Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor Thesis

Evaluating Rule-Based and Neural Merging Strategies of Entity Clusters for Coreference Resolution

Jan Mägdefrau

MIN-Faculty, Department of Informatics
Research Group: Language Technology
Course of study: Informatics
Matriculation number: 7260915

First examiner: Prof. Dr. Chris Biemann
Second examiner: Hans Ole Hatzel
Supervisor: Hans Ole Hatzel, Fynn Petersen-Frey

Date of Submission: 16.05.2023

Abstract

Coreference resolution is an essential pre-processing step for many natural language processing tasks. In the past, there has been a shift from rule-based approaches to machine-learning-based approaches. A recent approach that focuses on the German language involves two end-to-end trained models: a coarse-to-fine and an incremental model. While the first model suffers from the problem of requiring an increasing amount of memory, the second model performs worse than the coarse-to-fine model but only requires a constant amount of memory.

Unfortunately, there is currently no memory-efficient and well-performing method for predicting mentions and entities in long documents. To address this issue, we propose in this thesis the idea of splitting documents into shorter segments, predicting them using the coarse-to-fine model, and then merging the entities together. For the merging step, we propose three rule-based methods: string-matching, overlapping, and one based on word embeddings. Additionally, we introduce a neural merging method that adapts the incremental model.

The results of the thesis show that all the proposed methods underperform the given strong baseline of the coarse-to-fine model. Nevertheless, they all work with a limited amount of memory. Furthermore, we observed that the rule-based methods perform better than the more complex adapted incremental model. This thesis demonstrates that splitting documents and merging entities can be a viable solution for coreference resolution in long documents. However, further research is necessary to match the baseline of the coarse-to-fine model.

Contents

1 Introduction

In this first chapter of this thesis we will begin with a brief introduction to the importance of natural language processing, with a specific emphasis on coreference resolution. Following that, we will discuss the problem that this thesis aims to solve and the research question we seek to answer. Finally, we will explain the structure of this work.

1.1 Motivation

In recent years, the awareness and use of Natural Language Processing (NLP) has increased significantly. This trend can be observed in the global market value of NLP, which is expected to exceed 127 billion U.S. dollars by 2028 (Fortune Business Insights, 2022). One reason for this rise is the emergence of voice and chat assistants such as Amazon's Alexa or Google Assistant, which utilize NLP technology. In addition, Natural Language Processing (NLP) tools such as ChatGPT by OpenAI[1], which utilizes a large language model (Markovski, 2023), and the machine-learning based translator DeepL[2] have found widespread application in various industries.

To create NLP systems, computers need to process text and extract information from it. Texts can contain various types of information, one of which is the coreference of words. In a text, multiple persons or objects can appear multiple times, and specific words can refer to a particular person or object. In general, these persons or objects are called entities, and when two words or phrases in a text refer to the same entity, they are called coreferent. (Crystal, 2008) The computational methods that can extract all entities and their occurrences from a text are called coreference resolution. (Ng, 2010)

As demonstrated by Stojanovski and Fraser (2018), this information can be applied to natural language processing tasks such as machine translation. Azzam et al. (1999) also showed that it can be used in text summarization. Initially, rule-based algorithms were used for coreference resolution, but in the last two decades, machine-learning based approaches have become more prevalent. Recent models like the two architectures by Schröder et al. (2021) use end-to-end trained neural networks. However, these two introduced models have some limitations. The former model does not scale well for long texts due to the memory requirements, while the latter provides worse results than the first one but can scale well. Moreover, there is currently no satisfactory solution for evaluating long documents, such as whole books, with these models.

[1] https://chat.openai.com/
[2] https://www.deepl.com/translator

Therefore, the purpose of this thesis is to propose different approaches to utilize the two models by Schröder et al. (2021) to predict entities in text of arbitrary length and evaluate their performance. By doing so, we aim to overcome the limitations of the existing models.

1.2 Problem Description

As mentioned in the previous section, Schröder et al. (2021) adapted two machine learning models for coreference resolution in the German language. However, both models have some limitations. The coarse-to-fine model's memory usage grows quadratically as the text length increases, whereas the incremental model uses only a constant amount of memory but does not perform as well as the coarse-to-fine model. The main issue with the existing models is that there is no model that can process documents of arbitrary length while using a limited amount of memory and performing well. To address this issue, we propose a hybrid approach that combines the strengths of the coarse-to-fine model and the incremental model.

Our approach involves splitting long documents into shorter segments, predicting the entities using the coarse-to-fine model, and then merging them using one of four approaches. Three of these approaches are rule-based, while the fourth involves adapting the existing incremental model using a neural approach.

Our research question can be stated as follows:

> Can splitting long documents, predicting mentions and entities independently using the coarse-to-fine model, and merging them afterwards improve the performance of existing coreference resolution systems while requiring only a limited amount of memory?

The hypothesis for this thesis is that predicting splits using the coarse-to-fine model and merging entities together will perform equally or better than the existing incremental model, while also using limited memory.

1.3 Structure of the Work

We will start this work by presenting all theoretical background information necessary to understand the work in this thesis. The background section will include detailed explanations of coreference and coreference resolution, brief introductions to machine learning techniques used in this work, and an explanation of the metrics used to evaluate the performance of the proposed solutions.

The related work section will focus on existing coreference resolution systems and their architecture and functionality. We will specifically examine the two models by Schröder et al. (2021) that form the direct basis for this work.

The following methodology chapter will provide information about the datasets used to train the underlying models and evaluate the proposed approaches. The main part of this chapter will be dedicated to introducing the four merging approaches proposed in this work. We will start with the simpler rule-based approaches and progress to the more complex hybrid solution that adapts the incremental model.

Finally, we will evaluate all the proposed methods in detail. We will begin by providing an overview that can be used to compare the results to the baseline, followed by separate experiments that evaluate different settings of the proposed solutions. The thesis will conclude by answering the stated research question and providing an outlook on possible future work that could improve the results of the proposed approaches.

2 Background

This chapter serves as a comprehensive introduction to the fundamental concepts and techniques that are indispensable to comprehend the problem and proposed solution of this thesis. The focus is primarily on the essential knowledge required to understand coreference and coreference resolution, as well as the approaches used to process data, such as machine learning and cosine similarity, and represent data using word embeddings. The final section of this chapter will detail the methods used to evaluate the performance of a coreference resolution system

2.1 Coreference Resolution

Coreference is a part of lingustics and is defined by Crystal (2008, p. 116) as a "term used in linguistics, to refer to constituents in a sentence that have the same reference". A constituent is further defined as "a linguistic unit which is a functional component of a larger construction" (Crystal, 2008, p. 104). In other words a constituent is a single or a group of words which form a unit. Coreference resolution is a task in Natural Language Processing (NLP) where programmatically all mentions that point to an entity are found and partitioned into the same equivalence class, where in each class all mentions refer to the same physical entity. In this thesis we will follow the naming convention that has been introduced in the Automatic Content Extraction (ACE) task (Doddington et al., 2004) and so each individual phrase, which can consist of an arbitrary amount of words, is called a mention and each equivalence class is called an entity. (Luo, 2005)

Denn $[Alexander]_0$ erwiederte so von ganzem Herzen die Liebkosungen des $[Kleinen]_1$, sprach $[sich]_0$ so warm und unverholen über $[seine]_1$ herrlichen Anlagen, über $[sein]_1$ tiefes Gemüth aus, daß $[Linovsky]_2$ erfreut, von den Lippen eines $[Fremden]_0$ bestätigt zu hören, was die eigene Ueberzeugung $[ihm]_2$ oft zugeflüstert hatte, ein inniges Behagen an der Gerechtigkeit fand, die $[seinem]_2$ $[Otto]_1$ widerfuhr.

Figure 2.1: Excerpt of the book “Erna” by Charlotte von Ahlefeld and part of the Deutsches Roman Corpus (Krug et al., 2018)

Given the example sentence in Figure 2.1, all italic and colored words are mentions and all words with the same color refer to the same three physical entities Alexander, Otto and

Linovsky. Per definition all words with the same color form an entity and so the sentence contains a total of three entity. A special kind of entity can be seen in the sentence in Figure 2.2. The mention "Erna" has no other mention that it refers to (given that only the one sentence is seen as the context) and is therefore referred to as a "singleton". Singletons can be defined as "a cover term for mentions that are never coreferent [...] and mentions that are potentially coreferent but occur only once in a document" (Kübler and Zhekova, 2011, p. 261).

Da sah $[ihn]_0$ $[Erna]_1$ an mit einem Blick, dessen reine Klarheit, obwohl von Mitleid getrübt, $[ihn]_0$ hoch empor über allen irrdischen Kummer hob.

Figure 2.2: Sentence from the same document as Figure 2.6, that contains a singleton

Coreference resolution is used as a technique in many NLP tasks to improve the results of these tasks. For instance Azzam et al. (1999) used it as part of text summarization. Text summarization involves constructing a representation of the source text and generating a summary from it. Coreference resolution is applied to model the summary representation of the text. The best coreference chain is selected as the summary representation, based on the assumption that the text revolves around a central entity, which serves as the topic or focus of the discourse. Another use case has been introduced for example by Krishna et al. (2017). They made use of coreference resolution to identify on what exactly customers refer to in their review. Two other fields of application are question answering (Bhattacharjee et al., 2020) or machine translation (Stojanovski and Fraser, 2018).

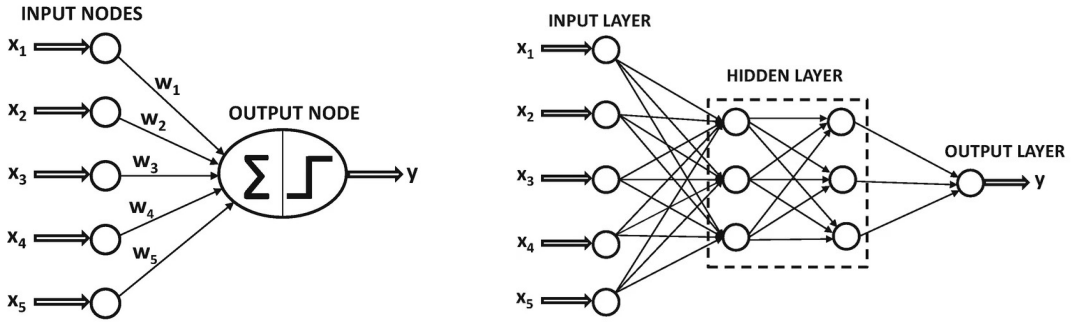
2.2 Machine Learning

According to Zhou (2021, p. 2), machine learning is "the technique that improves system performance by learning from experience via computational methods". In other words, it learns from multiple example cases or, more generally, data. The primary objective of machine learning is to create a model capable of solving a given task. This model is built solely by processing data, a step referred to as learning or training. If the required output of the model is present alongside the input in the training data, it is considered labeled. Machine learning models can be learned in two ways: supervised or unsupervised. The former uses labeled data, while the latter uses unlabeled data. A model that learns without labeled data and subsequently generates the labels itself is called self-supervised (Mikolov et al., 2013a).

2.2.1 Feed Forward Neural Network

One of the most popular machine learning techniques is artificial neural networks, which are inspired by the way the biological brain works. These artificial networks, called neural networks in the following, consist of neurons (sometimes called nodes) that are in-

terconnected with weights, just like the neurons in the human brain are connected using synapses (McCulloch and Pitts, 1943). Neurons are the processing nodes of the neural network. They sum up all the weights of the active neurons that they are connected to, and calculate their own output by processing this sum through an activation function (Sazli, 2006). The simplest way to connect neurons is called a perceptron, which can be seen in Figure 2.3a. It consists of one input layer and one neuron as an output node. Only the output processes any values. Neurons inside one layer are not interconnected, but only connect to nodes in other layers.



(a) Single Perceptron (b) Feed-forward neural network

Figure 2.3: Architecture of an exemplary perceptron with only one input and output layer and an exemplary feedforward neural network with 2 hidden layers. (graphics taken from Aggarwal (2018))

If more layers are added to the neural network, they are called multilayer neural networks. When the output of a node is only transported to nodes in a layer closer to the output, the network is referred to as a feed-forward neural network (FFNN). A FFNN consists of at least three layers: one input layer, one output layer, and at least one additional layer in between both. In the example FFNN in Figure 2.3b, there are two fully-connected additional layers, each consisting of three neurons (Aggarwal, 2018). These layers are called hidden layers because the calculations are not visible to the user at the output nodes. The number of hidden layers defines the depth of the network, which is why they are also often referred to as deep feedforward neural networks or just deep neural networks (Goodfellow et al., 2016).

2.2.2 End-to-End Learning

When it comes to using machine learning to solve a task, there are multiple approaches that can be taken. Machine learning approaches like decision trees, random forest, k-nearest neighbor, or even multiple neural networks can be used, depending on the specific problem at hand. Complex tasks may require multiple layers, and each layer may require a different machine learning approach. This can result in errors that add up across the layers and also requires specific knowledge for each layer. (Roza, 2019)

To address these challenges, a more complex deep neural network can be used instead

of multiple machine learning algorithms. In this approach, each layer of the neural network can specialize in solving a sub-task, resulting in only one model being learned. This training process is known as end-to-end learning, and it is widely used in many fields which rely on machine learning (Glasmachers, 2017).

End-to-end learning has been shown to be highly effective in NLP tasks, reducing the need for engineers to create task-specific solutions and not relying on deeper prior knowledge. Collobert et al. (2011) demonstrated the effectiveness of this approach, highlighting its ability to solve many NLP tasks through a single, end-to-end trained model.

2.3 Word Embeddings

Many tasks in Natural Language Processing require words to be represented in a way that machines are able to work with them. One of the best ways to achieve this is by converting the characters and whole words to real-valued vectors with multiple hundreds of dimensions. These vectors also encode the syntactic and semantic word relationships, so that words that are similar are placed near each other in the vector space (Almeida and Xexéo, 2019). Words that are dissimilar are placed far away in the vector space. Furthermore, simple algebraic operations with the vector representations of the words are possible. Thus, it is possible to identify a word that bears a similar semantic relationship to "small" as "biggest" does to "big". In a well trained embedding space, the vector of smallest should be the nearest on to the computed vector of vec("biggest") - vec("big") + vec("small"). These two characteristics of the embedding space also allow to calculate the similarity between words or even words clusters. This is for example achieved by calculating the angle between two word vector with the cosine similarity. The smaller the angle between the two vectors, the greater the similarity between the two words.

These real-valued vectors are called word embeddings. By definition word embeddings are dense, distributed, fixed-length word vectors (Turian et al., 2010) that can for example be built using word co-occurrence statistics as per the distributional hypothesis. The distributional hypothesis states that words which are similar in meaning also occur in similar contexts (Harris, 1981).

The word embeddings which are used in this thesis and are explained in detail in the following chapters are generated using self-supervised learning of neural networks. This means that the model is trained without annotated training data but rather by providing are large data set of texts. These word embedding models are also called prediction-based models because they use language models which predict the following word given a surrounding context (Baroni et al., 2014).

2.3.1 Word2vec

Word2vec was introduced in 2013 by Mikolov et al. (2013a). It is not a single algorithm, but rather a family of model architectures that can be used to learn word embeddings

from large datasets. It allows for two methods of learning vector representations for given words: continuous bag-of-words (CBOW) and continuous skip-gram model. As seen in Figure 2.4 the former model predicts the middle word based on the surrounding context words, which form the input for the neural network. The order of the words does not matter, hence the name "Bag-of-Words".

