the dark-orange squares in Figure 4.5a or the yellow crosses in Figure 4.5b. `SFHC-bert` produces fewer but larger clusters while the clusters in `SFGC-bert` are more fine-grained so that we only colored clusters with five or more members in Figure 4.5a. The larger clusters of `SFHC-bert` are visually seperable indicating that the clustering is too coarse. All in all, Figure 4.5 gives a visual insight into how the clusterings differ.

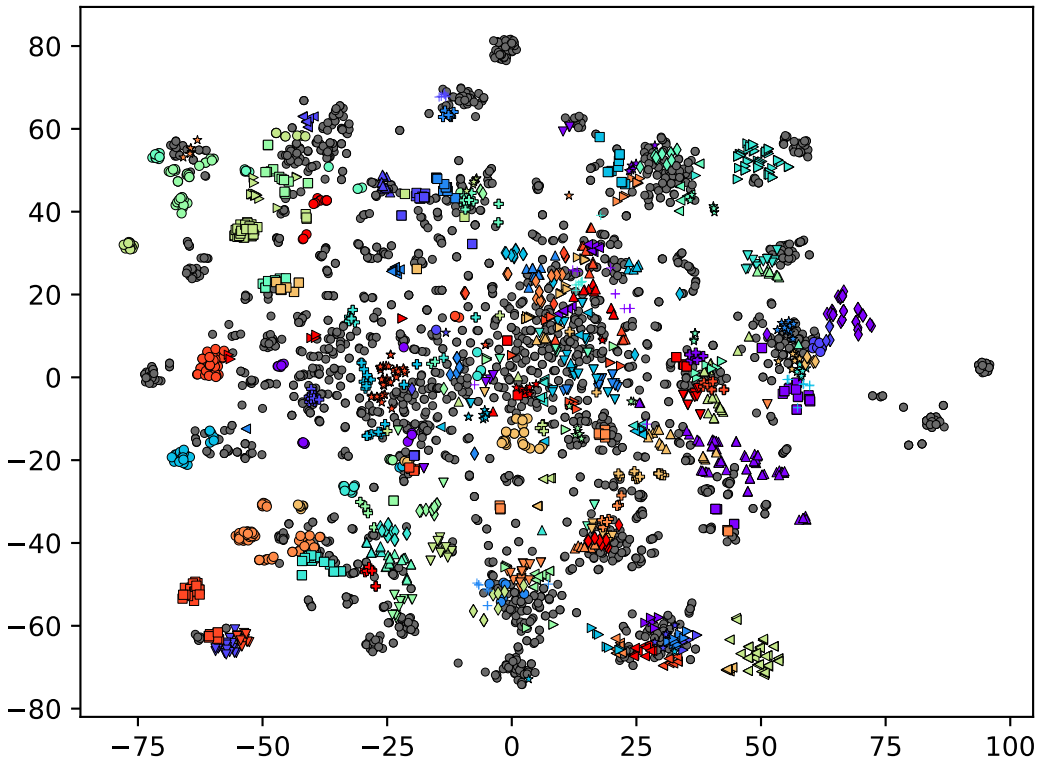
4.3.3. Entity Assignments

We describe the inference of our methodology in Section 3.6. Now, we investigate how the inference behaves on the development portion of the AIDA dataset. To start with an example, "The Netherlands" in the following sentence is assigned to the cluster in Table 4.3a.

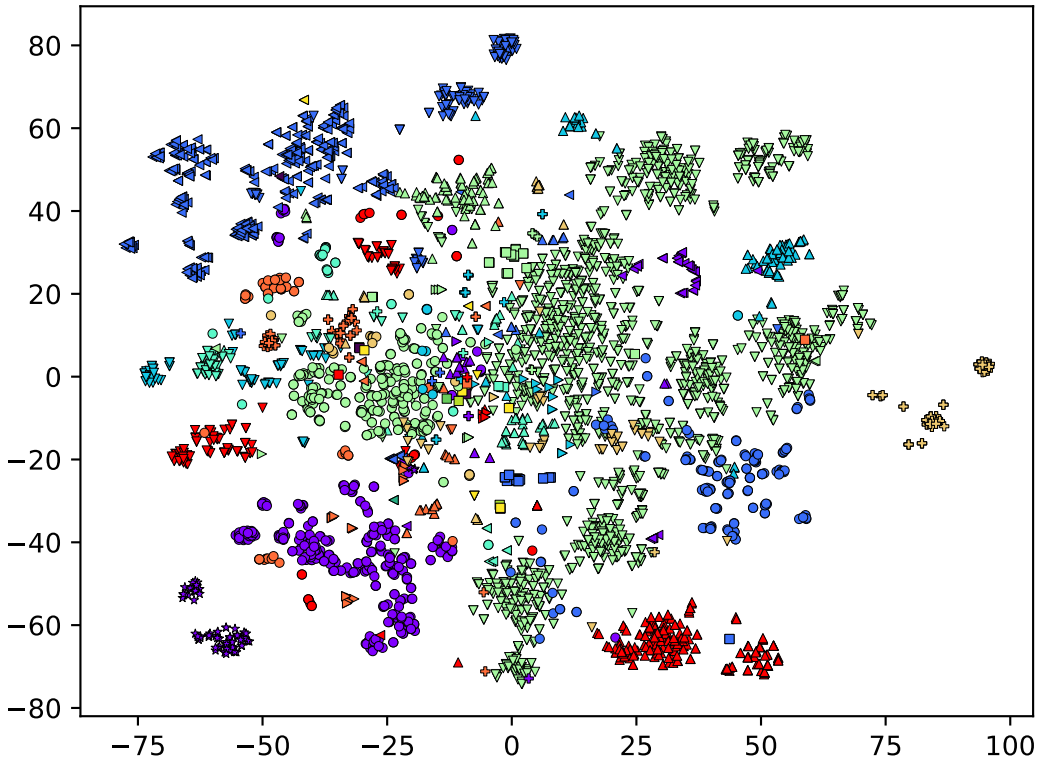
> The Netherlands drew 2 with Brazil (half 0) in a soccer friendly on Saturday.