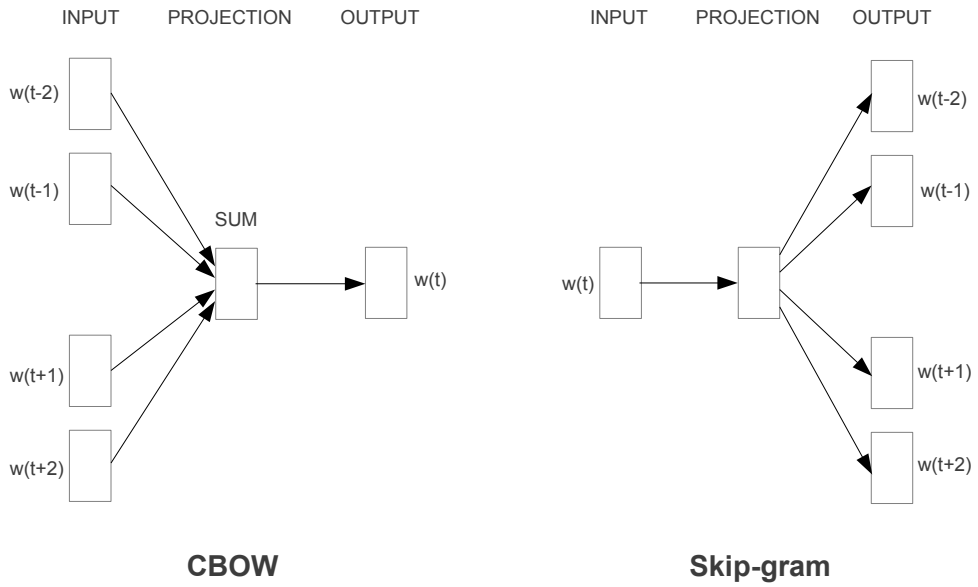


Figure 2.4: Architectures of the CBOW and the Skip-gram model (graphic taken from (Mikolov et al., 2013a))

The second model is the inverse of the first one. It takes one word vector as input and predicts the words within a given context range before and after the current word in the same sentence. Mikolov et al. (2013b) demonstrated that this model achieves better results than the CBOW approach. It is also superior to previous neural network approaches because it does not rely on dense matrix multiplications, which makes the method extremely efficient. Training time also depends on the number of context words. More context words result in more training examples, which can lead to higher accuracy, but at the cost of longer training times.

The training objective of the skip-gram model is to maximize the probability of predicting context words given a target word. This probability is increased by adjusting the numbers in the input vector (or multiple input vectors if the CBOW method is used) if the predicted word is not correct. As a result, words with similar contexts are brought closer together in the embedding vector space.

In this thesis, we use a model by deepset, that has been pre-trained for the German language based on a corpus of the German Wikipedia[1]. The model is trained with a

[1] https://gitlab.com/deepset-ai/open-source/word2vec-embeddings-de

window size of five and outputs real-valued vectors with a length of 300.

2.3.2 FastText

The biggest disadvantage of using word2vec embeddings is that they do not take into account the internal structure of words. Instead, each word in the vocabulary is represented by a distinct vector. This means that words which are not present in the training dataset, or are only rarely seen, may not have a good embedding vector assigned to them, or even none at all.

FastText by Bojanowski et al. (2017) is an extension of the word2vec model that takes into account the internal structure of words. To achieve this, each word is represented as a bag of character n-grams. The word is first prefixed and suffixed by the symbols < and >, respectively, to distinguish between prefixes or inner parts of the word. Then, it gets broken down into n-grams. For example, if $n = 3$, the word "where" will be represented by <*wh, whe, her, ere, re*>, and also the word in total, <*where*>. A vector representation is learned for each n-gram, and the word in total is represented by the sum of these n-gram vectors.

The experimental results indicate that the fastText model generally outperforms the word2vec skip-gram and CBOW approaches, and consistently performs better when computing vectors for out-of-vocabulary words. In particular, for Arabic, Russian, and German, the fastText model produces better results than the word2vec model (Bojanowski et al., 2017). As this thesis focuses only on German datasets, it is expected that using fastText word embeddings will lead to improved results in the corresponding merging approach.

For this thesis, we used a by Grave et al. (2018) pre-trained model for the German language that utilizes the Common Crawl dataset and a Wikipedia dump[2]. The model employs the CBOW method with a window size of 5 and character n-grams of length 5.

2.3.3 BERT

One known problem with the word2vec and fastText embedding models is that they only produce static vectors stored in a simple lookup table. This means that every known word is assigned exactly one vector in the embedding space. As a result, these models do not represent the existence of polysemy well, which is the fact that the same word can have multiple meanings. Often, these multiple meanings are a result of the context in which the word is used, but this context is not encoded in the embedding space representation (Peters et al., 2018).

Nowadays, models such as BERT, which was introduced by Devlin et al. (2019), solve this problem. BERT is an acronym for Bidirectional Encoder Representations from Transformers. BERT is based on the machine learning technique of transformers, which was

[2] `https://fasttext.cc/docs/en/crawl-vectors.html`

introduced by Vaswani et al. (2017). The transformer model uses an encoder-decoder architecture that can process sequential data, such as natural language. The embeddings created by this model are contextualized by employing an attention mechanism during the encoding and decoding phases, which is the specialty of the transformer model. BERT is a further development of ELMo (Embeddings from Language Model) (Peters et al., 2018), which also provided context-sensitive word embeddings. So this model produces different vectors for words that have the same spelling but a meaning that depends on the context. BERT is thus a bi-directional encoder, because the model is pre-trained by "reading" the sentences from the left to the right and from the right to the left, so that the context is taken into account in both directions.

The framework is built in two steps: a pre-training step and a fine-tuning step (Devlin et al., 2019). The pre-training step creates a general knowledge base using a large set of unlabeled data. Therefore, this step is rather computationally expensive because it takes a substantial amount of processing time, but only needs to be done once. This pre-training step has two training objectives. The main objective is a "masked language model" (MLM). A random percentage of input words are masked and the goal is to predict the original vocabulary ID of the masked words based on the left and right surrounding context. The second objective is called Next Sentence Prediction and is used to pre-train the understanding of the relationship between two sentences. Here two sentences A and B are given, where only 50% of the time B is the correct next sentence. The other 50% are random sentences from the corpus. The goal is to predict whether B is the correct following sentence. This pre-training objective is necessary because many NLP tasks require an understanding of the relationship between two sentences, which is not given by the language model alone. The second step of the framework is the fine-tuning phase, where the parameters of the pre-trained model are adapted to task-specific topics. This step is not as costly as the pre-training, and therefore the BERT model can be adjusted to the desired downstream task at low cost.

2.4 Cosine Similarity

One way to determine the similarity between two documents or clusters of words is by using the cosine similarity method (Singhal, 2001). This mathematical technique calculates the angle between two vectors. As seen in Equation 2.1, the cosine similarity score is retrieved by calculating the dot product of the vectors and dividing it by the product of the length of both vectors. The resulting similarity score is independent of the magnitude of the word vectors.

$$\text{cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{2.1}$$

The resulting decimal numbers ranges from 0 to 1, since the numbers in the word embedding vectors are never negative. A cosine similarity score of 0 indicates that the

vectors are orthogonal to each other, while a score of 1 means that they point in the same direction (Kotu and Deshpande, 2019). This can also be applied to the similarity of the word clusters. A score of 1 indicates that both clusters are identical, while a cosine similarity value of 0 indicates maximal dissimilarity between them.

2.5 Evaluation Metrics

In order to evaluate and compare the approaches proposed in this thesis with each other and with existing coreference solutions, we need to find a suitable way to score the performance of our system. However, this is not a simple task for coreference resolution systems because there is no straightforward way to count "correct" and "wrong" answers due to the complexity of the coreference task. As stated by Luo (2005), a metric needs to achieve two important properties: discriminativity and interpretability. The former refers to the score's ability to differentiate "good" systems from "bad" systems based solely on the score, while the latter means that the value should be easily understandable with human sense only. For instance, if a score of 95% is given, one would expect that most mentions in a text have been correctly identified and classified into the correct entity.

As shown in Figure 2.5, creating a single metric that satisfies these requirements without any drawbacks is a challenging task. While several metrics have been proposed to address this issue, they all focus on different aspects simultaneously. The CoNLL-F1 score, introduced during the CoNLL shared task by Pradhan et al. (2012), is a combined metric that averages three different F1 scores: the B-Cubed score, a mention-based score; the MUC score, a link-based score; and the CEAF metric, an optimal mapping-based score. All these metrics provide a way to calculate both precision and recall values, which can then be combined to calculate the F1 score by taking the harmonic mean of both.

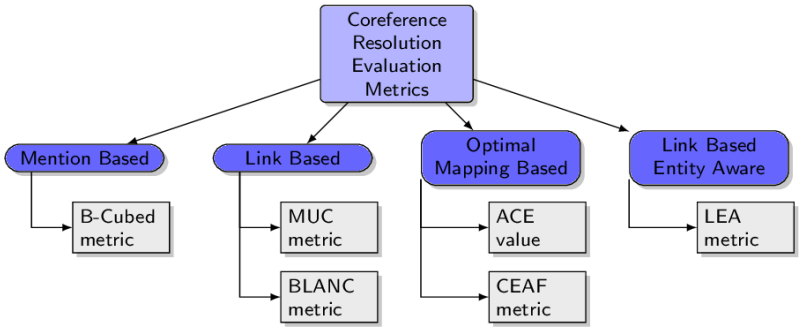
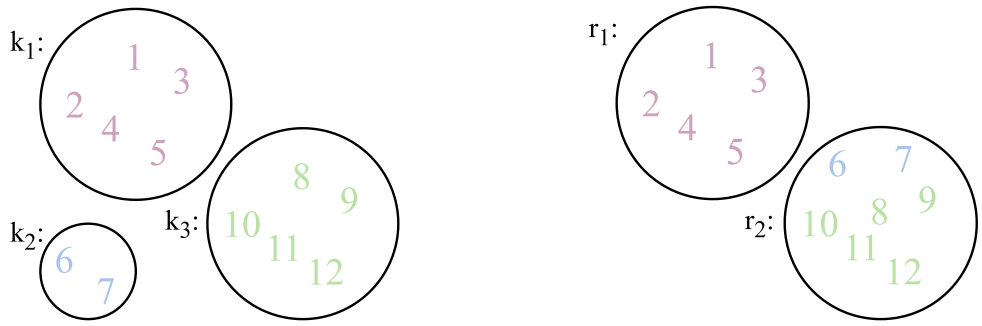


Figure 2.5: Classification of metrics that can be used for evaluation of coreference system (graphic taken from (Sukthanker et al., 2020))

In the following subsections, we will explain in detail how these three scores are calculated. To provide a better understanding, we will use an example to calculate a recall and precision value. This will also demonstrate how strongly the values can vary even on a relatively simple example. The example mentions and entities for the calculation of the recall are depicted in Figure 2.6. The annotated entities are referred to as the "key"

or "gold" entities and are considered the ground truth. On the other hand, the entities produced by a coreference system are called "response" entities and are shown on the right side of the figure. In the following, the set of entities from the key is denoted by K, and the set of entities from the response is denoted by R. Furthermore, k_i and r_i denote individual entities from these sets, respectively.



(a) Key entities set

(b) Response entities set

Figure 2.6: An example of three entities, where each number represents a mention. The color of the number indicates the true clustering, whereas the circles show the resulting clustering of the algorithm.

2.5.1 MUC

The MUC score was introduced by Vilain et al. (1995) at the 6th Message Understanding Conference, hence the name. It is sometimes also referred to as the MUC-6 score. The scores compare the number of links in a response set to the number of links in a key entity set. The recall is calculated as follows:

$$Recall = \frac{\sum_{k_i \in K} (|k_i| - |p(k_i)|))}{\sum_{k_i \in K} (|k_i| - 1))} \tag{2.2}$$

To evaluate a system's entity linking performance, we count the number of mentions in each entity in the key or gold set and subtract the number of partitions found in the response sets. For a given entity in the key set, which consist of multiple mentions, we count how many entities in the response set contain those mentions. For example, if the key set generates only one entity $k_1 = \{A, B, C, D\}$, but the response set only includes the relation $< A - B >$, the partition of $p(k_i)$ would be $\{A, B\}$, $\{C\}$, and $\{D\}$. This difference is then divided by the minimal number of links required to create the key set. In a perfect result, all mentions would be in the same entity in the response, requiring only one link, hence $|k_i| - 1$ as the denominator. The precision is hereby calculated by switching the key and response entities.

Based on the example in Figure 2.6, the number of partitions for all three entities in the gold/key data would be one, since they all intersect with only one entity in the response set. Since this is also the minimum number of required links, the recall is 100%. To calculate precision, the partition for the two entities in the response must be deter-

mined. For S_1, the resulting partition is $\{1,2,3,4,5\}$, while for the entity S_2, the partition is $[\{6,7\},\{8,9,10,12,12\}]$. This results in a precision of 90%.

$$\textit{Final Precision} = \frac{(5-1)+(7-2)}{(5-1)+(7-1)} = \frac{9}{10} = 0.9 \tag{2.3}$$

$$\textit{Final Recall} = \frac{(5-1)+(2-1)+(7-1)}{(5-1)+(2-1)+(7-1)} = 1.0 \tag{2.4}$$

2.5.2 B-cubed

The B-cubed scoring algorithm, introduced by Bagga and Baldwin (1998), examines the presence or absence of mentions relative to each of the other mentions in the same entities that were produced. This results in the calculation of both precision and recall. Precision describes how many of the found mentions are correctly placed in this entity, while recall describes the number correct entities that have been found. Specifically, for every mention i, precision and recall are defined as follows:

$$\textit{Precision}_i = \frac{\textit{number of correct elements in the entity containing mention}_i}{\textit{number of elements in the response entity containing mention}_i} \tag{2.5}$$

$$\textit{Recall}_i = \frac{\textit{number of correct elements in the entity containing mention}_i}{\textit{number of elements in the key entity containing mention}_i} \tag{2.6}$$

The final precision and recall of the entire document are calculated as the weighted sum of the individual recall and precision values, as shown in Equation 2.7 and Equation 2.8. The weight can be chosen based on the task being addressed. In our case, every mention is equally important, so every mention is assigned the same weight of $1/N$, where N is the total number of mentions found in the document.

$$\textit{Final Precision} = \sum_{i=1}^{N} w_i * \textit{Precision}_i \tag{2.7}$$

$$\textit{Final Recall} = \sum_{i=1}^{N} w_i * \textit{Recall}_i \tag{2.8}$$

Given the example clustering of mentions into entities shown in Figure 2.6, recall and precision can be calculated as shown in Equation 2.9 and Equation 2.10. The recall of 100% means that for every mention, every other mention that is coreferent to it is also in the same entity. However, the recall only achieves a value of 75% because only 2 of 7 and 5 of 7 mentions in the second and third gold entities, are correctly clustered in the algorithm's response.

$$\textit{Precision} = \frac{1}{12} * \left(\frac{5}{5}+\frac{5}{5}+\frac{5}{5}+\frac{5}{5}+\frac{5}{5}+\frac{2}{7}+\frac{2}{7}+\frac{5}{7}+\frac{5}{7}+\frac{5}{7}+\frac{5}{7}+\frac{5}{7}\right) = 0.75 \tag{2.9}$$

$$Recall = \frac{1}{12} * \left(\frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{2}{2} + \frac{2}{2} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5} + \frac{5}{5}\right) = 1.0 \tag{2.10}$$

2.5.3 CEAF

The CEAF metric, introduced by Luo (2005), calculates a best alignment of entities from a key set to entities from a response set, hence its name, Constrained Entity Alignment F-Measure. To find the best alignment, a matching problem in a maximum bipartite graph is formulated. For every entity in the key set and in the response set, a vertex is added to create the graph. All pairs (k_i, r_i) of nodes, where k_i is an entity from the key set and r_i is an entity from the response set, are connected via an edge with the weight of $\phi(k_i, r_i)$. The Kuhn-Munkres algorithm (Kuhn (1955) and Munkres (1957)) can then be used to solve the maximum bipartite graph problem in polynomial time and find an optimal mapping of the entities. The function g^* represents the one-to-one mapping, returning an entity from the response set when given an entity from the key set as input.

$$\phi_3(k_i, r_j) = |k_i \cap r_j| \tag{2.11}$$

$$\phi_4(k_i, r_j) = \frac{2 * |k_i \cap r_j|}{|k_i| + |r_j|} \tag{2.12}$$

The weights of the edges in the graph use a similarity metric $\phi(k_i, r_i)$, where k_i and r_i are entities. The metric should be large when both entities share most of the mentions and small when they are different. Based on the similarity metric ϕ, two different types of CEAF can be defined: ϕ_3 Equation 2.11, also called mention-based CEAF, and ϕ_4 Equation 2.12, called entity-based CEAF (Moosavi and Strube, 2016). To calculate the recall and precision, the maximum total similarity Φ Equation 2.13 is first calculated using the previously found best entity alignment g^*. We assume here that K^* are all entities from the key set, that are part of the found optimal mapping. The final recall and precision are calculated using the entity self-similarity $\phi(k_i, k_i)$ where $k_i \in K$, representing the number of entities in the key set is used for recall. For precision, the self-similarity $\phi(r_i, r_i)$ equals the number of entities in the response set.

$$\Phi(g^*) = \sum_{k_i \in K^*} \phi(k_i, g^*(k_i)) \tag{2.13}$$

$$Precision = \frac{\Phi(g^*)}{\sum_{r_i \in R} \phi(r_i, r_i)} \tag{2.14}$$

$$Recall = \frac{\Phi(g^*)}{\sum_{k_i \in K} \phi(k_i, k_i)} \tag{2.15}$$

To calculate the CEAF ϕ_3 and ϕ_4 precision and recall scores using the key and response

entities shown in Figure 2.6, we need to identify their one-to-one mapping. The key entity $\{1,2,3,4,5\}$ perfectly matches the response entity $\{1,2,3,4,5\}$ and is therefore mapped. Additionally, the key entity $\{8,9,10,11,12\}$ is aligned to $\{6,7,8,9,10,12,12\}$. Since each response entity can only be mapped once, the entity $\{6,7\}$ remains unaligned. Once we have established the mapping, we can calculate the maximum total similarity for both the ϕ_3 and ϕ_4 similarity functions.

$$\begin{aligned}\Phi_3 &= \phi_3(\{1,2,3,4,5\},\{1,2,3,4,5\}) + \phi_3(\{8,9,10,11,12,13\},\{6,7,8,9,10,11,12\}) \\ &= |\{1,2,3,4,5\}| + |\{8,9,10,11,12\}| \\ &= 5+5 \end{aligned} \tag{2.16}$$

$$\begin{aligned}\Phi_4 &= \phi_3(\{1,2,3,4,5\},\{1,2,3,4,5\}) + \phi_3(\{8,9,10,11,12,13\},\{6,7,8,9,10,11,12\}) \\ &= \frac{2*|\{1,2,3,4,5\}|}{5+5} + \frac{2*|\{8,9,10,11,12\}|}{5+7} \\ &= 1+\frac{5}{6} \end{aligned} \tag{2.17}$$

To obtain the actual recall and precision scores, we need to divide the Φ values by the self-similarity score of the key entities for recall and the response entities for precision. In the case of the ϕ_3 similarity function, the self-similarity score corresponds to the number of mentions across all entities, which is why it is called the mention-based CEAF. As seen in Equation 2.18 and Equation 2.19 for ϕ_3 recall and precision both equal $\frac{5}{6}$.

$$\begin{aligned}Precision_{\phi_3} &= \frac{10}{\phi_3(\{1,2,3,4,5\},\{1,2,3,4,5\})} + \frac{10}{\phi_3(\{6,7\},\{6,7\})} \\ &\quad + \frac{10}{\phi_3(\{8,9,10,11,12\},\{8,9,10,11,12\})} \\ &= \frac{10}{5+2+5} = \frac{5}{6} \approx 0.83 \end{aligned} \tag{2.18}$$

$$\begin{aligned}Recall_{\phi_3} &= \\ &= \frac{10}{\phi_3(\{1,2,3,4,5\},\{1,2,3,4,5\})} \\ &\quad + \frac{10}{\phi_3(\{6,7,8,9,10,11,12\},\{6,7,8,9,10,11,12\})} \\ &= \frac{10}{5+7} = \frac{5}{6} \approx 0.83 \end{aligned} \tag{2.19}$$

The precision and recall functions used in the entity-based CEAF score are the same as those used in the mention-based score, except that the ϕ_4 similarity function is used.

In Equation 2.20 and Equation 2.21, we omit displaying the entities for the similarity function and instead show the result of the ϕ_4 function to simplify the calculation.

$$\begin{aligned} Precision_{\phi_4} &= \frac{1+\frac{5}{6}}{\frac{2*5}{5+5}+\frac{2*7}{7+7}} \\ &= \frac{1+\frac{5}{6}}{2} = \frac{11}{12} \approx 0.92 \end{aligned} \tag{2.20}$$

$$\begin{aligned} Recall_{\phi_4} &= \frac{1+\frac{5}{6}}{\frac{2*5}{5+5}+\frac{2*2}{2+2}+\frac{2*5}{5+5}} \\ &= \frac{1+\frac{5}{6}}{3} = \frac{11}{18} \approx 0.61 \end{aligned} \tag{2.21}$$

3 Related Work

This chapter introduces the work that this thesis is based on and discusses the current state of existing coreference resolution solutions. Firstly, a brief overview of the progress in the development of coreference resolution is provided. Secondly, two models used in the field, a coarse-to-fine model and an incremental model, are delved into in deeper detail. Finally, we will explain which problems still exist in the presented models for coreference resolution, for which we want to present possible solutions in this thesis.

3.1 Existing Coreference Resolution Solutions

Coreference resolution has been a core task of natural language processing for a long time. The first rule-based approach was introduced by Hobbs (1978) with the Hobbs' naïve algorithm. In the 1990s, a paradigm shift occurred from heuristic and statistical approaches to machine-learning-based coreference resolution. The first learning-based solution by Aone and Bennett (1995) used decision trees, and today's state-of-the-art approaches outperform rule-based ones by far. Ng (2010) published a survey about the first years of machine learning-based coreference resolution research and clustered the different models into three categories: mention-pair model, entity-mention model, and ranking models.

Mention-pair models consist of a classifier that determines whether two mentions are coreferent, like the first introduced decision tree approach by Aone and Bennett (1995). This approach entails the problem that the transitivity property of the coreference relation cannot be enforced. The decisions are made independently, so two mentions that are marked as coreferent to a third mention may not be marked as coreferent to each other. This problem has been tackled by the entity-model. Here a machine learning model classifies whether a mention is coreferent with a preceding cluster of entities. The model by Schröder et al. (2021) that this thesis is based on can be categorized as a ranking model. For a given mention, all possible antecedents are scored and the best candidate is chosen.

They adapted two existing end-to-end trained models for the German language, taking advantage of the fact that neural approaches can be easily transferred to different languages. These models, and thus the focus of this thesis, are on German datasets, which will be further introduced in section 4.1. The architecture of both models can be seen in Figure 3.1, and will be described in the following section, along with a brief overview of the research progress that led to these models.

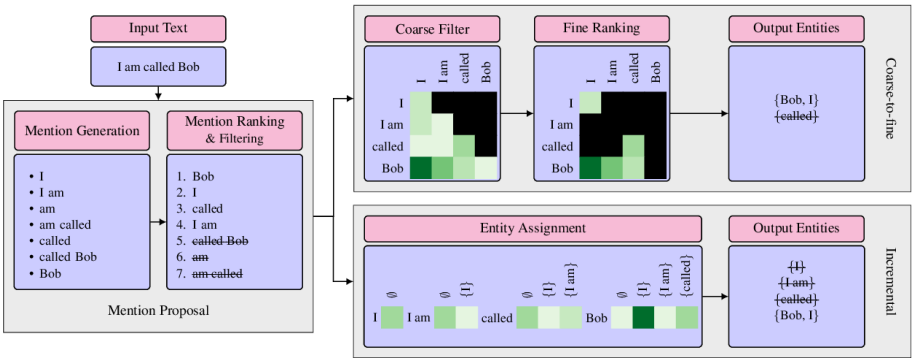


Figure 3.1: The Figure provides an overview of how the two end-to-end neural models process an input sentence to retrieve a list of entities as output. (graphic taken from (Schröder et al., 2021))

3.1.1 Coarse-to-Fine Model

Lee et al. (2017) introduced the first model that learned an end-to-end approach using only gold mention clusters. Unlike previously introduced models, it does not rely on syntactic parsers. Instead, the model learns which spans are entity mentions and can cluster them based on their coreference.

Given an input document D containing T words, the model's basic task is to assign an antecedent span y_i to each of the $N = \frac{T(T+1)}{2}$ possible spans. The possible assignments for each y_i are $\{\epsilon, 1, ..., i-1\}$, which include a dummy antecedent ϵ and all preceding spans. An assignment to an antecedent j (everything except ϵ) indicates a coreference between span i and span j. A dummy antecedent means, that the span is not a mention at all, or that the mention is not coreferent to any previously seen span in the text.

The model aims to learn, for every mention span x, a probability distribution over every possible antecedent spans Y.

$$P(Y) = \frac{e^{s(x,y)}}{\sum_{y' \in Y} e^{s(x,y')}} \tag{3.1}$$

Hereby $s(x, y)$ is the calculated coreference score. As seen in Figure 3.2 and Equation 2.1 the coreference score is calculated using the mention score s_m and the antecedent score s_a. The mention score indicates the likelihood that a span is a mention, while the antecedent score indicates the likelihood that spans x and y refer to the same entity. Both scores are retrieved using a feed-forward neural network (FFNN), which takes a vector representation of the span as input and returns the corresponding score as output. In the original paper by Lee et al. (2017), a bidirectional Long Short-Term Memory (LSTM) was used in conjunction with ELMo and GloVe word embeddings to encode lexical information. Joshi et al. (2019) further improved this by replacing the LSTM-based encoder with the BERT transformer.

$$s(x, y) = s_m(x) + s_m(y) + s_a(x, y) \tag{3.2}$$

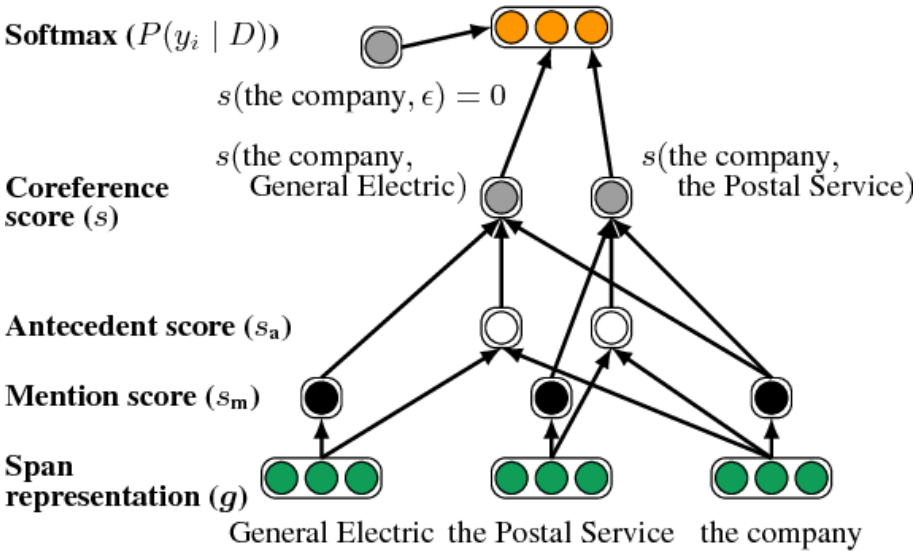


Figure 3.2: Calculation the coreference score as a sum of the mention scores and the antecedent score (graphic taken from (Lee et al., 2017))

The problem with this approach is, that computing the antecedent score is challenging because it requires computing a tensor of size $M * M * (3|g| + |\phi|)$, where M is the number of previous mention spans, $|g|$ is the size of one embedding vector and $|\phi|$ is the size of meta-informations (like for example information about the speaker). To reduce the computational size, a "coarse-to-fine" approach has been proposed by Lee et al. (2018). They introduced an alternate bilinear scoring function, which requires less computational power.

$$s_c(i,j) = g_i^T W_c g_j \tag{3.3}$$

W_c is a weight matrix learned using a feed-forward network. This scoring function is less accurate than the previous score, resulting in a performance loss of over 3 F1. However, it only requires matrices of size $M * |g|$ and $M * M$. This new, efficient scoring function is used in the coarse-step of the model. Up to M spans are kept based on their mention score, and the top K antecedents of all remaining span i are kept based on the sum of their mention score and the lightweight coarse-score.

$$s_m(i) + s_m(j) + s_c(i,j) \tag{3.4}$$

In the model architecture by Schröder et al. (2021), that this thesis is directly based on, K is limited to $min(4096, 0.4 * |D|)$, where $|D|$ is the number of words in the document.

Up to a maximum of the top 64 antecedents per mention are further processed in the following Fine Ranking step, as shown in Figure 3.1. This step sums up the computationally expensive coreference-score $s(i,j)$ and the coarse-score $s_c(i,j)$ (Equation 3.3) from the previous coarse step. Based on this final score, the probability distribution will be calculated, and the assignments of antecedents for all mentions will be decided. The resulting

chain of antecedents will be used to group together all mentions that belong to the same entity and these will be the output entities.

3.1.2 Incremental Model

The previously described coarse-to-fine model has limitations when processing long documents. The issue is that even when using the coarse step, it requires simultaneous access to all spans $\Theta(n)$ for a document with length n and all scores $\Theta(n^2)$. Additionally, the size of the vector representations of the span embeddings increases as the dimensionality of contextualized encoders may increases with further research. As a result, the required memory for long documents can exceed the available resources when they are limited.

To address this, Schröder et al. (2021) adapted a model whose underlying idea was introduced by Xia et al. (2020) and Toshniwal et al. (2020). Instead of scoring all possible mention spans against each other, a growing list of entities is created incrementally, where each possible mention span is scored against all available clusters.

The first step, the Mention Proposal, is taken from the coarse-to-fine model, as seen in Figure 3.1. Retrieving the top possible mention spans uses the same scoring function, and following Xia et al. (2020), the weights from the coarse-to-fine model are reused. As stated earlier, the clustering of the mentions to the corresponding entities does not require scoring of all mention spans against each other. Instead, it will work in an incremental manner, as seen in Figure 3.1, where Schröder et al. (2021) call this step "Entity Assignment".

Listing 3.1 shows the code for the `FindClusters` function that processes this step. Each document is split into segments, and a growing list of entities is created during the iteration of all segments. For each mention out of the mention proposal step, a score is calculated using a feed-forward neural network. This network takes as inputs the embedding of the mention span and the representations of the existing entity clusters. These representations of the entities are updated every time a new mention is added to the entity, using a weight that has been learned through a FFNN.

The weight network takes the concatenation of the current representation of the entity cluster with the representation vector of the added mention span as input. The result is then mapped to a value between 0 and 1 using the sigmoid function, as shown in Equation 3.5. The resulting weight vector is used to update the representation of the entity cluster as a weighted sum of the existing representation and the embedding of the newly added mention (Equation 3.6).

$$\alpha = \sigma(FFNN([e_{top\_e}, e_m])) \tag{3.5}$$

$$e_{top\_e} \leftarrow \alpha * e_{top\_e} + (1 - \alpha) * e_m \tag{3.6}$$

The `FindClusters` algorithm adds every mention to an existing entity or creates a new

entity to hold the mention. The only way to discard mentions, either because they are not true mentions or because they are singletons, is to remove all entities that have only one mention in a post-processing step. However, removing singleton clusters eliminates the possibility of finding them, as they cannot be distinguished from non-mention entities. To address this, an additional "dummy" cluster can be added to the scoring function where non-mention entities can be added. In the post-processing step, only the non-mention entities are removed, and singleton mentions that belong to an entity without a referent are retained.