The assignment captures the soccer topic of the sentence. The cluster also is composed of

[15] https://www.wikidata.org/wiki/Q20976246



(a) `SFGC-bert`; grey nodes belong to clusters with less than 5 members.



(b) `SFHC-bert`

Figure 4.5.: T-SNE (Van der Maaten and Hinton, 2008) plots of the mentions' embeddings in comparison of two clustering methods. Clusters are denoted by combination of color and shape of the nodes.

(among others) Dutch football clubs, which should contribute to the similarity between the sentence and the cluster.

The development set incorporates 5846 mentions in 3227 sentences. We plot the distribution of clusters being assigned to multiple mentions in Figure 4.6. It shows that most of the clusters are assigned to only a few mentions in the dev-set. Further, most of the mentions are assigned to one of those clusters. However, there are also a few clusters that are the most similar to many mentions. While such a distribution is expected, the often assigned clusters might be favored due to the nature of the AIDA dataset that we described in Section 4.1.1. Complementing Figure 4.3a, Figure 4.7 shows the number of assignments per cluster size. Comparing both figures, the latter's distribution follows the former's. This observation suggests that the assignment is independent of the cluster sizes.

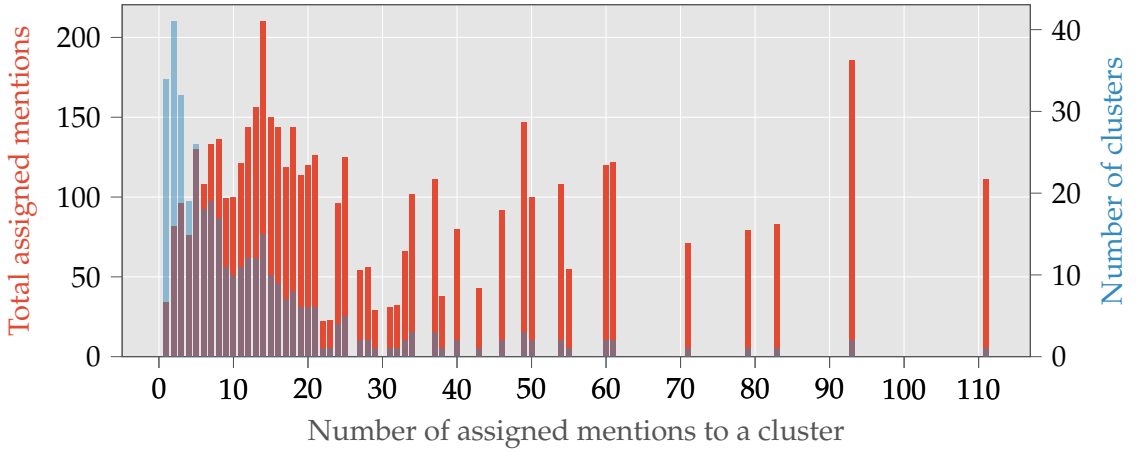


Figure 4.6.: Number of assigned mentions and number of cluster by how many mentions are assigned a cluster. E.g., at 50 on the x-axis shows that there are two clusters to which 50 mentions are assigned during evaluation, totaling 100 mentions.

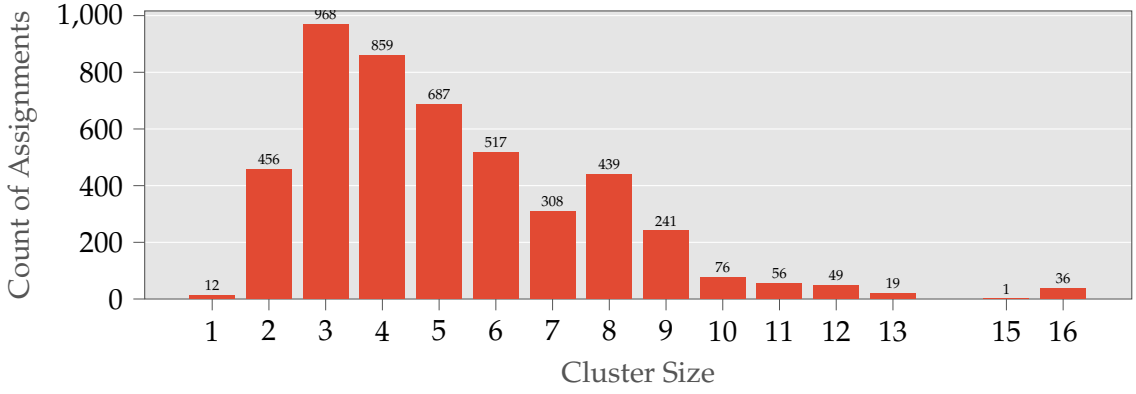


Figure 4.7.: Number of entity mentions from the AIDA dev-set grouped by cluster sizes. Here, cluster size means the number of mentions that the cluster is build by during training.

Surface form	Wikipedia ID	Wikidata ID	Wikidata description
Sparta Rotterdam	79818	Q209895	Dutch association football club
Vitesse Arnhem	834256	Q219233	Dutch association football club based in Arnhem
Fortuna Sittard	1336325	Q854167	association football club in the Netherlands
Willem II Tilburg	834275	Q332664	association football club in the Netherlands
Genk	1433871	Q189692	city an municipality in Limburg, Belgium
NAC Breda	505173	Q332642	association football club
Red Star Belgrade	361966	Q173009	Serbian association football club
Roda JC Kerkrade	834196	Q24719	Dutch professional association football club
RKC Waalwijk	1107824	Q24699	Dutch association football club
Andruw Jones	875934	Q1376903	retired Major League Baseball center fielder
Ajax Amsterdam	2273	Q81888	Dutch professional football club based in Amstedam
AMSTERDAM	844	Q478362	None
NEC Nijmegen	834154	Q318348	Dutch association football club from Nijmegen

(a) Entities that comprise the first example cluster. Most of the entities are association football clubs.

Surface form	Wikipedia ID	Wikidata ID	Wikidata description
Jordi Burillo	14891740	Q964079	Spanish tennis player
Tommy Haas	1118005	Q53560	German tennis player
Renzo Furlan	3929800	Q1852367	Italian tennis player
Kris Goossens		Q6437173	tennis player
Francisco Clavet	4076952	Q974264	Spanish tennis player
Dominique Van Roost	1849753	Q270967	Belgian tennis player
Roberto Carretero	1723684	Q1640440	Spanish tennis player

(b) Entities that comprise the second example cluster. The empty Wikipedia ID means that the mention is not labeled in the AIDA dataset.

Table 4.3.: Entities in two example clusters with the surface form that they are mentioned with, the Wikipedia ID that is labeled in Aida, the Wikidata ID based on the unambiguity heuristic and the corresponding Wikidata description. The Wikipedia ID can be used in `https://en.wikipedia.org/?curid={ID}` and Wikidata ID in `https://www.wikidata.org/wiki/{ID}`.

To display the disambiguation capabilities of our approach, we show the mentions with the surface form "Estes" as an example. Estes is mentioned two times in the AIDA development set and is one of only 116 surface forms with multiple different labels. The labels refer to Bob Estes[16], an American golfer and Shawn Estes[17], a former MLB pitcher. Our approach produces *former Major League Baseball player* for Shawn Estes by the best-scored n-gram from the most similar cluster. Judging by the Wikipedia page, this is an almost perfect description. Only the specific role within a baseball team is missing from our description. For Bob Estes, our model does not perform as well. Only the third most similar cluster provides a description incorporating the *golfer* profession. The most similar cluster is described as *Miss Universe* and the second most similar as *sprinter*. However, this example shows that our model can disambiguate mentions through similarity of CWEs.

4.4. Description-focused Experiments

We now layout experiments focused around the cluster descriptions, which we introduced in Section 3.5. First, we exemplarily compare the three methods for generating the cluster descriptions. Second, we introduce an evaluation scheme based on the n-gram overlap of the descriptions with gold standard Wikipedia pages. Thirdly, we compare variations on our methodology by evaluating the variations using the evaluation scheme. Last, we show the scalability of our approach by applying it to the Wikipedia corpus that we introduced in Section 4.1.2.

4.4.1. Describing Clusters

The three methods for generating descriptions, presented in detail in Section 3.5, are now compared to yield the one used in the following experiments. For each method we present the generated descriptions for two clusters shown in Table 4.3. The first example cluster consists of football clubs that are mostly based in the Netherlands. There are also a few clubs from other countries, a baseball player and the mention "AMSTERDAM", which is capitalized as being the dateline of a news report. The second cluster comprises seven European tennis players. The goal is to produce a text for each cluster that is a proper description of the entities that are part of the respective cluster. While the first cluster is a mix of entities that appear to be rather different, we expect the generated description to focus on the football clubs, optionally taking the focus on the Netherlands into account. Being more concise, we expect the second cluster to express clearly that the incorporated entities are tennis players. The two example clusters are used to exemplary display the capabilities of the methods.

[16] https://en.wikipedia.org/?curid=5841428
[17] https://en.wikipedia.org/?curid=3902338

Language Models

The first of our experiments generates descriptions using statistical language models. For each cluster, we build a separate language model based on its cluster's Wikidata descriptions. The Wikidata descriptions are tokenized and transformed into uni-, bi-, and trigrams. We use n-gram counts with Kneser-Ney smoothing to train the model. In the following, we display five sample outputs for both example clusters. Both examples show only cases in which the language model merely reproduces the input sequences rather than abstracting from them. Even though we do not generate proper sentences, we do not have enough data that the LMs can learn from to generate abstracted descriptions.

First example:

- retired Major League Baseball center fielder
- association football club in the Netherlands
- Dutch association football club based in Arnhem
- Dutch association football club
- association football club in the Netherlands

Second example:

- Belgian tennis player
- Italian tennis player
- German tennis player
- Italian tennis player
- Spanish tennis player

tf-idf

For the second experiment, we generate descriptions based on tf-idf scores. Again, from each cluster, we use the Wikidata descriptions to produce uni-, bi-, and trigrams. The n-grams serve as the terms and the clusters as documents. Calculating the tf-idf scores yields the n-grams that are most relevant to the cluster in the context of all clusters, similar to how tf-idf works in information retrieval. We display the top three n-grams for each regarded level along with their respective tf-idf scores for both example clusters in Table 4.4. For both examples, some n-grams represent the cluster's entities well while also generalizing rather than just reproducing the entire entity descriptions. However, it is not entirely clear which n-gram length is the best representation of the cluster. In the second example, "Spanish tennis player" has a higher tf-idf score than "tennis player" although the latter can be considered the better representation. Evidently, the scores cannot be compared easily. An adjustment might be to compute tf-idf across the n-gram lengths and not seperately for each length.

Log Likelihood Ratio

The third experiment continues the second by substituting the tf-idf scores with the LLR (Dunning, 1993) measure. LLR is based on event co-occurrence and originally used with bi-grams. We measure the co-occurrence of n-gram X with cluster Y across the three n-gram lengths. The counted event combinations can be seen in Table 3.1. Based on the

club	**0.5284**
football	0.3655
association	0.3645
football club	**0.6712**
association football	0.4470
Dutch association	0.3241
association football club	**0.8116**
Dutch association football	0.4191
in the Netherlands	0.3265

(a) Example 1 from Table 4.3a

tennis	**0.7273**
Spanish	0.5473
player	0.4129
tennis player	**1.1243**
Spanish tennis	1.0375
Belgian tennis	0.3910
Spanish tennis player	**2.2480**
Belgian tennis player	0.8473
Italian tennis player	0.7912

(b) Example 2 from Table 4.3b

Table 4.4.: Top n-grams by tf-idf score, n-gram lengths seperated by rules.

football club	**47.28**
club	44.89
association football club	42.87
Dutch association football club	32.22
association football	26.36

(a) example 1 from Table 4.3a

tennis player	**29.17**
tennis	29.11
Spanish tennis player	25.05
Spanish tennis	25.05
Spanish	19.4

(b) example 2 from Table 4.3b

Table 4.5.: Top n-grams by LLR score for the two examples clusters.

event occurrence counts, we use the LLR score formula shown in Equation 1 to calculate scores for the n-grams in each cluster. The five highest scored n-grams for the two examples from Table 4.3 can be seen in Table 4.5. Both "football club" and "association football club" are pretty good descriptions for the example (a) cluster because they represent most of the cluster's entities abstracting from their descriptions while also neglecting outliers. For the second example "tennis player" represents the entities nicely, "tennis" at least gives a general topic and the n-grams including "Spanish" are already somewhat specific to s smaller subset of the cluster's entities.