As the list of entities grows, so does the required amount of memory needed to store them. In order to keep the required memory amount constant after each segment iteration, entities will be evicted using the `EVICT` function (Listing 3.1, line 15). This function removes entities from the list used for the scoring function, but keeps them on the CPU so that they persist until the end of inference. Entities that will be removed are chosen based on their cluster size and their distance from their last appearance to the current position in the document.

```
1  FindClusters(Document):
2     Create an empty Entity List, E
3     for segment ∈ Document do
4        M ← SPANS(segment)
5        for m ∈ M do
6           scores ← PAIRSCORE(m, E)
7           top_score ← max(scores)
8           top_e ← argmax(scores)
9
10          if top score > 0 then
11             UPDATE(top_e, m)
12          else
13             ADD_NEW_ENTITY(E, m)
14
15       EVICT(E)
16    return E
```

Listing 3.1: Pseudocode for incremental entity assignment step (Xia et al., 2020)

3.2 Remaining Problems of Coreference Resolution Solutions

For both models the results are still not optimal and two major problems remain. The most significant problems of the coarse-to-fine model is the requirement of a matrix of $n \times n$, where n is the number of possible spans in the document. This leads to a quadratic growth in the required memory proportional to the length of the document. To reduce the required memory, the coarsing step was introduced, but the proportional growth remains. To address this issue, the incremental model was introduced, which reduces

the required memory and eliminates the quadratic dependency. The incremental model even reduces the required memory by using the evict step to maintain a constant memory requirement. This can be seen in Figure 3.3, where the orange markers show the required amount of memory from the incremental approach proposed by Xia. Earlier models, such as the coarse-to-fine model, had an increasing required amount of memory, whereas the incremental model remains constant.

Figure 5.1 illustrates two major problems related to the performance of both models. Firstly, the incremental model performs worse than the coarse-to-fine model across all document lengths. Secondly, the performance of both models decreases as document length increases. The optimal system should achieve the same or better performance as the coarse-to-fine model, while using constant memory like the incremental model. To address this issue, this thesis proposes adapting both models introduced in this chapter.

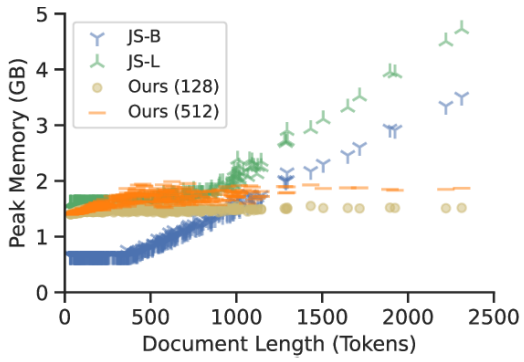


Figure 3.3: Required amount of allocated tensors based on their document length (graphic taken from Xia et al. (2020))

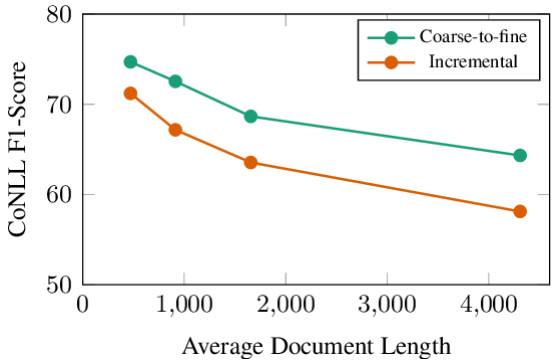


Figure 3.4: CoNLL-F1-score for both models for increasing document length (graphic taken from Schröder et al. (2021))

4 Methodology

In the previous chapter, we explained that the coarse-to-fine model generally outperforms the incremental model for coreference resolution. However, its memory requirements are quadratic to the document length, limiting its usefulness for longer documents. To address this limitation, this thesis proposes a neural method that combines the strengths of both models.

The concept is based on the premise to split long documents into shorter ones and use the coarse-to-fine model to predict mentions and corresponding entities in the splits. Subsequently, the incremental model will merge the entities of the splits together. In this chapter, we introduce this neural method and three other simpler methods based on pre-defined rules.

All methods create a growing list of entities by iterating over the split document and merging the entities of the current split into the entities that have already been seen. The entities of the current split are called "local" entities, while the entities that have already been merged are "global" entities. Each document is processed independently, thereby maintaining the document-level scope of the term "global"

This chapter will begin with introducing the dataset used to train the neural models and evaluate the merging methods. Before introducing the proposed merging methods, an explanation of how the data is pre-processed will be provided.

4.1 Datasets

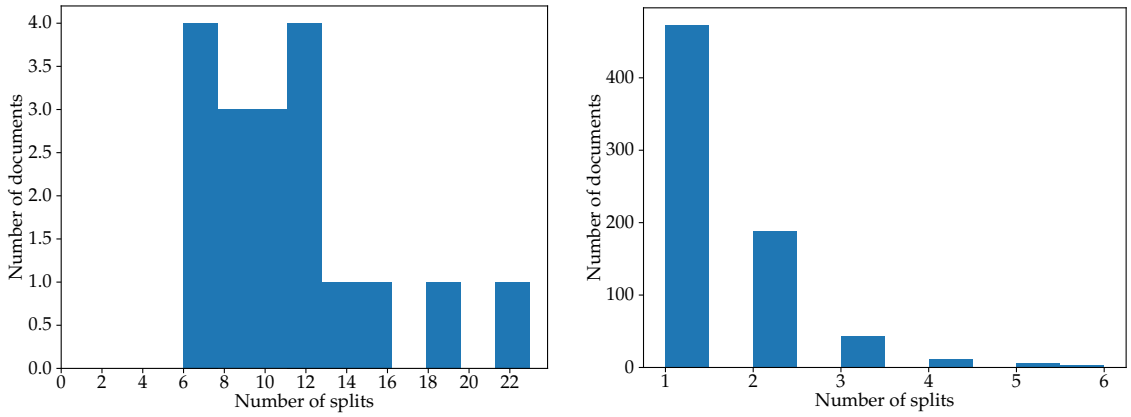
To evaluate the different merging approaches, we will use two datasets containing German-language documents with manually annotated entities. Although none of the proposed merging approaches rely on unique features of the German language, we will use these sets since they were also used to train and evaluate the models introduced in section 3.1, which this work is based on. This allows us to compare our results to existing solutions without the need to retrain the models.

The first dataset is the "Tübinger Baumbank des Deutschen / Zeitungskorpus" (TüBa-D/Z), published by Telljohann et al. (2017) from the University of Tübingen. This corpus exists in different versions, where the latest one contains 3,816 manually annotated articles from the German newspaper "die tageszeitung". The data is given in the CoNLL-2012 format, which was introduced by Pradhan et al. (2012) for the CoNLL (Conference on Natural Language Learning) shared task of 2012. This format defines the de facto standard for datasets used in NLP. To establish a comparison with the baseline, we will

employ version 10 of this dataset, despite the existence of more recent versions. Version 10 was used in the SemEval-2010 shared task and therefore provides a state-of-the-art comparison for the paper that this thesis is based on (Recasens et al., 2010).

The second dataset is the Deutsches Roman Corpus (DROC) by Krug et al. (2018). It contains 90 literary documents from German novels. Unlike TüBa-D/Z, only the mention heads are annotated in DROC, meaning that "der kleine Max" would form a mention in TüBa-D/Z, whereas only "Max" would be annotated in the DROC dataset. This results in a shorter average mention span length in the DROC data. Additionally, only references to characters are annotated in the DROC dataset, and mentions of other entities are not part of the gold data.

Although only character references are annotated, the documents contain an average of 575.52 mentions on the DROC dataset, whereas on the TüBa-D/Z dataset, there are an average of 10.89 entities per article, with 3.65 mentions per entity. The main reason for this is that the documents in the DROC dataset are considerably longer, with an average length of 3,945.39 tokens, compared to the TüBa-D/Z set, which has an average of 347.53 tokens. As such, the TüBa-D/Z dataset can be divided into fewer splits. Figure 4.1 shows the number of splits with a maximum length of 512 tokens for both corpora. All documents in the DROC dataset can be split into at least six splits, whereas most of the documents in the other dataset cannot be split when using 512 tokens as the maximum length. We will exclude from the evaluation all documents that cannot be split into at least two splits because no merging step would be required for them.



(a) DROC (b) Version 10 TüBa-D/Z[1]

Figure 4.1: Amount of splits using a maximum split length of 512 tokens

4.2 Data Preparation

In order to utilize one of the merging algorithms or the existing coarse-to-fine and incremental model, preprocessing of the textual data provided by the datasets is necessary. As

[1]For a better overview, an outlier document with 21 splits is not shown.

the existing models and pre-processings steps by Schröder et al. (2021) and their predecessors are implemented in Python, the code for this thesis has also been realised in it. In order to maintain a better overview and comprehensibility, code examples are written in Python-like pseudocode throughout the work. Most of the pre-processing, like converting the input to the ConLL-2012 format and further into an input format for the neural networks or splitting the words into sub-tokens, has already been done in the work on which this thesis is based on. Two major steps still need to be done before the merging algorithm can be applied:

1. Splitting the long documents into shorter sub-documents
2. Predicting the mentions and entities in the splits using the coarse-to-fine model

The second step is only necessary if entities in unlabeled documents need to be identified, or if merging algorithms are to be tested using the coarse-to-fine model. If the sole purpose is to evaluate merge quality, the gold mentions within the splits can be used, and the prediction step is not necessary.

The splitting method needs to handle three different parameters to support all proposed merging algorithms: the length of each split (maximum number of sub-tokens per split), whether the splits should be overlapping, and if so, how long the overlap should be. It is crucial to consider the length parameter as a maximum value rather than a fixed length. The reason is that splitting sentences should be avoided whenever possible, as it can result in the model losing valuable contextual information, ultimately causing its performance to suffer. As a result, the split may be shorter than the maximum number of sub-tokens, which can lead to some variation in the length of the splits. However, the hard cut-off is the more important criterion because the coarse-to-fine model has a maximum length that it can work with.

The sentences in the document will be added iteratively, and the index of the last added sentence will be stored in a counter variable. The index of the next added sentence will be the value of the counter variable plus one. This approach ensures that each sentence is added precisely once to its corresponding split in the default configuration. If the overlapping parameter is active, the sentence counter will be decreased by the length specified by the overlapping-length parameter after creating each split. Consequently, every split, except the first one, will include the last sentences from the preceding split.

Another sub-step that needs to be done during the splitting process is changing some consecutive indices. These indices must be realigned relative to the split start, rather than being absolute values for the entire document. Attributes in the data that require index alignment include for example the `subtoken_map`, which contains the index of the word in the document that each sub-token belongs to, or the `sentence_map`, which contains the index of the sentence that each sub-token belongs to.

Once the splitting and realigning of data has been completed, each split can be regarded as an independent document. This allows the coarse-to-fine model to be used

to predict mentions and entities without any modifications. If the merge method needs to be tested independently of entity prediction, the gold clusters from the dataset can be copied into the corresponding split. In this case, the mention indices must also be adjusted accordingly.

4.3 Rule-Based Approaches

In the following three sections, we introduce three approaches for merging clusters of two splits together if they belong to the same entity. As explained previously, the underlying idea is that the coarse-to-fine model is used to identify mentions and belonging entities in the split. Then, the clusters are merged using our proposed merging methods so that the whole document is analyzed afterwards. These methods are based on predefined rules rather than (directly) on a neural network.

Each approach will be explained in detail and illustrated with an example merging step.

4.3.1 String-Based Merging

The first rule-based method is based on the following basic idea: if two entities of consecutive splits share the same word, and no other entities do so as often, then they are likely the same entity.

This idea is realized through an iterative approach over the splits created in the document. The iteration is done in order. For the first split, nothing needs to be merged, and each entity will be copied to the document-wide set of entities. For every following split, each entity will be checked separately. As illustrated in Figure 4.2, for every mention of the “to-be-merged” entities, the appearances in the already existing global entities will be counted. The string that appears most often in any existing cluster will determine the merge.

There are three possible outcomes when checking if an entity will be merged with an existing one:

1. There is exactly one other entity in the already merged ones, that shares a string with "to-be-merged" entity more often than any other string in any other entity. Those two entities are probably the same one, and consequently will be merged.

2. There are two or more entities that share the same string the same number of times as the "to-be-merged" entity, and no other string is shared more often. As an example, let’s consider the string "ihr" appearing in Figure 4.2. It exists in two entities within the current split once and also in two global entities once. In this case, both clusters are equally viable for merging. Since we cannot determine a definitive winner, the entity will not be merged and will instead be added as a new entity in the global document.

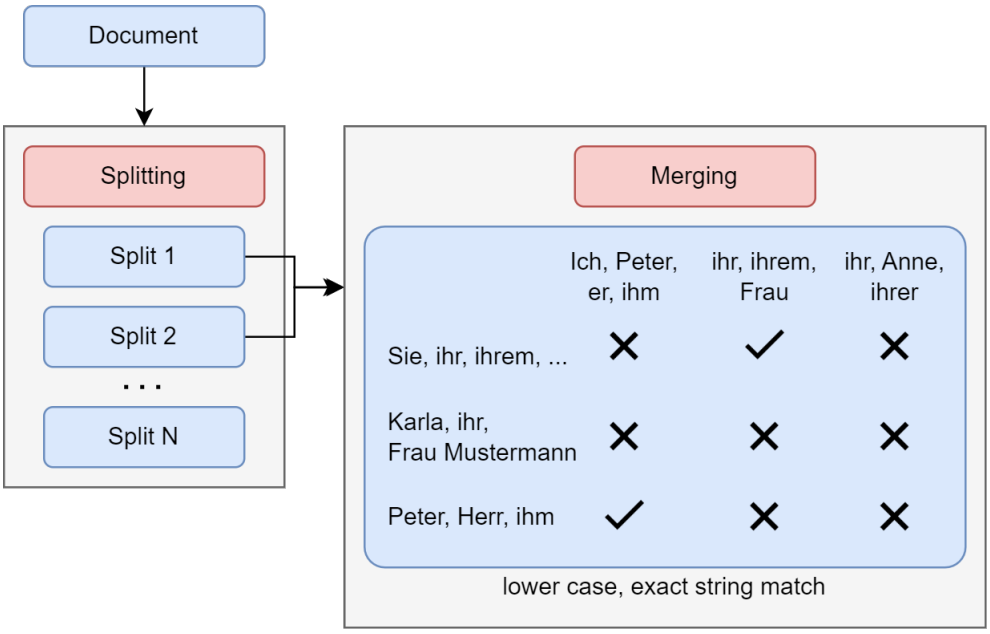


Figure 4.2: Conceptual visualisation of the string matching approach

3. No strings are shared between the "to-be-merged" entity and any other already existing entity. We cannot determine any entity that is probably related to this one, so the merge is skipped. Like in scenario 2, the entity is added as a new one.

To clarify the merging algorithm, we will explain it based on the entities and corresponding mentions from the first two splits of the document "Erna" by Charlotte von Ahlefeld from the DROC corpus (Krug et al., 2018), which are displayed in Table 4.1 and Table 4.2. Initially, all entities from the first split are taken over into the document-wide entities, as there are no entities to merge. In the following, all entities that have already been merged are referred to as "global entities".

Table 4.1: Entities and corresponding mentions of the first split of document "Erna". The index column displays the index of the entitiy in the entire unsplit document.

Index	Mention-Strings
0	"Alexander", "ihm", "Herr von Norbeck", "Alexander", "Herr von Norbeck", "ihm", "er", "seinem", "seines", "sich", "er", "Alexander", "ihn", "seinen", "sein", "ihn", "Freund", "seines", "seinen"
1	"ich", "wer", "Linovsky", "Ich", "er", "sein", "er", "sich", "Gemahls", "er", "sein", "ihm", "er", "Herr von Linovsky"
2	"meine", "dir", "Erna", "Erna", "ich", "ich", "sie", "ihr", "du", "ich", "Erna", "sie"
3	"seines", "Wundarzt", "er", "Wundarzt"
4	"Kinder", "Kindern"
5	"uns", "unserm"

In the first iteration step, the goal is to find a matching entity for the first entity of the second cluster. The first string to be checked is "seiner". This string does not occur in any

Table 4.2: Entities and corresponding mentions of the second split of document "Erna"

Index	Mention-Strings
0	"seiner", "ihm", "Alexander", "ihn", "sich", "ihm", "seines", "seinem", "ihm", "sich", "seine", "Alexander", "dieser", "seinen", "ihm", "seine", "ihm", "seinen", "Leidenden", "er"
1	"Linovsky", "sein", "er", "Linovsky"
2	"ihrer", "sie", "Sie", "sie", "sie", "sich", "Erna"
3	"er", "Arztes", "der", "seiner", "Arzt", "Chirurgus"

entity of the global entities and is therefore not relevant for the merge.