4.4.2. Evaluation Scheme

We introduce an evaluation scheme based on n-gram overlap with gold standard Wikipedia pages. In the following sections, this scheme is used for a quantitative comparison of variations on our methodology as well as on the training data. From the description generation methods introduced in Section 4.4.1, we use the LLR method from now on because it showed the most promising results in preliminary experiments.

This quantitative evaluation is based on the labeled data from the AIDA-CoNLL dataset described in Section 4.1. We use the development set of AIDA providing us with 6452 labeled mentions in 2296 sentences. For each of those mentions we build embeddings and calculate the cosine similarities to the centers of the trained clusters. Derived from precision@k in information retrieval, we look at the k most similar clusters for each mention

and for those at the k highest LLR scored n-grams per cluster. We argue that depending on the downstream task multiple candidates might be needed or might improve the downstream results or that a downstream implementation needs a tasks specific selection of the best candidate so that using the top-k for evaluation makes sense. Ultimately, we have three clusters with three descriptions each, yielding nine candidate descriptions for scoring. Each candidate is compared to the lead section[18] (i.e. the first section before the table of contents and main page content) of the annotated Wikipedia page. Figure 4.8 shows the n-gram matching at the example of Shawn Estes being mentioned in a sentence, the Wikipedia page referred to by the AIDA label, and the top-scored descriptions for the most similar clusters. The comparison uses n-gram co-occurrence based on ROUGE (Recall-Oriented Understudy for Gisting Evaluation, Lin, 2004) scores. ROUGE is commonly used for evaluation of summarization and translation tasks to compare a machine output with a gold standard. We calculate precision for ROUGE-N with uni-, bi-, and tri-grams, ROUGE-L, ROUGE-S, and ROUGE-SU with skips of up to four tokens. We use precision rather than recall that ROUGE normally focuses on because the target sequence, i.e. the lead section, often consists of multiple sentences while the system sequence, our description, consists of only a few tokens based on the n-gram length used for the LLR scoring. The best scores for each mention are averaged to yield the scores on the whole dataset.

Estes (3) was lifted for closer Rod Beck after yielding a single with two out in the ninth.

1. baseball player
   baseball
   from the United States
2. baseball player
   baseball
   Puerto Rican baseball player
3. former Major League Baseball player
   former Major League Baseball
   Major League Baseball player

Figure 4.8.: Example sentence mentioning Estes that is labeled with the Wikipedia page (retrieved 06.04.22) on the right. Below the sentence are the 9 descriptions for the 3 clusters with the highest similarity to the mention.

In Section 4.3.2, we already looked at different cluster sizes. With the cluster descriptions and the evaluation scheme, we investigate how the cluster descriptions represent their clusters. We calculate the ROUGE scores as described above but do not assign

[18] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section

ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU4	Cluster Size	Count of Clusters
0.800	**0.750**	**0.760**	**0.743**	1	5
0.468	0.207	0.457	0.334	2	79
0.451	0.213	0.438	0.308	3	81
0.508	0.182	0.497	0.339	4	76
0.563	0.192	0.552	0.366	5	63
0.576	0.253	0.575	0.419	6	46
0.524	0.148	0.515	0.356	7	19
0.491	0.189	0.486	0.289	8	24
0.538	0.201	0.538	0.351	9	14
0.603	0.127	0.597	0.345	10	6
0.315	0.160	0.310	0.213	11	4
0.710	0.028	0.710	0.308	12	3
0.667	0.583	0.667	0.625	13	1
0.308	0.308	0.308	0.308	15	1
0.293	0.133	0.293	0.279	16	2
0.416	0.185	0.081	0.207	all	430

Table 4.6.: ROUGE scores for n-gram overlap between cluster descriptions and the Wikipedia pages annotated at the cluster's mentions; Scores are grouped by cluster size and number of clusters in each group; Using the `SFGC-bert` model, i.e. cluster size means number of nodes in the cluster with possibly more mentions

unknown entities, yet. Instead, we use the Wikipedia entities as targets that are annotated at the mentions that constitute the cluster. We show a selection (for brevity) of the ROUGE scores across cluster sizes in Table 4.6. The clusters with only a single node score best across all metrics. As mentioned before, single-node clusters should be avoided because there is no gain by generalization. However, the high scores suggest that we found matching Wikidata entities to the gold standard Wikipedia entities. The scores for the other sizes are lower and vary, for example, from about 0.3 to 0.7 in ROUGE-1. This indicates that the description generation has potential for improvement when it actually needs to generalize over multiple Wikidata descriptions, in contrast to the single-node clusters. The number of clusters with a larger size declines and the low numbers might explain the variance of the scores there. The smaller sizes have more clusters and are more stable around 0.5 in ROUGE-1.

4.4.3. Baseline & Variation Experiments

We explore variations of our methodology to get an insight into how it behaves with different implementations. The quantative evaluation is based on the scheme presented in Section 4.4.2 and the resulting scores can be seen in Table 4.7. Each experiment is trained on the AIDA training set and evaluated on the AIDA development set. The baseline experiments are expected to perform noticeably worse than the other approaches and are used to show the gain of using CWEs and clustering those. The variant experiments,

		Baseline Experiments		Variant Experiments		
	SFGC-bert	MNN-bert	SFGC-SE	SFGC-roberta	MGC-bert	SFHC-bert
ROUGE-1	0.452	0.350	0.323	**0.461**	0.443	0.321
		-23%	-29%	+2%	-2%	-29%
ROUGE-2	**0.155**	0.109	0.055	0.102	0.152	0.059
		-30%	-65%	-35%	-2%	-62%
ROUGE-3	0.050	0.037	0.013	0.024	**0.052**	0.007
		-26%	-74%	-52%	+4%	-86%
ROUGE-L	0.422	0.331	0.307	**0.445**	0.418	0.291
		-22%	-28%	+5%	-1%	-32%
ROUGE-S4	0.152	0.110	0.055	0.117	**0.158**	0.059
		-28%	-64%	-24%	+3%	-62%
ROUGE-SU4	0.275	0.221	0.170	0.261	**0.277**	0.148
		-20%	-39%	-6%	+1%	-47%

Table 4.7.: ROUGE scores for the main method in comparison with baseline and variant experiments. All are trained on the AIDA dataset. Percentages are the gain/loss on `SFGC-bert`.

however, change only minor parts of the methodology. Thus, we expect them all to perform similarly.

In the previous sections of this chapter, we already focused our discussion on `SFGC-bert` (surface form merged, graph clustered, using BERT embeddings). It represents our methodology as we initially imagined and implemented it. Thus, we use it here as a central point for comparisons as well. In `SFGC-bert`, we use CW for clustering. The mentions are merged on their surface form so that all mentions of a surface form build up a single node in the graph and embeddings for the mentions are built using BERT.

For `MNN-bert` (mention-specific, nearest neighbor, using BERT embeddings) we do not build a graph from the training mentions and cluster that graph but treat each mention as a separate candidate during evaluation. In contrast to `SFGC-bert`, this cannot generalize over the similarities of CWEs for similar entities because we do not group the mentions. Indeed, the scores decrease by 20 to 30 percent. However, a benefit is the less demanding training due to the lack of clustering that becomes more complicated with large amounts of training data.

In `SFGC-SE` (surface form merged, graph clustered, using static embeddings) we substitute the BERT embeddings with static embeddings produced by FastText (Bojanowski et al., 2017). While we introduce FastText shortly here, it is described in more depth in Section 2.2. FastText is a library that can generate static word embeddings even for OOV words. The OOV capability is the reason for us choosing FastText over the prominent Word2Vec, which cannot produce embeddings for words that are not part of the training dictionary. FastText works on character n-grams rather than whole tokens. As long as one of the n-grams has been seen during training of the embedding model, it can use those to produce embeddings for unseen words. This is important for embedding entity mentions because a finite length dictionary cannot incorporate the open class entity names.

Compared with CWEs as produced by BERT, static embeddings do not encapsulate the context of a token in the sequence it occurs. Instead, the embeddings are always the same for a surface form. This leads to lexically ambiguous words like homographs, i.e. words with the same spelling that carry different senses, being represented by the same embedding. We expect this to be disadvantageous to our approach and thus decrease the scores in the evaluation. In comparison, we show with this experiment that CWEs incorporate enough context for our model to differentiate ambiguous entity mentions. The respective scores in Table 4.7 indicate this lack of disambiguation as they are worse than those for the unclustered mentions in `MNN-bert` which uses CWEs.

`SFGC-roberta` (surface form merged, graph clustered, using RoBERTa embeddings) continues the `SFGC-bert` experiment by using RoBERTa (Liu et al., 2019) for building embeddings. Robustly optimized BERT Pretraining Approach (RoBERTa) itself is an evolved version of BERT that aims to outperform BERT by improving training conditions with, among others, a larger training dataset, more focus on word masking and adjusted hyperparameters (see also Section 2.2). In contrast to our expectation that the revised training of RoBERTa would definitely benefit our approach, the scores in most metrics decreased compared to `SFGC-bert`. Still, `SFGC-roberta` outperforms all other models on ROUGE-1 and ROUGE-L and is close behind `SFGC-bert` in ROUGE-SU4.

`MGC-bert` (mention-specific, graph clustered, using BERT embeddings) is a variant on the graph construction. In `SFGC-bert`, all mentions with the same surface form are merged into a single node in the graph. That follows our assumption that all those mentions sharing a surface form refer to the same entity because, based on our heuristic, only unambiguous mentions are used for the clustering. We now investigate the impact of that merging of mentions into a single node by comparing this unmerged version in `MGC-bert`. The Table 4.7 shows that the `MGC-bert` scores are the closest to `SFGC-bert`. It seems that merging mentions on their surface form does not have an impact as large as expected.

A comparison for the whole clustering approach is taken with `SFHC-bert` (surface form merged, hierarchical clustering, using BERT embeddings). Here, we use hierarchical agglomerative clustering as a substitute for CW along with our graph construction methodology. We choose hierarchical clustering because it allows us to gain an insight into the clustering decisions, such as dendrograms. We conduct a hyperparameter optimization whose evaluation scores are shown Table A.1 and whose top score we use for the comparison Table 4.7. In Section 4.3.2, Figure 4.5b shows a t-SNE plot of the hierarchical clustering. While the plot shows a few rather concise clusters with some outliers, those clusters include nodes that are visually seperated. There is also a large mass of mentions where no clear clusters are recognizable. Despite the hyperparameter optimization and the t-SNE showing some concise clusters, `HC` does not perform well in our evaluation scheme. The scores show even worse performance than the baseline experiments. In general, `HC` should perform on par with `GC` so that we believe that there might be an

error somewhere in `SFHC-bert`.

4.4.4. Wikipedia Dataset

We now show that our approach is applicable to other datasets and scales with increasing training data. In Section 4.1.2, we introduce a second corpus in addition to the AIDA dataset. It is based on the plain text of random Wikipedia pages, of which we use 20 000 and 30 000 mentions as two sets for comparison. We train our model on both sets and then compare the scores with the model trained on the AIDA corpus. The scores shown in Table 4.8 are calculated as introduced in Section 4.4.2 and all models are build and trained the same as `SFGC-bert` in Section 4.4.3. As before, all three experiments are evaluated on the AIDA development set.