Next, the string "ihm" is checked. This string occurs in entity 0 and 1 of the global entities, with a maximum appearance count in entity 0. The same appearance counts are checked for all remaining strings of the first entity of the second split. The results can be seen in Table 4.3, where the "best matching cluster" is the index of the entity in the global entities where the string appears most often. The maximum number of appearances in this entity can be seen in the second column, and the total number of entities in which the string occurs is shown in the last column.

In this example, the merge-deciding mention/string is "alexander" because it only appears in exactly one existing entity and appears more often than "ihn", "seinem", and "seinen". Therefore, the first entity of the second split is merged with the first entity of the global entities, which is indeed the correct merge.

Table 4.3: Results of the merging algorithm for the first entity of the second split

Mention-String	"seiner"	"ihm"	"alexander"	"ihn"	"sich"	"seines"	"seinem"	"seine"	"dieser"	"seinen"	"leidenden"	"er"
best matching cluster	-1	0	0	0	0	0	0	-1	-1	0	-1	1
max. appearance count	0	2	3	2	1	2	1	0	0	2	0	4
clusters with match	0	2	1	1	2	2	1	0	0	1	0	2

For all other entities in the second split, the merging process will be similar. Entity 1 will be merged with entity 1 based on the word "linovsky," and entity 2 will be merged with entity 2 based on the word "erna." As for the last entity, the previously described second outcome will occur. The string "er" appears twice in entity 0, four times in entity 1, and once in entity 3. Since no other words are found in any global entity, no entity can be declared as the best match. Therefore, nothing will be merged, and the entity will be taken over as a new one in the global entities.

After merging the first two splits, there are a total of seven global entities, and three out of the four entities were merged correctly. This process will now be repeated for all other existing splits.

4.3.2 Overlapping Merging

We propose a second approach for merging entities, which is based on overlapping splits. The underlying idea (like in all other approaches) is that the document is not predicted in its entirety, but rather we have a moving window of the document where the mentions and entities are predicted. If in two overlapping splits, the same string (at the same position) is marked as a mention, we assume that it is likely that they belong to the same entity. These two entities will then be merged together to one. Because the window is larger than only the overlapping part, the entities grow when the window is moved forward. Two mentions are marked as overlapping if they share at least one token at the same global index. For example, consider the string "die Landesvorsitzende Ute Wedemeier". If the first split includes all four words as part of the mention, but the second split only marks "Ute Wedemeier" as a mention, the two splits would still be considered intersecting. This can also be seen in Figure 4.3, where the mention `[17, 18]` of the first split and the mention `[18, 18]` are considered overlapping, even though only the second token is marked in the second split. Therefore, these two entities likely belong to the same global entity and should be merged.

As described in section 4.2, the splits in this approach always contain the last segments of the previous split and all following segments until the maximum number of tokens is reached.

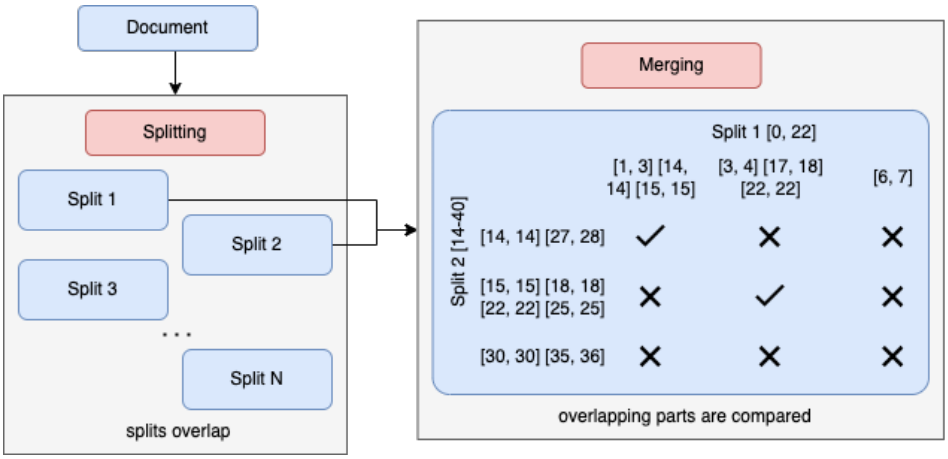


Figure 4.3: Conceptual visualisation of the overlapping approach

Entities will always be merged with the entity they most frequently intersect with. This is necessary because it is possible for entities to collide in the overlapping part of the split. For example, in the first split, there are three mentions that all belong to entity A. In the overlapping part of the second split, all these mentions are also present, but two belong to entity A and one is categorized as entity B. In this situation, entity A will be merged because the amount of overlap is larger, and it is more likely that the one mention is misclassified. However, entity B will not be merged into entity A, given that it does not intersect with any other entity, because once an entity was merged into a global entity, it

will not be merged again during the same merging step. Instead, the entity will be added as a new one to the global entities.

An example of the overlapping approach can be seen in Figure 4.4, which shows two splits of the first document. In this example, entity 4 intersects ten times in the overlapping split and would therefore be merged. Although only one mention overlaps in entity 0, both clusters will be merged. The same applies for entity 5. In the case of clusters 0 (Erna) and 4 (Linovsky), the example shows that new mentions have been added to existing clusters because of the overlap. Cluster 2 does not appear in the overlapping section of the two splits and thus cannot be merged. A new global entity would be added, containing the mentions "Wundarzt" and "der".

Ruhig sagte $[Erna]_0$: Weder die $[Kinder]_1$ noch $[ich]_0$ bedürfen $[seines]_2$ Beistandes, aber einen Freund von $[uns]_3$ hat nicht weit von $[unserm]_3$ Hause ein Unfall betroffen, der zwar, dem Himmel sei Dank, nicht bedeutend ist, aber dessen Behandlung denn doch $[meine]_0$ wenigen medicinischen Kenntnisse übersteigt.
Auch dauerte es einige Minuten, während $[er]_4$ $[sich]_4$ mit $[Erna]_0$ in leisem Gespräch in eine Fenstervertiefung zurückgezogen hatte, wohin $[sein]_4$ Blick $[ihm]_4$ nicht folgen konnte, bis $[er]_4$ zu $[seinem]_4$ Lager trat und $[ihm]_4$, als einen der Höflichkeit nicht gern dargebrachten Zoll, einige halb unverständliche Worte von dem Vergnügen, $[ihn]_4$ wieder zu sehen und von der Freude, daß $[sein]_4$ Haus gerade das nächste bei dem sich ereigneten Unfall gewesen sei, um $[ihm]_4$ als Zufluchtsort dienen zu können, zumurmelte.
$[Alexander]_5$ stellte $[sich]_5$ kränker als $[er]_5$ war, um diese Gemeinplätze nur nicht beantworten zu müssen.

Auch dauerte es einige Minuten, während $[er]_4$ $[sich]_4$ mit $[Erna]_0$ in leisem Gespräch in eine Fenstervertiefung zurückgezogen hatte, wohin $[sein]_4$ Blick $[ihm]_4$ nicht folgen konnte, bis $[er]_4$ zu $[seinem]_4$ Lager trat und $[ihm]_4$, als einen der Höflichkeit nicht gern dargebrachten Zoll, einige halb unverständliche Worte von dem Vergnügen, $[ihn]_4$ wieder zu sehen und von der Freude, daß $[sein]_4$ Haus gerade das nächste bei dem sich ereigneten Unfall gewesen sei, um $[ihm]_4$ als Zufluchtsort dienen zu können, zumurmelte.
$[Alexander]_5$ stellte $[sich]_5$ kränker als $[er]_5$ war, um diese Gemeinplätze nur nicht beantworten zu müssen.
In $[Linovsky's]_4$ frostigen Mienen, und den feindseligen Blicken, mit denen er $[ihm]_4$ gegenüber stand, spiegelte sich ein Theil $[seiner]_4$ eigenen Gesinnung, und der herbeigerufene $[Wundarzt]_2$, $[der]_2$ eben herein trat, unterbrach sehr willkommen die Spannung dieses nicht wohlthuenden Beisammenseyns.
$[Erna]_0$ hatte bereits Charpie und Leinwand zum Verband zurecht gelegt.

Figure 4.4: Excerpt from the DROC dataset (Krug et al., 2018) split into overlapping splits

4.3.3 Embedding Space Merging

The third proposed rule-based approach makes use of word embeddings introduced in section 2.3. Specifically, we utilize the feature that allows algebraic operations on embeddings. The basic idea is to calculate an embedding vector for all entities in a split, and then use cosine similarity to compare all mentions of the entities to all entities that have already been processed in previous splits. Entities that exceed a given threshold for similarity will be merged together, as they likely refer to the same entity.

As shown in Figure 4.5, we will split the documents into smaller pieces and predict them separately, as in all proposed approaches. In this approach, the splits will not overlap. The merging process will iterate over all splits in order, and for every entity of the split, it will either merge with an already existing "global" entity (an entity that has been processed in a previous split merging step) or create a new global entity. For the first split, all entities are simply taken over because nothing can be merged there.

For each subsequent split, we will use word embeddings of entity mentions. For every mention, we will obtain a vector that represents the word in the embedding space. To obtain these embeddings, we will make use of pre-trained models of word2vec (Mikolov et al., 2013a) and fastText (Bojanowski et al., 2017). Two things must be considered when obtaining these embeddings. First, mentions can consist of multiple words, but word embeddings only exist for individual words. If we encounter mentions with more than one word, we will split them at the whitespace and treat them as two independent mentions. Second, when using the word2vec model, not all mentions may have word embedding vectors. This can be the case if the words did not appear in the training data that the word2vec model was trained on. Therefore, before retrieving all the word embeddings, we will need to filter out all words without a representation in the embedding space. As explained in subsection 2.3.2, this filtering is not required for fastText because it can generate embeddings for words not in the training data.

After obtaining all the embedding vectors, we will calculate the mean vector for each entity. This will result in a vector of dimensionality 300 for each entity, both for the split and the already merged ones. To compare these vectors, we will use cosine similarity, which is explained in section 2.4. The score ranges from 0 to 1, where 0 means that the entities are as different as possible, and 1 means that both entities are exactly the same. This step results in a matrix of scores where all split entities have a similarity score for all "global" entities.

To perform the actual merging step, we will iterate over all entities in the current split and merge them with the entity in the "global" entities that they have the highest score with, provided that two other conditions are true:

1. The similarity score exceeds the threshold.

2. No other entity has its highest score on this entity and has a higher score than the entity to be merged.

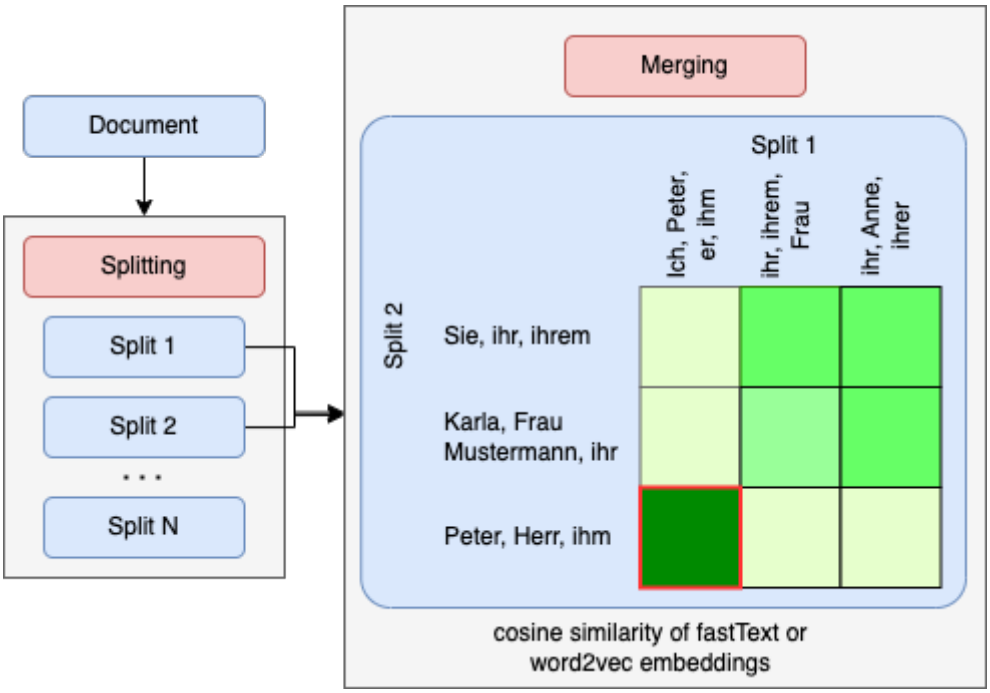


Figure 4.5: Conceptual visualisation of the embedding space approach

By following these two rules, we will always merge the entity with its best matching cluster, but we will not merge two entities of a split into the same global entity. We make this assumption because we assume that the predictions of the splits are perfect. If two mentions are not in the same entity in the split, they should not be in the same entity after the merge. This process is then repeated for every split to obtain all the predictions for the entire document.

Figure 4.6 displays an example matrix showing cosine similarity scores for entities from Table 4.1 and Table 4.2. Rows represent entities from the current split, while columns represent entities that have already been processed. The embeddings were calculated using the pre-trained word2vec model, and two entities are merged if their score exceeds the 0.8 threshold. In the filtering step, the mention "Linvosky" was removed from the second entity of the current split and also from the second entity of the "global" entities because it does not appear in the training data, and no embedding vector exists for the word when using word2vec. The entities that were merged are marked with a red border. The only entity that does not get merged is the third one of the entities of the current cluster. Even though the correct "global" entity has the highest similarity score, it does not pass the 0.8 threshold and is therefore not considered for merging. This entity will be added as a new one to the "global" entities.

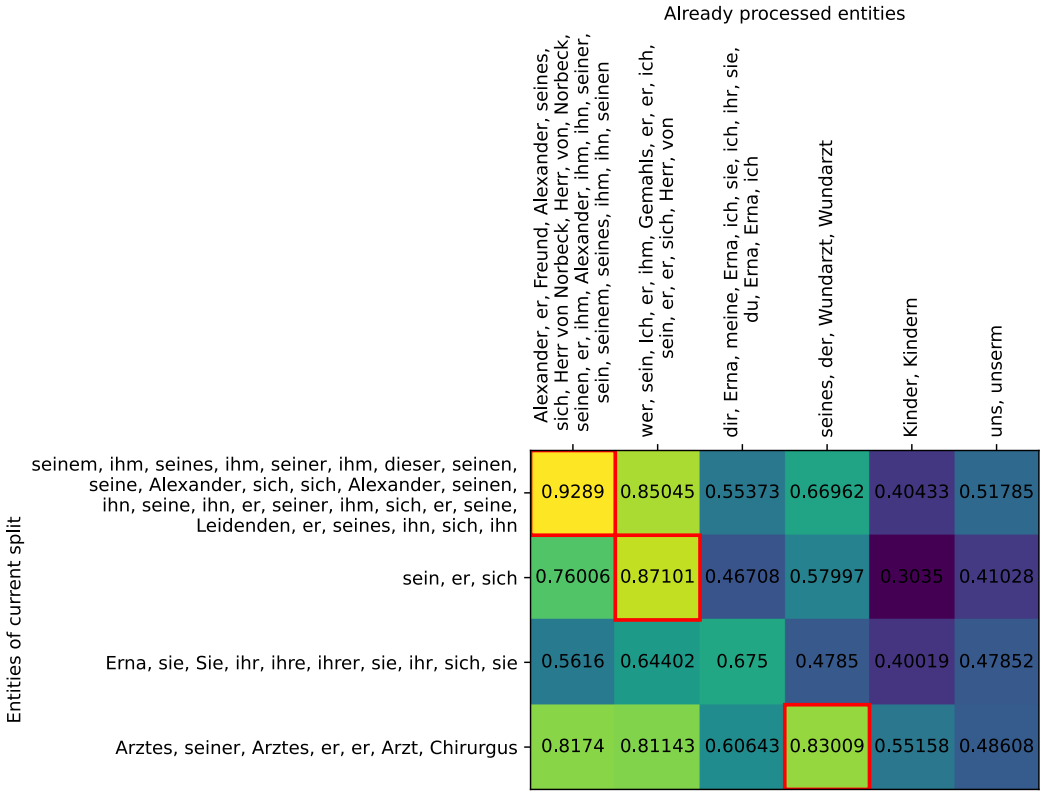


Figure 4.6: Example of cosine similarity matrix using word2vec and a threshold of 0.8. Entities that are merged are marked with a red border.