	AIDA	Wikipedia 20k	Wikipedia 30k
ROUGE-1	0.452	0.541	**0.544**
ROUGE-2	**0.155**	0.123	0.131
ROUGE-3	**0.050**	0.031	0.043
ROUGE-L	0.422	0.520	**0.526**
ROUGE-S4	**0.152**	0.131	0.138
ROUGE-SU4	0.275	0.280	**0.293**

Table 4.8.: ROUGE score comparison of the datasets. All three models are build using `SFGC-bert`.

The smaller Wikipedia dataset already improves the method's performance compared to the training on AIDA in three of the six reported metrics. It outperforms AIDA in ROUGE-1, ROUGE-L and ROUGE-SU4. With the additional training data, the larger Wikipedia model outperforms the smaller one in all metrics. As described in Section 4.1.2, the 30k corpus just extends the smaller by the 10 000. Increasing the score only by extending the dataset shows that the increase is caused by the larger training data rather than the selection of the data. While the larger model gets closer to the AIDA model in the lacking scores, it cannot overtake the AIDA model. However, as mentioned in Section 4.1.1, the AIDA corpus does not consist completely of full sentences and is special in being based on news items. This bias might have led to the AIDA model being over-specialized for that kind of sequence and evaluation being harder for other models.

5. Summary, Conclusion & Future Work

5.1. Summary

In this thesis, we introduced an unsupervised approach to entity disambiguation. This novel approach is based on a heuristic approach to unambiguous entity mentions. Using the heuristic, we found matching Wikidata entities to the mentions in two datasets without requiring annotations. We clustered the unambiguous mentions based on the similarity of their CWEs. For clustering, we compared a graph clustering algorithm with hierarchical clustering. We constructed a sparse graph for the graph variant that allows computationally efficient clustering. Then, we showed three methods to generate short texts describing the clusters based on the descriptions of the individual entities.

Once trained, our model generates short descriptions for arbitrary entity mentions. To generate a description, the mention's VSR is matched to the VSRs of the clusters. The most similar cluster supplies its description, which was generated during training. We evaluated the approach with the Wikipedia labels of the AIDA CoNLL-YAGO dataset. As no labeled dataset specific to our needs exists, we introduced a custom evaluation scheme. It compares the description with the Wikipedia pages annotated in the AIDA dataset. Using the evaluation scheme, we investigated the performance gains and losses of model variants. We found three models to perform similarly, with each having a top score in different metrics. Additionally, we showed that our model improves performance on more extensive training data.

We described the data at each step of our methodology. First, we introduced the two datasets on which we base our work. Then, we showed examples of the unambiguity heuristic succeeding, contrasting them with examples for which the heuristic produced incorrect results. Additionally, our evaluation scheme quantitatively indicated a good performance of the heuristic. We illustrated the properties of the constructed graph and the clusters found in it. Primarily, we showed that the graph is sparse and possesses small-world characteristics. Further, we gave an insight into how entity mentions are assigned to clusters during inference.

5.2. Conclusion

In Section 1.3, we set four research questions to our hypothesis. First, we asked whether our heuristic produces correct links. While we do not have a metric proving it, we found some evidence that suggests correct links are produced. In Section 4.4.2, we saw that the

scores for single-node clusters are substantially higher than those for other sizes. While the sample size is small, this difference indicates that we can produce correct links because the score is high when we merely reproduce descriptions. In contrast, the score decreases when the description-generation method abstracts over multiple entities. In Section 4.2, we showed examples of the heuristic both succeeding and failing. During our experiments, we saw many more examples of both. However, some of the negatives seemed to result from the issues of the AIDA corpus that we described in Section 4.1.1. We reason that the heuristic performs up to our expectations. The evidence suggests a high precision and a recall high enough to produce sufficient numbers of unambiguous mentions.

Second, we researched whether we can produce semantically related clusters of the unambiguous mentions. We showed an example of a semantically concise cluster in Figure 4.4. Naturally, it is only one cluster among many and there are also clusters whose entities have no obvious semantic similarities. Looking at the scores in Table 4.7, two variants change the clustering method: `MNN-bert` and `SFHC-bert`. As a reminder, `MNN-bert` does not cluster mentions at all and `SFHC-bert` uses hierarchical clustering. The scores of both are inferior to those of the graph clustered methods. Furthermore, `SFGC-SE` shows that static embeddings do not work as well as CWEs. We answer the second research question positively because the effect of clustering CWEs is apparent.

Third, we examine methods to generate text descriptions for the clusters. We devised three methods that we described in Section 3.5: a statistical language model as well as n-grams scores by both tf-idf and LLR. They were compared by examples in Section 4.4.1 showing the capabilities of each. We found LLR to work best.

Last, we investigated the disambiguation capabilities of our approach. We gave an example on those capabilities in Section 4.3.3. Unfortunately, the AIDA corpus does not offer many of those examples. The example shows that the clusters definitely can disambiguate entity mentions. However, we can not yet determine the performance in comparison with other approaches.

5.3. Future Work

Throughout the thesis, we already left some improvements up for future work. First and foremost, the issues of the AIDA corpus that we described in Section 4.1.1 caused problems throughout our work for this thesis. Consequently, future work should include changing the dataset. We believe that a training corpus that is more general than AIDA will benefit the evaluation in general and interpretability specifically. Furthermore, we already showed that our approach is easily adapted to a plaintext corpus and increases performance on more training data. Thus, the models would benefit from using a large plaintext corpus for training. A new dataset for evaluation should focus on disambiguation rather than EL. The task that we imagine is to determine whether mentions in multi-

ple contexts refer to the same or different entities. A model fulfilling that task could then be used in EL as well as other tasks.