4.4 Neural Approach

In the final approach, we will use both existing neural models and create a hybrid solution to evaluate documents of arbitrary lengths. Like in all rule-based approaches, the document will be split into shorter parts, in this case, without any overlap. The coarse-to-fine model will be used to predict all mentions and cluster them into corresponding entities. These predictions will then be merged using the incremental model. The incremental model iterates over segments of the document and can handle split documents, as described in subsection 3.1.2. In this model, each mention is assigned individually to an existing entity or a new entity is created for the mention candidate. As shown in Figure 4.7, we will modify this algorithm so that instead of merging on a mention-level, it will merge based on whole entities. As such, instead of single mentions, complete entities will be assigned to existing entities.

As shown in Listing 4.1, we will iterate over all entities that have been found using the coarse-to-fine model (or gold entities in the evaluation) instead of iterating over all mentions of the split. Unlike the original incremental model, we will need to calculate the score for all entities of the current split before modifying any existing entities. This is because, like in all rule-based approaches, we see the predictions of the coarse-to-fine model as absolute. To prevent merging two entities of the same split into the same global

entity, we first need to calculate the score. Afterward, every entity gets merged into the entity that has the highest score, and no other entity has a higher score.

We want to reuse the weights of the already trained incremental model and cannot modify the structure of the input of the scoring function. To be compatible, we need to provide a representation of the to-be-merged entity as a single vector, together with the representation of all existing models. The latter representation does not need to be modified, as it already exists in that way in the original incremental model.

To create the representation of the to-be-merged entity, we propose two different solutions. In the first solution, we will calculate the mean of all embeddings of the mention span without any weighting. In the second solution, we will make use of the weighting introduced in subsection 3.1.2 for the representation of the existing entities. In the Listing 4.1 this can be seen in line 10 by using exemplary the `mean` function. We will iteratively calculate the weights for all mentions in the order of their appearance in the document and add them up as a weighted sum in the same way as the representation of the existing entities is created.

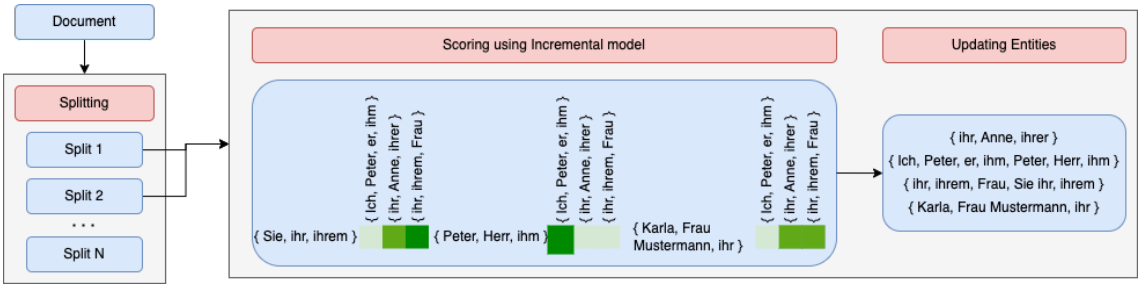


Figure 4.7: Conceptual visualisation of the adapated incremental approach

Listing 4.1: Pseudocode for modified incremental entity assignment step

```
MergeClusters(Document):
   Create an empty Entity List, global_E
   for split ∈ Document do
      local_E ← Coarse-To-Fine(split)
      all_scores ← []
      already_merged_entities ← []

      # Calculating the scores for all entities of the current split
      for e ∈ local_E do
         scores ← PAIRSCORE(mean(e), global_E)
         all_scores.append(scores)

      # Merging with the best global entity
      for e, index ∈ local_E do
         scores ← all_scores[index]

         top_score ← max(scores)
         top_e ← argmax(scores)

         # We will stop when an entity has been found in that
         # has not been merged, or creating a new entity
         # (top_e = 0) has the highest score
         while top_e in already_merged_entities and top_e > 0
            scores[top_e] ← -1 # Ignoring the entity from now on

            top_score ← max(scores) # Finding the next best entity
            top_e ← argmax(scores)

         already_merged_entities.append(top_e)
         if top score > 0 then
            MERGE_ENTITIES(top_e, e)
         else
            ADD_NEW_ENTITY(global_E, e)

      EVICT(global_E)
   return global_E
```

5 Evaluation

In this chapter, we will focus on the performance evaluation of our proposed approaches for merging entity clusters. Our goal is to determine the best performing approach and test our hypothesis. Additionally, we will investigate how adjusting different settings of the approaches affects performance. To achieve this, we introduce several experiments that target different aspects of the approaches.

The chapter begins by introducing the baseline for German coreference resolution, which serves as a reference point for our comparison. Subsequently, we explain the experimental setup and how we collect performance scores. Within the principal section of this chapter, we describe each experiment and highlight its significance. For each experiment, we present the resulting scores and provide a discussion of the outcomes.

All results will be presented using the evaluation metrics introduced in section 2.5, with the CoNLL-F1-score serving as the main metric for comparison.

5.1 Baseline

As explained in the previous chapters, in this thesis, we are comparing our modified neural models to the baseline introduced by Schröder et al. (2021), which is explained in section 3.1. We aim to determine if our modifications improve performance, building upon the existing work. Table 5.1 displays the CoNLL-F1-scores for both models on both data sets. The prefix "base" and "large" refer to the versions of ELECTRA used in the study. Specifically, the base model refers to the, by the German NLP Group pre-trained, GNG_ELECTRA model[1], while the large model is the GELECTRA model[2].

Table 5.1: CoNLL-F1-score of the incremental and coarse-to-fine model by Schröder et al. (2021) on TüBa-D/Z 10.0 and DROC datasets

Dataset	Model	CoNLL-F1-Score
TüBa-D/Z 10.0	Incremental	65.79
TüBa-D/Z 10.0	Coarse-to-fine (base)	77.21
TüBa-D/Z 10.0	Coarse-to-fine (large)	78.79
DROC	Incremental	64.72
DROC	Coarse-to-fine (base)	61.66

[1] https://huggingface.co/german-nlp-group/electra-base-german-uncased
[2] https://huggingface.co/deepset/gelectra-large

The table shows that the CoNLL-F1 score is higher for the TüBa-D/Z new data set, regardless of the neural model used. Moreover, it is noteworthy that in the DROC data set, the incremental model outperforms the coarse-to-fine model, whereas in the news data set, the incremental model is significantly behind (-13 F1) the coarse-to-fine model. This pattern can also be seen in Figure 5.1, which additionally indicates that both models exhibit decreased performance as document length increases.

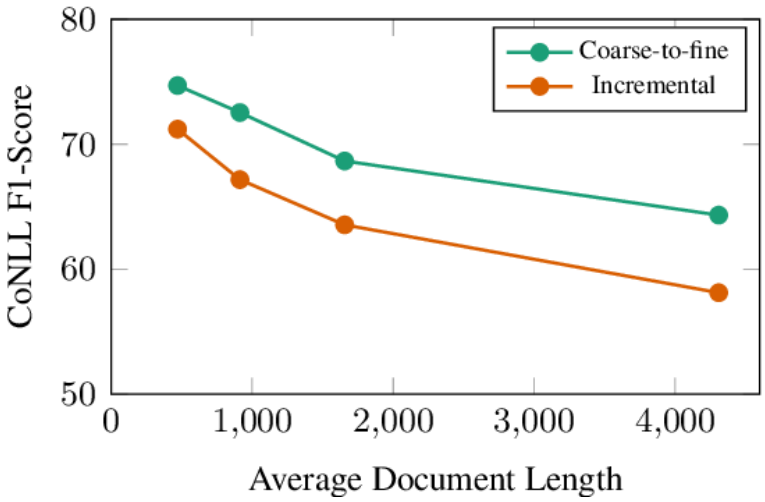


Figure 5.1: Performance of the coarse-to-fine and incremental model with increasing document length. (graphic taken from Schröder et al. (2021))

5.2 Experimental setup

In the following, we will explore various factors of the proposed merging approaches. However, we will focus on evaluating only one specific setting in all experiments, with different values for these settings. This approach allows us to observe how changes in these settings directly affect performance. We will use the data sets introduced in chapter 3.3 for evaluation purposes. As previously mentioned, the datasets differ significantly in the length of the individual documents. As a result, the information we gather from the news dataset is less relevant, as the documents are short, requiring only a few merging steps. For most experiments, we will focus primarily on the DROC dataset, as these documents can be split into more segments, requiring more merging steps.

To create the splits, we will utilize the method introduced in chapter 3.4. For each experiment, we will provide information on the maximum length of the splits. It should be noted that the splits will not all have the exact same number of tokens since a split will not break a sentence and will retain context. Therefore, the length of the splits should be viewed as a maximum number of tokens.

To compare the approaches against each other or against different settings, we will use the gold data provided by the datasets for evaluation purposes. By doing so, the results will not be influenced by the predictions of the neural model. When comparing

our approaches to the baseline, we will use the coarse-to-fine model to predict the splits. The base model is employed for the DROC dataset, whereas the large model is utilized for the news dataset. If a neural model is required, we will use the models trained by Schröder et al. (2021), which are publicly available on GitHub[3]. This will save us time and enable us to compare our results more precisely to the baseline.

5.3 Experiments

In the next section of this chapter, we will delve into the experiments that we conducted to evaluate the proposed merging approaches. We will explain each experiment in detail, present the results, and discuss the performance of the approaches. In total, we present five separate experiments.

5.3.1 Overview

The first experiment will also be the most important one, as it will determine whether we can accept or reject our hypothesis. The hypothesis of this thesis is that predicting splits and merging entities together will perform equally or better than the incremental model while also using limited memory. To prove or disprove this, we will provide an overview of the performance of all proposed approaches in this experiment. We will compare the approaches and determine which perform best and worst, and create a comparison with the baseline.

To create this overview, we will calculate the CoNLL-F1 score for both datasets using gold-labeled data and predicted splits from the coarse-to-fine model. We will use the settings that work best for both datasets and split them into a maximum length of 512 tokens. Although larger splits perform better, as shown in a later experiment, we chose this maximum length to evaluate the shorter documents of the TüBa-D/Z 10.0 dataset. However, not all documents in this set are long enough to create two splits, so we removed all documents that cannot be split into at least two splits and evaluated only on the remaining ones.

For the overlapping approach, we added the last two sentences of the previous split to create the overlap. The embedding approaches will utilize fastText and word2vec with the pre-trained values specified in section 2.3, and we will define a merging threshold of 0.9. In the neural approach, we create the representation of the to-be-merged entities by taking the mean of the mention embeddings. This approach performs better than the more complex weighted sum, as we have seen in pre-testing.

Figure 5.2 and Table 5.2 display the CoNLL-F1 scores for all methods used in this study on the TüBa-D/Z news dataset and the DROC dataset, respectively, when using gold entities. The results indicate that all methods perform better on the TüBa-D/Z news dataset than on the DROC dataset. One reason for this is that the documents in the

[3] https://github.com/uhh-lt/neural-coref/releases/tag/konvens

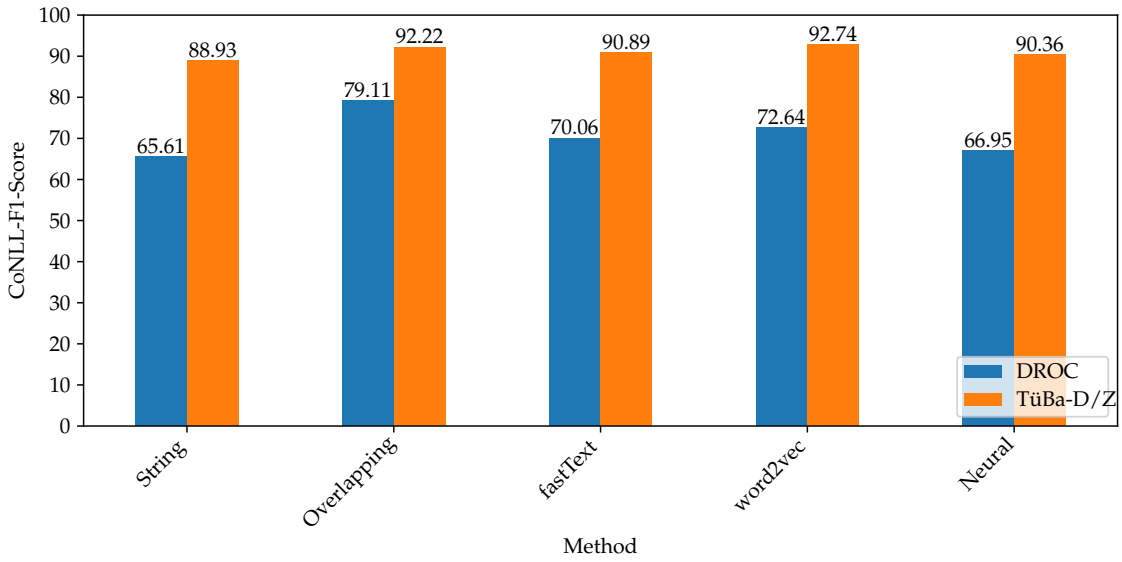


Figure 5.2: Performance of all approaches on the TüBa-D/Z 10.0 and DROC dataset. Using gold data and maximum split length of 512 tokens.

DROC set are much longer than those in the news set, as discussed in section 4.1. When using gold data, the F1-score can only decrease due to two reasons: incorrect merging of entities into a global entity or the creation of a new entity instead of merging an existing one. These errors are introduced in each merging step and accumulate over time. Longer documents result in more splits, which lead to more merging steps and, consequently, more errors. This trend is illustrated in Figure 5.3, which shows the average CoNLL-F1 score of all documents for each merging step, using the string-based merging approach. The news dataset contains shorter documents (we exclude one outlier document with 21 splits, because it would disproportionately affect performance) and therefore experiences fewer errors and a more gradual decline in F1-score than the DROC dataset. Furthermore, Figure 5.3 also indicates that the merging approach performs better on the news dataset in the beginning. This could be due to the fact that the news dataset has fewer entities on average and many of these entities only appear briefly in the text and do not require merging.

The overlapping method outperforms all other methods with F1-scores of 79.11 and 92.22 for the DROC and TüBa-D/Z 10.0 datasets, respectively. This is because, when using the gold data, every merge performed is correct, and errors are only introduced when an entity is absent in one split but appears later. This leads to the loss of context, resulting in the creation of a new entity instead of merging into the correct global one. This reasoning is further supported by the performance of the overlapping method when the coarse-to-fine model is used to predict the entities in the splits. In this case, the overlapping approach underperforms, with a CoNLL-F1-score of only 50.04 on the DROC dataset, making it the worst performing method, as seen in Figure 5.4.