Changing the dataset also allows for investigating the transferability of our approach to specific domains like medical texts. That would require access to a domain-specific KB to determine unambiguity. We believe that our approach should work similarly in a domain as long as the VSR captures the semantic similarities. Domain-adaption can be achieved in two ways. On the one hand, combined training of a single graph would require amassing corpora and KBs for all desired domains before training. On the other hand, a separate graph for each domain can be constructed and clustered. In that case, the inference of a mention in a multi-domain setting only requires the cluster centers. These can be calculated independently of each other, assuming that entities from different domains should not be mingled regardless.

In Section 3.3, we described how we build VSRs for entity mentions. Future work might consider different techniques to extract embeddings from the BERT model. We average the last layer of the respective model for all WordPiece tokens that compose the mention. Similar to the different approaches to sentence embeddings (Reimers and Gurevych, 2019), there are possible alterations to our approach on embeddings of entity mentions. Instead of averaging, we could directly use the embedding of the first or last token of a mention. The other layers of the BERT model could also be included in the averaging. There are also other works like LUKE (Yamada et al., 2020) that extend BERT architecturally to improve the understanding of entity mentions. Their learnings and models are left for further examination that could improve the VSR in capturing similarities and differences of multiple mentions.

During graph construction, we build three edges for each mention. The number of edges is a configurable hyperparameter and changes the structure of the graph. With more edges, the graph becomes denser and is less prone to have bridges or even disconnected components. However, increasing denseness has drawbacks, e.g. the clustering becomes less efficient. Instead of specifying the number of edges per mention, the graph could also be pruned by means of a similarity threshold. With a threshold, mentions that are similar to many other mentions would lead to high-degree nodes. In contrast, there might be nodes with no edges when a mention does not reach the similarity threshold for any other mention. Further experiments would be required to identify an appropriate threshold.

Bibliography

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics,* 1638–1649. Santa Fe, NM, USA, August. (Cited on page 11).

Hiba Arnaout, Simon Razniewski, Gerhard Weikum, and Jeff Z. Pan. 2021. Negative Knowledge for Open-World Wikidata. In *Companion Proceedings of the Web Conference 2021,* 544–551. New York, NY, USA: Association for Computing Machinery, April. (Cited on page 6).

Chris Biemann. 2006. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing,* 73–80. Manchester, UK: Association for Computational Linguistics, June. (Cited on pages 5 sqq., 22, 24, 35).

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* (Cambridge, MA, USA) 5 (June): 135–146. (Cited on pages 8, 20, 48).

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 87–96. Melbourne, Australia: Association for Computational Linguistics, June. (Cited on pages 3, 14 sq.).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 4171–4186. Minneapolis, MN, USA: Association for Computational Linguistics, June. (Cited on pages 1, 9 sq., 20).

Antonio Di Marco and Roberto Navigli. 2013. Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction. *Computational Linguistics* (Cambridge, MA, USA) 39, no. 3 (September): 709–754. (Cited on page 24).

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics* (Cambridge, MA, USA) 19, no. 1 (March): 61–74. (Cited on pages 27 sq., 44).

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* Miyazaki, Japan: European Language Resources Association (ELRA), May. (Cited on page 13).

Zellig S Harris. 1954. Distributional structure. *WORD* 10 (2-3): 146–162. (Cited on pages 8, 20).

Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. 2010. Foundations of Semantic Web Technologies. 1st ed. Chapman / Hall/CRC. (Cited on pages 3, 19).

Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. 2008. Semantic Web. Springer, Berlin, Heidelberg. (Cited on pages 5 sq.).

Johannes Hoffart, Mohamed Yosef, Ilaria Bordino, Hagen Fuerstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* 782–792. Edinburgh, UK: Association for Computational Linguistics, July. (Cited on page 31).

Mark D. Humphries and Kevin Gurney. 2008. Network 'Small-World-Ness': A Quantitative Method for Determining Canonical Network Equivalence. *PLOS ONE* (San Francisco, CA, USA) 3, no. 4 (April): 1–10. (Cited on page 37).

Anil K. Jain and Richard C. Dubes. 1988. Algorithms for Clustering Data. Hoboken, NJ, USA: Prentice-Hall, Inc. (Cited on page 25).

Manoj Prabhakar Kannan Ravi, Kuldeep Singh, Isaiah Onando Mulang', Saeedeh Shekarpour, Johannes Hoffart, and Jens Lehmann. 2021. CHOLAN: A Modular Approach for Neural Entity Linking on Wikipedia and Wikidata. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume,* 504–514. Stroudsburg, PA, USA: Association for Computational Linguistics, April. (Cited on pages 12 sq.).

R. Kneser and H. Ney. 1995. Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing,* vol. 1, 181–184 vol.1. (Cited on page 26).

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-End Neural Entity Linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning,* 519–529. Brussels, Belgium: Association for Computational Linguistics, October. (Cited on page 13).

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 260–270. San Diego, CA, USA: Association for Computational Linguistics, June. (Cited on page 12).

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the Sentence Embeddings from Pre-trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 9119–9130. Stroudsburg, PA, USA: Association for Computational Linguistics, November. (Cited on page 21).

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering* (Washington, DC, USA) 34, no. 1 (January): 50–70. (Cited on page 19).

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out,* 74–81. Barcelona, Spain: Association for Computational Linguistics, June. (Cited on page 46).

Xiao Ling and Daniel Weld. 2012. Fine-Grained Entity Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence* (Toronto, ON, Canada) 26, no. 1 (September): 94–100. (Cited on page 13).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv: 1907.11692 [cs.CL]. (Cited on pages 10, 20 sq., 49).

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-Shot Entity Linking by Reading Entity Descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics,* 3449–3460. Florence, Italy, July. (Cited on pages 12 sq.).

Oded Maimon and Lior Rokach. 2005. Data mining and knowledge discovery handbook. 2nd ed. New York City, NY, USA: Springer Science+Business Media, LLC. (Cited on page 24).

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering* (New York City, NY, USA) 16, no. 1 (January): 100–103. (Cited on pages 26 sq.).

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.* (Cited on pages 8 sqq., 20).