The embedding approach using fastText performs slightly worse than word2vec on

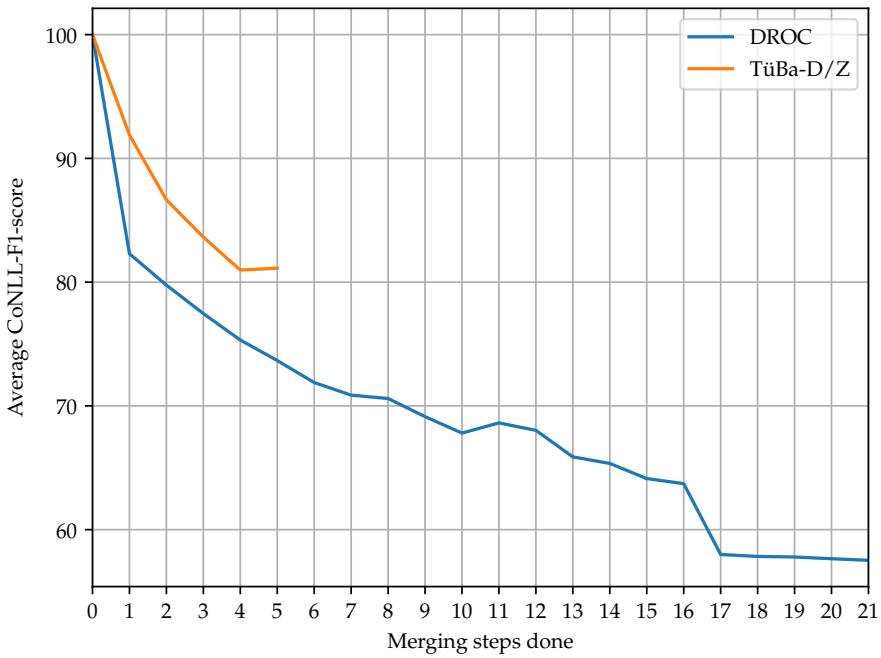


Figure 5.3: Average CoNLL-F1-score of all documents, for the string-based merging approach on both datasets for each merging step

both datasets. This result is surprising, as fastText has the capability to generate embeddings for out-of-vocabulary words, which word2vec does not. However, this could be due to the fact that most words, which were not part of the training corpus, are proper nouns, and the way fastText generates embeddings for them may lead to representation vectors that negatively affect the calculated mean of the mentions.

The neural merging approach, which is the most complex and advanced method using the adapted incremental model, underperforms on both the DROC and news datasets. Even much simpler word embedding approaches that calculate only the cosine similarity and do not train any weights perform better, with an F1-score of only 66.95 on the DROC data for the neural approach.

Upon examining the performance of all proposed merging approaches when using the coarse-to-fine model to predict entities, it becomes apparent that all approaches perform significantly worse compared to the baseline models by Schröder et al. (2021). The best performing approach, which is the word embedding based merging using the word2vec model, is 7.99 F1 worse for the DROC dataset and even 8.49 worse for the news dataset. The high F1-score of 70.09 for the overlapping method on the news data is probably a result of the few entities, that are in this data. Most entities are singletons there or appear only in a short section of the text, so that no merging is needed at all. As already observed for the gold data, all approaches perform better on the news dataset than on the DROC dataset. This is also the case when using the coarse-to-fine model.

Dataset	Approach	Precision	Recall	CoNLL-F1-Score
DROC	string-based	71.32	68.94	65.61
	overlapping	82.28	77.40	79.11
	word embeddings (fastText)	78.78	63.89	70.06
	word embeddings (word2vec)	80.70	66.73	72.64
	neural	75.23	62.46	66.95
TüBa-D/Z 10.0	string-based	90.27	87.95	88.93
	overlapping	94.90	89.82	92.22
	word embeddings (fastText)	93.59	88.42	90.89
	word embeddings (word2vec)	95.38	90.37	92.74
	neural	92.46	88.35	90.36

Table 5.2: Precision, recall and CoNLL-F1-score for all approaches on the TüBa-D/Z 10.0 and DROC dataset. Using gold data and maximum split length of 512 tokens.

The loss of power of the overlapping approach, which has already been mentioned earlier, can also be seen in Figure 5.2, when taking a look at the score for the DROC data. The errors that are introduced during inference of the coarse-to-fine model make the method the worst approach. This problem can also bee seen by looking at the recall and precision values. The overlapping approach has a low recall value of only 44.95 for the DROC dataset, compared to a recall value of 57.32 for the best performing method, which is the string-based merging. While most of the entities are not merged at all, those that are merged are likely to be correct, as evidenced by the high precision value of 59.94, which is the second highest precision value of all methods, topped only by the neural approach with 60.56.

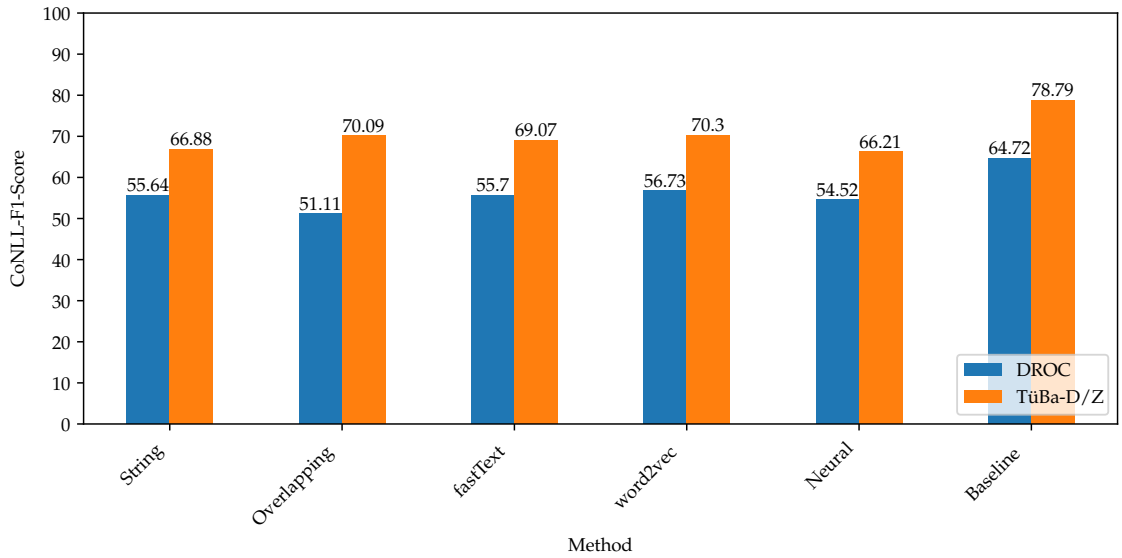
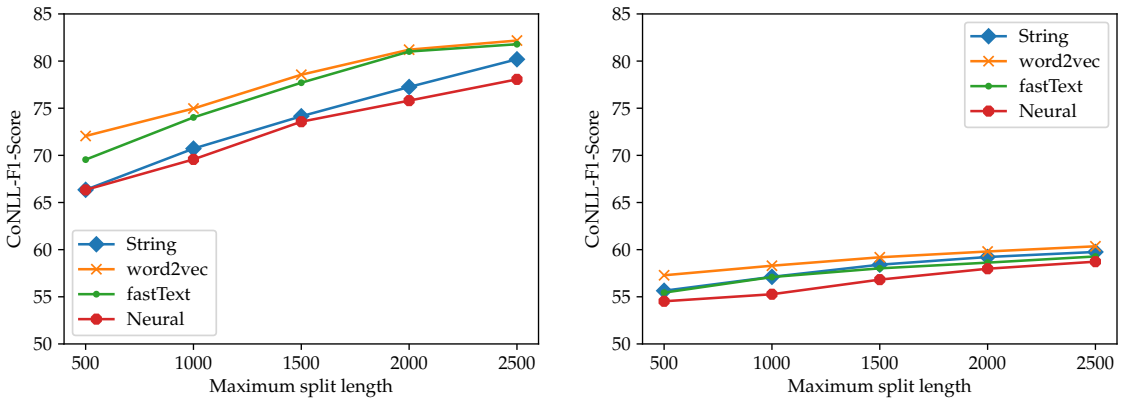


Figure 5.4: Performance of all approaches on the TüBa-D/Z 10.0 and DROC dataset. Using coarse-to-fine model to predict splits and maximum split length of 512 tokens.

5.3.2 Length of Splits

In the previous experiment, we utilized a relatively short split length of only 512 tokens. The reason behind this was that we aimed to maintain the same configuration for both datasets. However, we also wanted to ensure that we had at least two merging steps. Though this approach served our purpose, it did not reflect the real-world scenario. The coarse-to-fine model can efficiently process slightly longer documents with an acceptable amount of memory. In this experiment we will evaluate how the merging approaches perform when maximum length of the splits is increased.

To achieve this, we use only the DROC dataset because the documents in the news dataset are too short. The evaluation will begin by taking a maximum length of 500 tokens and gradually increasing up to 2,500 maximum tokens. With this upper limit we will still perform at least one merging step, even on the shortest DROC document. However, for this experiment, we will not consider the overlapping approach as we investigate it more deeply in the following experiment. For the embedding-based approaches we are using a similarity threshold of 0.8. For the neural approach, we create the representation of the to-be-merged entities again by taking the mean of the mention embeddings instead of using the weighted update-gate.



(a) using gold data

(b) splits predicted with corase-to-fine model

Figure 5.5: CoNLL-F1-score depending on maximum split length using the DROC dataset

Figure 5.5 demonstrats that increasing split length improves the performance of all methods, whether evaluated on gold labeled data or with coarse-to-fine predicted splits. The findings from the initial overview experiment are also evident in both graphs. Gold-labeled data consistently outperforms the coarse-to-fine model predicted evaluation. The performance of all merging methods on the coarse-to-fine model predicted splits never matches the baseline. Among the methods tested, embedding methods exhibit the best performance, while neural and string-based methods perform similarly well.

When evaluating on real predicted data, the word2vec-based embedding method exhibits an improvement in F1 score, increasing from 57.28 F1 for a split-length of 500 to-

kens to 60.36 F1 for a maximum split length of 2,500 tokens. This is close to the coarse-to-fine model baseline CoNLL-F1-score of 61.66. The reasons for the observed improvement in performance are three-fold. Firstly, the coarse-to-fine model has more context to predict the splits, resulting in better entities in the splits themselves. Secondly, longer splits yield fewer opportunities to introduce errors, as fewer merging steps are required. Finally, merging methods have longer context to decide on the merges. These last two factors also account for the observed improvement in performance on gold data.

In summary, the experiment results suggest that all merging methods perform better with longer splits. These findings should be considered when determining the optimal split length, while also accounting for the fact that the performance of the coarse-to-fine model decreases when the document length is too long, as indicated in section 5.1 on the baseline.

5.3.3 Threshold for Word Embeddings

As discussed in the introduction of the embedding method, two entities are merged only when the cosine similarity between them exceeds a given threshold. The threshold determines the similarity between the mentions of both entities required to consider them related to the same global entity. In this experiment, we will evaluate the impact of different threshold values on the performance of both embedding methods (fastText and word2vec). The evaluation will be conducted on both datasets, and to ensure that we can evaluate most of the documents from the news dataset, a relatively short maximum split length of 500 tokens will be used. This evaluation will help us determine the optimal threshold value that maximizes the performance of both embedding methods. We will only evaluate using the gold data to ensure that the results are independent of any errors introduced by the coarse-to-fine model.

As seen in Figure 5.6, we observe that the highest CoNLL-F1 score was achieved for word2vec on both datasets with a threshold value of 0.9. Similarly, fastText achieved the highest score on DROC for a threshold of 0.9. However, for the news dataset, fastText achieved the highest score for a threshold of 0.95. The performance then decreased as the threshold value decreased until a threshold of 0.65. The performance changes only slightly from there on.

We attribute the high performance at high threshold values to the higher probability of two entities belonging to the same global entity when the similarity of their mentions is higher. However, as the similarity decreases, this probability drops. Setting the threshold too high at 0.95 in most cases leads to no merging at all, as the mentions of the two entities would need to be nearly the same.

In conclusion, our findings suggest that a threshold value of 0.9 is optimal for both word2vec and fastText. Although fastText performs slightly better with a threshold of 0.95 for the news dataset, the other three cases show significant improvement with a threshold of 0.9.

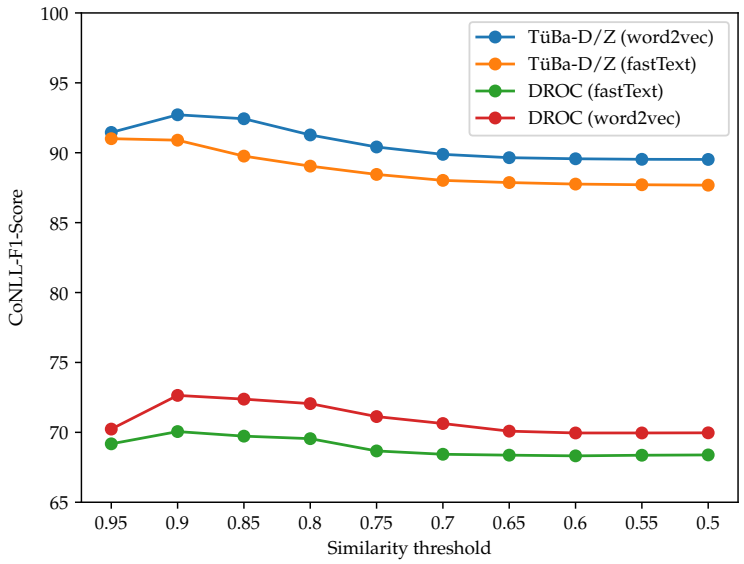


Figure 5.6: Performance of the word embedding approaches on different similarity thresholds. Using gold data and maximum split length of 500 tokens

5.3.4 Size of Overlap

As mentioned in the previous chapter, this experiment aims to investigate the optimal overlapping size further. We hypothesize that with a larger overlap, the performance should increase because more entities are potentially included in the overlap.

For this evaluation, we will only use with the coarse-to-fine model predicted splits since evaluating the overlapping approach on gold data is not meaningful. This is because all overlapping mentions are part of the correct entity, and the evaluation would not reflect real-world performance.

We will evaluate using a fixed split length of a maximum of 1,500 tokens and increase the overlapping length from one sentence up to 11 overlapping sentences. The maximum overlapping size for this dataset with this maximum length is 11 sentences since every split must contain at least one new sentence. This condition is only met up to the maximum overlapping size of 11 sentences. We only evaluate on the DROC dataset due to the required maximum tokens per split length.

As seen in Figure 5.7 surprisingly, the performance already declines when the overlapping size exceeds three sentences. The maximum F1 score was reached at three overlapping sentences, with a CoNLL-F1-score of 57.17. Even at the optimal overlapping size, our approach still falls behind the word2vec based embedding approach, with a difference of 2.01 in F1 score. Word2vec achieved a CoNLL-F1-score of 59.19 for the same maximum split length of 1,500 tokens. We achieved higher results for the overlapping approach than those presented in the overview in subsection 5.3.1, because we used a larger maximum split length in this experiment. As demonstrated in the last experiment, larger split lengths lead to better performance.

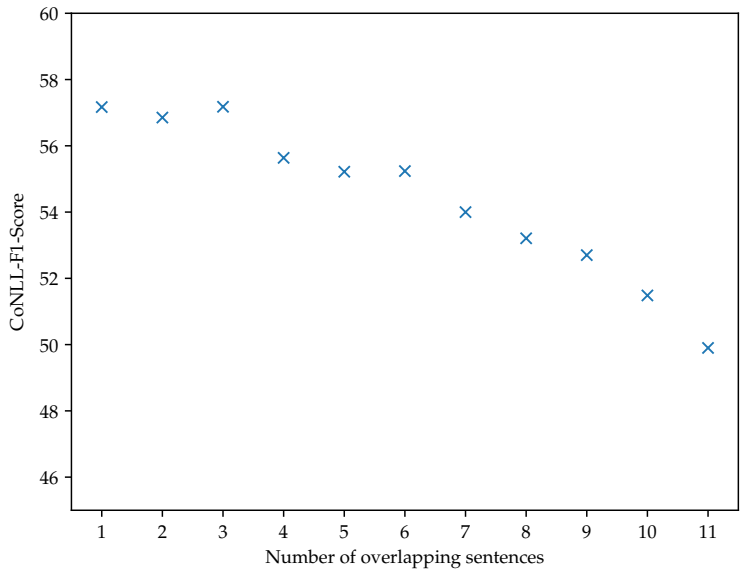


Figure 5.7: CoNLL-F1-Score depending on the number of overlapping sentences. Using coarse-to-fine model and an maximum split length of 1500 tokens

One reason for the decline in performance with larger overlapping windows is that more mentions overlap, but they are not part of the correct entity in the split itself. This results in incorrect merges in the global entity, leading to a lower F1 score. In addition to the poor performance of the overlapping method, another disadvantage is its large computational expense. Due to the numerous overlaps, much of the text needs to be processed twice. When overlapping with only one sentence, the coarse-to-fine model only needs to predict 73 sub-documents in total. However, this value increases to 294 documents when 11 sentences overlap.