Stanley Milgram. 1967. The small world problem. *Psychology today* (New York City, NY, USA) 2 (1): 60–67. (Cited on page 6).

Robert C. Moore. 2004. On Log-Likelihood-Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing,* 333–340. Barcelona, Spain: Association for Computational Linguistics, July. (Cited on page 27).

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* (Baltimore, MD, USA) 2 (June): 231–244. (Cited on page 11).

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* (Amsterdam, Netherlands) 30 (1): 3–26. (Cited on pages 1 sq., 11, 13).

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers),* 2227–2237. New Orleans, Louisiana, USA: Association for Computational Linguistics, June. (Cited on page 11).

Lisa F. Rau. 1991. Extracting company names from text. In *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application,* 29–32. Miami Beach, FL, USA: IEEE. (Cited on page 11).

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP),* 3982–3992. Hong Kong, China: Association for Computational Linguistics, November. (Cited on pages 8, 21, 23, 53).

Steffen Remus. 2012. Automatically Identifying Lexical Chains by Means of Statistical Methods — A Knowledge-Free Approach. Master's thesis, Technische Universität Darmstadt, October. (Cited on pages 27 sq.).

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* 5149–5152. Kyoto, Japan, March. (Cited on pages 9 sq.).

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An Attentive Neural Architecture for Fine-grained Entity Type Classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction,* 69–74. San Diego, CA, USA: Association for Computational Linguistics, June. (Cited on page 14).

Chuanqi Tan, Furu Wei, Pengjie Ren, Weifeng Lv, and Ming Zhou. 2017. Entity Linking for Queries by Searching Wikipedia Sentences. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* 68–77. Copenhagen, Denmark: Association for Computational Linguistics, September. (Cited on page 1).

Qawi K Telesford, Karen E Joyce, Satoru Hayasaka, Jonathan H Burdette, and Paul J Laurienti. 2011. The ubiquity of small-world networks. *Brain connectivity* (New York City, NY, USA) 1 (5): 367–375. (Cited on page 37).

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4,* 142–147. Edmonton, Canada: Association for Computational Linguistics. (Cited on page 31).

Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, and Harshit Surana. 2020. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. Sebastopol, CA, USA: O'Reilly Media, Incorporated, June. (Cited on page 20).

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (11): 2579–2605. (Cited on pages 11, 39 sq.).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems,* vol. 30. Long Beach, CA, USA: Curran Associates, Inc. (Cited on pages 9 sq., 20).

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM* (New York, NY, USA) 57, no. 10 (September): 78–85. (Cited on page 5).

Duncan J Watts. 2000. Small worlds: The dynamics of networks between order and randomness. Princeton, NJ, USA: Princeton University Press. (Cited on page 6).

Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, no. 6684 (June): 440–442. (Cited on page 7).

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers,* 161–170. Erlangen, Germany: German Society for Computational Linguistics & Language Technology. (Cited on pages 3, 8, 10 sq., 20).

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.08144. (Cited on pages 9, 20).

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level Fine-grained Entity Typing Using Contextual Information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 715–725. Lisbon, Portugal: Association for Computational Linguistics, September. (Cited on page 13).

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 6442–6454. Online: Association for Computational Linguistics, November. (Cited on pages 14, 53).

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers),* 291–296. Beijing, China. (Cited on page 13).

G. Zipf. 1936. The Psychobiology of Language. London, UK: Routledge. (Cited on page 27).

Acronyms

Acronyms in `teletype` refer to our experiments while the others are more generally acronyms.

BERT Bidirectional Encoder Representations from Transformer

CBOW continuous bag-of-words

CW Chinese Whispers

CWE contextualized word embeddings

DAP domain-adaptive pre-training

ED entity disambiguation

EL entity linking

ET entity typing

GC graph clustering

HC hierarchical clustering

KB knowledge base

KNN k-nearest neighbor

LLR log likelihood ratio

LM language model

LSTM long short-term memory

M mention level

MD mention detection

MGC-bert mention-specific, graph clustered, using BERT embeddings

MLB Major League Baseball

MLM masked language model

MNN-bert mention-specific, nearest neighbor, using BERT embeddings

NER named entity recognition

NLP natural language processing

NSP next sentence prediction

OOV out-of-vocabulary

OWA open world assumption

POS part-of-speech

RoBERTa Robustly optimized BERT Pretraining Approach

SF surface form merged

SFGC-bert surface form merged, graph clustered, using BERT embeddings

SFGC-roberta surface form merged, graph clustered, using RoBERTa embeddings

SFGC-SE surface form merged, graph clustered, using static embeddings

SFHC-bert surface form merged, hierarchical clustering, using BERT embeddings

t-SNE t-distributed stochastic neighbor embedding

VSR vector space representation

WSD word sense disambiguation

A. Appendix

Hyperparameter Tuning

	euclidean		cosine									
	ward		average			complete			single			
	0.850	0.925	0.150	0.300	0.600	0.150	0.300	0.700	0.150	0.300	0.400	0.500
ROUGE-1	0.240	0.240	0.205	0.226	**0.321**	0.196	0.215	0.317	0.206	0.230	0.246	0.271
ROUGE-2	0.027	0.025	0.021	0.023	**0.059**	0.019	0.024	0.051	0.022	0.022	0.055	0.047
ROUGE-3	0.002	0.002	0.003	0.001	**0.007**	0.002	0.002	0.005	0.002	0.001	**0.007**	**0.007**
ROUGE-L	0.224	0.223	0.194	0.210	**0.291**	0.186	0.201	0.290	0.196	0.209	0.228	0.244
ROUGE-S4	0.033	0.028	0.026	0.030	**0.059**	0.025	0.028	0.057	0.027	0.024	0.037	0.032
ROUGE-SU4	0.125	0.124	0.106	0.115	**0.148**	0.102	0.111	0.145	0.107	0.110	0.119	0.137

Table A.1.: Hyperparameter Optimization for Hierarchical Clustering

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der elektronischen Abgabe entspricht.

Hamburg, den 25.04.2022

Frederik Wille

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 25.04.2022

Frederik Wille