5.3.5 Excluding Pronouns in String-Based Merging

All previous experiments have utilized the string-matching approach, as described in subsection 4.3.1, in its default configuration. However, this approach can lead to problems because, in the default configuration, all strings are considered when finding the merging token. In some cases, a pronoun may appear most often in two clusters and not in any other entity of the split, resulting in the clusters being merged based on this token. The issue here is that pronouns without any context only provide limited information about an entity, while other nouns may appear less frequently but provide much more information.

For instance, the appearance of a name like "Linovsky" in two clusters almost certainly indicates that it is the same global entity. Therefore, it is crucial to evaluate how the string matching approach performs when pronouns are not considered in the process of deciding the merge. To accomplish this, a list of all German pronouns will be used to check each mention against it before counting the appearances.

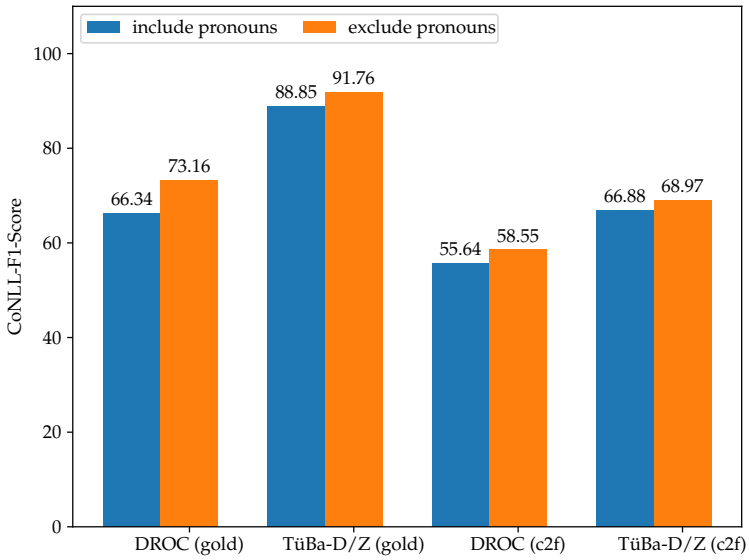


Figure 5.8: CoNLL-F1-Score for both datasets when exlcuding pronouns from merge decision. Using maximum split length of 500 tokens

The results presented in Figure 5.8 indicate that, for both datasets, the CoNLL-F1-score improves when the string-matching approach is used without considering pronouns. This holds true for using gold data and using the coarse-to-fine model to predict splits. For the DROC dataset, the string-based approach without considering pronouns achieved the best working approach, with a CoNLL-F1-score of 55.64. Meanwhile, for the news data, it was only outperformed by the overlapping approach, with a score of 68.97. However, the baseline scores were not reached for both datasets.

In conclusion, ignoring pronouns for the merging process can improve the CoNLL-F1-score, but it is not sufficient to surpass the baseline.

6 Conclusion

In the previous chapter, we conducted an evaluation of various settings for all approaches and thoroughly discussed the outcomes. Now, in this final chapter, we aim to provide an answer to the research question presented in this thesis and reiterate the key findings we have uncovered. Additionally, we will conclude our work by offering a list of suggestions for potential future endeavors in this field.

6.1 Key Findings of This Work

Our evaluation clearly showes that none of the proposed approaches could match the performance of the coarse-to-fine model. Therefore, we can conclude that the split-and-merge strategy, which involves using the coarse-to-fine model to split and predict splits before merging them together, does not improve the performance of existing coreference resolution systems. This finding leads us to reject the hypothesis presented in this thesis, and answers our research question.

In summary, our thesis has yielded the following several key findings.

All approaches perform worse than the baseline

As we have demonstrated through our overview and all other experiments, none of our proposed merging methods could match the CoNLL-F1-Score of the baseline established by Schröder et al. using the coarse-to-fine and incremental model. Even more significantly, our approaches performed worse than the unchanged incremental model, which already employs a constant amount of memory. Therefore, the split-and-merge approach does not provide any benefits over the existing model.

Longer splits lead to better results

All proposed approaches yield better results when the document is split into longer subdocuments. The coarse-to-fine model benefits from this approach by having more context to accurately predict mentions and corresponding entities in the splits itself. Additionally, fewer merging steps need to be performed while also having more context to decide the optimal split.

String-based merging without considering pronouns works best

While the string method does not perform as well as other approaches when considering all words, our evaluation has shown that it outperforms all other approaches when pronouns are ignored. However with a CoNLL-F1-Score of 58.55 for the DROC dataset and 68.97 for the TüBa-D/Z 10.0, the string method falls short of the baseline's performance

Overlapping approach performs worst on real data

As shown in the overview and fourth experiment, the overlapping approach performs worse than all other proposed approaches when evaluated with the coarse-to-fine model predicted splits. Increasing the overlap does not improve performance either. Additionally, this algorithm requires more computational power than the other approaches because the text in the overlap needs to be processed twice by the neural model.

In theory, the overlapping-based merging method we utilized could potentially achieve a near-perfect score, as we observed when evaluating on the gold data. However, this is only possible in the unrealistic scenario where the coarse-to-fine model predicts perfectly. Thus, while this approach may seem promising in theory, it is not practically feasible.

word2vec performs better than fastText

In the embedding-based approach, using pre-trained word embeddings from word2vec leads to better results than using pre-trained embeddings from fastText. This is despite the fact that some words are not part of the training dataset and therefore do not have any vector representation when using word2vec. Although fastText can generate embeddings for these words, its overall performance in evaluation is worse.

In conclusion, the results revealed limitations of all approaches, since no approach matched the baseline performance. Furthermore, the less complex rule-based approaches outperform the more complex neural approach.

6.2 Suggestions for Future Work

Our results showed that the neural model did not perform optimal. Hower it offers the highest potential for improvement, since the modification of the other approaches is limited. In future work, two main issues would be interesting to explore on this merging approach. Firstly, the calculation of the representations of the to-be-merged entities could be improved. Currently, only the mean or the already trained update-gate is used to calculate a representation vector from the embeddings. Fine-tuning the weights of the update-gate could further improve performance. Secondly, the weights for the scoring were not changed in our work. It would be interesting to see how the incremental model performs when these weights are fine-tuned in a further training step.

Additionally, we suggest evaluating all proposed methods on a dataset with longer documents to see if the merging approach performs better at even longer document lengths. Currently, all documents could be processed with only the coarse-to-fine model. If this model can no longer be used, it would be interesting to see how the merging approaches compare to the incremental model. However, there is currently no dataset of long documents with manually annotated mentions and corresponding entities.

Bibliography

C. C. Aggarwal. *An Introduction to Neural Networks*. Springer International Publishing, 2018. ISBN 978-3-319-94463-0. doi: 10.1007/978-3-319-94463-0_1. URL https://doi.org/10.1007/978-3-319-94463-0_1.

F. Almeida and G. Xexéo. Word embeddings: A survey. *CoRR*, abs/1901.09069, 2019. URL http://arxiv.org/abs/1901.09069.

C. Aone and S. Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. In H. Uszkoreit, editor, *33rd Annual Meeting of the Association for Computational Linguistics, 26-30 June 1995, MIT, Cambridge, Massachusetts, USA, Proceedings*, pages 122–129. Morgan Kaufmann Publishers / ACL, 1995. doi: 10.3115/981658.981675. URL https://aclanthology.org/P95-1017/.

S. Azzam, K. Humphreys, and R. J. Gaizauskas. Using coreference chains for text summarization. In *Coreference and Its Applications@ACL 1999, College Park, Maryland, USA, June 22, 1999*. Association for Computational Linguistics, 1999. URL https://aclanthology.org/W99-0211/.

A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566, 1998.

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 238–247. The Association for Computer Linguistics, 2014. doi: 10.3115/v1/p14-1023. URL https://doi.org/10.3115/v1/p14-1023.

S. Bhattacharjee, R. Haque, G. M. de Buy Wenniger, and A. Way. Investigating query expansion and coreference resolution in question answering on BERT. In E. Métais, F. Meziane, H. Horacek, and P. Cimiano, editors, *Natural Language Processing and Information Systems - 25th International Conference on Applications of Natural Language to Information Systems, NLDB 2020, Saarbrücken, Germany, June 24-26, 2020, Proceedings*, volume 12089 of *Lecture Notes in Computer Science*, pages 47–59. Springer, 2020. doi: 10.1007/978-3-030-51310-8_5. URL https://doi.org/10.1007/978-3-030-51310-8_5.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011. doi: 10.5555/1953048.2078186. URL https://dl.acm.org/doi/10.5555/1953048.2078186.

D. Crystal. *A dictionary of linguistics and phonetics*. Blackwell Publishing, 6 edition, 2008. ISBN 9781405152969. doi: 10.1002/9781444302776.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

Fortune Business Insights. Natural language process worldwide market value from 2020-2028 (in billion u.s. dollars), June 2022. URL https://www.statista.com/statistics/1316991/nlp-market-growth/. [Online; accessed 09.05.2023].

T. Glasmachers. Limits of end-to-end learning. In M. Zhang and Y. Noh, editors, *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017, Seoul, Korea, November 15-17, 2017*, volume 77 of *Proceedings of Machine Learning Research*, pages 17–32. PMLR, 2017. URL http://proceedings.mlr.press/v77/glasmachers17a.html.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages. In N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language*

Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA), 2018. URL http://www.lrec-conf.org/proceedings/lrec2018/summaries/627.html.

Z. S. Harris. *Distributional Structure*, pages 3–22. Springer Netherlands, Dordrecht, 1981. ISBN 978-94-009-8467-7. doi: 10.1007/978-94-009-8467-7_1. URL https://doi.org/10.1007/978-94-009-8467-7_1.

J. R. Hobbs. Resolving pronoun references. *Lingua*, 44:311–338, 1978. ISSN 0024-3841. doi: https://doi.org/10.1016/0024-3841(78)90006-2. URL https://www.sciencedirect.com/science/article/pii/0024384178900062.

M. Joshi, O. Levy, L. Zettlemoyer, and D. S. Weld. BERT for coreference resolution: Baselines and analysis. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5802–5807. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1588. URL https://doi.org/10.18653/v1/D19-1588.

V. Kotu and B. Deshpande. *Data science : concepts and practice*. Morgan Kaufmann, 2 edition, 12 2019. ISBN 978-0128147610.

M. H. Krishna, K. Rahamathulla, and A. Akbar. A feature based approach for sentiment analysis using svm and coreference resolution. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 397–399, 2017. doi: 10.1109/ICICCT.2017.7975227. URL https://ieeexplore.ieee.org/document/7975227.

M. Krug, L. Weimer, I. Reger, L. Macharowsky, S. Feldhaus, F. Puppe, and F. Jannidis. Description of a corpus of character references in german novels-droc [deutsches roman corpus]. *DARIAH-DE Working Papers*, 27:1–16, 2018.

S. Kübler and D. Zhekova. Singletons and coreference resolution evaluation. In G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, editors, *Recent Advances in Natural Language Processing, RANLP 2011, 12-14 September, 2011, Hissar, Bulgaria*, pages 261–267. RANLP 2011 Organising Committee, 2011. URL https://aclanthology.org/R11-1036/.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

K. Lee, L. He, M. Lewis, and L. Zettlemoyer. End-to-end neural coreference resolution. In M. Palmer, R. Hwa, and S. Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September*

9-11, 2017, pages 188–197. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1018. URL https://doi.org/10.18653/v1/d17-1018.

K. Lee, L. He, and L. Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 687–692. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-2108. URL https://doi.org/10.18653/v1/n18-2108.

X. Luo. On coreference resolution performance metrics. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 25–32. The Association for Computational Linguistics, 2005. URL https://aclanthology.org/H05-1004/.

Y. Markovski. How chatgpt and our language models are developed, 2023. URL https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed. [Online; accessed 09.05.2023].

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5:115–133, 12 1943. ISSN 0007-4985. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013b. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

N. S. Moosavi and M. Strube. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1060. URL https://doi.org/10.18653/v1/p16-1060.

J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. ISSN 03684245. URL http://www.jstor.org/stable/2098689.

V. Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1142.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In M. A. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1202. URL https://doi.org/10.18653/v1/n18-1202.

S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In S. Pradhan, A. Moschitti, and N. Xue, editors, *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning - Proceedings of the Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, EMNLP-CoNLL 2012, July 13, 2012, Jeju Island, Korea*, pages 1–40. ACL, 2012. URL https://aclanthology.org/W12-4501/.

M. Recasens, L. Màrquez, E. Sapena, M. A. Martí, M. Taulé, V. Hoste, M. Poesio, and Y. Versley. Semeval-2010 task 1: Coreference resolution in multiple languages. In K. Erk and C. Strapparava, editors, *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 1–8. The Association for Computer Linguistics, 2010. URL https://aclanthology.org/S10-1001/.

F. Roza. End-to-end learning, the (almost) every purpose ml method, 2019. URL https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4. [Online; accessed 09.05.2023].

M. Sazli. A brief review of feed-forward neural networks. *Communications Faculty Of Science University of Ankara*, 50:11–17, 01 2006. doi: 10.1501/commua1-2_0000000026.

F. Schröder, H. O. Hatzel, and C. Biemann. Neural end-to-end coreference resolution for German in different domains. In *Proceedings of the 17th Conference on Natural Language Processing (KONVENS 2021)*, pages 170–181, Düsseldorf, Germany, 6–9

Sept. 2021. KONVENS 2021 Organizers. URL https://aclanthology.org/2021.konvens-1.15.

A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4): 35–43, 2001. URL http://sites.computer.org/debull/A01DEC-CD.pdf.

D. Stojanovski and A. Fraser. Coreference and coherence in neural machine translation: A study using oracle experiments. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 49–60, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6306. URL https://aclanthology.org/W18-6306.

R. Sukthanker, S. Poria, E. Cambria, and R. Thirunavukarasu. Anaphora and coreference resolution: A review. *Inf. Fusion*, 59:139–162, 2020. doi: 10.1016/j.inffus.2020.01.010. URL https://doi.org/10.1016/j.inffus.2020.01.010.

H. Telljohann, E. W. Hinrichs, S. Kübler, H. Zinsmeister, and K. Beck. Stylebook for the tübingen treebank of written german (tüba-d/z). *Seminar für Sprachwissenschaft, Universität Tübingen, Germany*, 2017. https://www.sfs.uni-tuebingen.de/resources/tuebadz-stylebook-1707.pdf.

S. Toshniwal, S. Wiseman, A. Ettinger, K. Livescu, and K. Gimpel. Learning to ignore: Long document coreference with bounded memory neural networks. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8519–8526. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.685. URL https://doi.org/10.18653/v1/2020.emnlp-main.685.

J. P. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In J. Hajic, S. Carberry, and S. Clark, editors, *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394. The Association for Computer Linguistics, 2010. URL https://aclanthology.org/P10-1040/.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

M. B. Vilain, J. D. Burger, J. S. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message*

Understanding, MUC 1995, Columbia, Maryland, USA, November 6-8, 1995, pages 45–52. ACL, 1995. doi: 10.3115/1072399.1072405. URL https://doi.org/10.3115/1072399.1072405.

P. Xia, J. Sedoc, and B. V. Durme. Incremental neural coreference resolution in constant memory. In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 8617–8624. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.695. URL https://doi.org/10.18653/v1/2020.emnlp-main.695.

Z.-H. Zhou. *Introduction*. Springer Singapore, 2021. ISBN 978-981-15-1967-3. doi: 10.1007/978-981-15-1967-3_1. URL https://doi.org/10.1007/978-981-15-1967-3_1.

List of Figures

List of Tables

List of Listings

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang 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 16.05.2023 Unterschrift: 

Veröffentlichung in der Bibliothek

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

Hamburg, den 16.05.2023 Unterschrift: