BERGISCHE
UNIVERSITÄT
WUPPERTAL

School of Electrical,
Information and Media Engineering

Chair for Technologies and Management of Digital Transformation

Master-Thesis

Coreference Resolution to Improve Data Quality in the Logs of Daily Operations: Restoring Missing Links Between Problem and Solution Records

Thomas Ebner
1526180

Electrical Engineering

Wuppertal, 19. October 2022

Supervisor Anastasia Zhukova M.Sc.

First reviewer Prof. Dr.-Ing. Tobias Meisen
Second reviewer Prof. Dr.-Ing. Bela Gipp

BERGISCHE UNIVERSITÄT WUPPERTAL

Fakultät für Elektrotechnik, Informationstechnik und Medientechnik

Campus Freudenberg
Rainer-Gruenter-Straße
42119 Wuppertal

Lehrstuhl für
Technologien und Management der Digitalen Transformation

Univ.-Prof. Dr.-Ing. Tobias Meisen

Thema für die Master-Thesis

Kandidat*in: Ebner, Thomas

Matrikel-Nr.: 1526180

Studiengang: Elektrotechnik

Fachliche*r Ansprechpartner*in: Anastasia Zhukova M. Sc.

Thema: Coreference resolution to improve data quality in the logs of daily operations: Restoring missing links between problem and solution records

Aufgabenstellung:
A common dilemma in industry is finding a solution to a problem that has already been solved in the past. When problem solutions are not efficiently tracked, it makes it difficult to solve recurring problems. One way to address this problem in the context of production is for plant operators to manually create log files in which they report on events in production. These log files contain problem and solution descriptions as well as other information. In this thesis, the underlying assumption will be validated, if such log files can be used to suggest solutions to emerging problems with the help of historical data. This requires related problem and solution descriptions to be automatically extracted and linked. The task of linking solutions to problems can be viewed as a coreference resolution task, where a problem is viewed as the antecedent of a solution.
In industrial setups some problems occur that increase the complexity of the coreference resolution task - such as the procedures for logging the different types of events are not precisely defined, or problems can be created as an information description and later extended by a solution description. In this work, the research objective is, to give an answered to the research question: "How can coreference resolution be used for linking solutions to problems?". The result of this thesis is a pipeline that uses coreference resolution as a tool that can be used either as a preprocessing step to link problems and solutions retrospectively, or as a standalone application that searches for not linked records from the last 8-24 hours at the end of a shift and links them.

Wuppertal, den 19. April 2022

Tobias Meisen

Unterschrift Erstgutachter*in

Erstgutachter*in: Prof. Dr.-Ing. Tobias Meisen
Zweitgutachter*in: Prof. Dr. Bela Gipp

Prüfungsamt

Kennziffer: 22ITHMET & Mei
Ausgabedatum: 19.4.2022
Abgabedatum: 19.10.2022

Unterschrift Prüfungsamt

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Wuppertal, den 19. Oktober 2022 _____

(Unterschrift)

Einverständniserklärung

Ich bin damit einverstanden, dass meine Thesis wissenschaftlich interessierten Personen oder Institutionen und im Rahmen von externen Qualitätssicherungsmaßnahmen des Studienganges zur Einsichtnahme zur Verfügung gestellt werden kann. Korrektur- oder Bewertungshinweise in meiner Arbeit dürfen nicht zitiert werden.

Wuppertal, den 19. Oktober 2022 _____

(Unterschrift)

Kurzfassung

Diese Arbeit leistet einen Beitrag zu einem umfangreicheren Projekt des Solution Recommender Systems, d.h. ein System schlägt bei einer Problembeschreibung Lösungen vor, die auf zuvor gelösten ähnlichen Problemen basieren. Das Erlernen korrekter Lösungsvorschläge setzt voraus, dass die Datenqualität der Datenbasis hoch genug ist, damit ein solches Modell die richtigen Lösungsvorschläge erlernen kann. Als Datenbasis werden die Logdateien aus dem operativen Betrieb in der verarbeitenden Industrie in deutscher Sprache verwendet. In dieser Arbeit wird die Koreferenzauflösung auf Satz- und Absatzebene eingesetzt, um die Datenqualität solcher Logdateien zu verbessern, d.h. fehlende Verknüpfungen zwischen Problemen und Lösungen wiederherzustellen.

Zwei verschiedene State-of-the-Art-Modelle zur Koreferenzauflösung werden evaluiert, ausgewählt und angepasst: 1) das Feature-basierte neuronale Netzwerk von Clark und Manning ([13, 14]) und 2) das Transformer-basierte neuronale Modell für CDCR von Caciularu et al. [7].

Die Ergebnisse zeigen, dass die Definition der Koreferenzauflösung auf das aktuelle Problem anwendbar ist. Das leistungsstärkste Modell ist das implementierte Transformer-basierte PR-CDLM (problem-solution cross-document language model) mit domänenspezifischen Merkmalen (CoNLL $F_1 = 76.7$). Die Kombination der Transformer-Architektur mit zusätzlichen Merkmalen zeigt einen signifikanten Vorteil gegenüber reinen Transformer-basierten Modellen (CoNLL $F_1 = 68.6$) und gegenüber reinen Feature-basierten Modellen (CoNLL $F_1 = 55.8$).

Abstract

This thesis contributes to a larger project of the solution recommender system, i.e., given a problem description, a system suggests solutions based on previously solved similar problems. To learn correct solution suggestions presupposes that the data quality of the database is high enough for such a model to learn the correct solution suggestions. As a database, the log files from daily operations in the processing industry in the German language are used. This thesis uses coreference resolution on the level of sentences and paragraphs to improve the data quality of such log files, i.e., to restore missing links between problems and solutions.

Two diverse state-of-the-art models for coreference resolution are evaluated, selected, and adjusted: 1) the feature-driven neural network by Clark and Manning ([13, 14]) and 2) the transformer-based neural model for CDCR by Caciularu et al. [7].

The results prove that the coreference resolution definition is applicable to the current domain problem. The best-performing model is the implemented transformer-based PR-CDLM (problem-solution cross-document language model) with domain-specific additional features (CoNLL $F_1 = 76.7$). The combination of the transformer architecture with additional features demonstrates a significant advantage over pure transformer-based models (CoNLL $F_1 = 68.6$), and over pure feature-based models (CoNLL $F_1 = 55.8$).

Contents

List of Figures

List of Tables

1 Introduction

This master thesis addresses how coreference resolution (CR) can be applied at the level of sentences and paragraphs as a tool for linking problems and solutions of log files created in an industrial environment in the German language. The introductory chapter begins with the problem definition and the motivation derived from it (Section 1.1), presents the proposed research objective pursued in this thesis (Section 1.2), and outlines the remaining chapters (Section 1.3).

1.1 Problem Setting and Motivation

A common dilemma in the workplace is how to find a solution to a problem that has already been solved in the past. Knowledge management (KM) has generated considerable interest in business and management circles due to its capability to deliver strategic results relating to profitability, competitiveness, and capacity enhancement [10]. Knowledge management is promoted as an essential and necessary factor for organizational survival and maintenance of competitive strength [38]. Successful organizations now understand why they must manage knowledge, develop plans to accomplish the objective, and devote time and energy to these efforts. KM has been described as a critical driver of organizational performance [6] and one of the essential resources for the survival and prosperity of organizations [51]. In the age of rapid technological innovation and change, KM is a crucial enabler for value creation.

One way to address knowledge management in an industrial setting is for plant operators to manually create log files in which they report on events in production, such as the machines used in production. Ideally, related problem and solution events are entered into text fields for this purpose. This would build a database that can be used to refer back to an already found solution in case of recurring problems. Unfortunately, the procedures for logging the different types of events are typically not precisely defined, which leads to an inconsistent structure of the reported events. For example, problem and solution events can be entered into one text field, i.e., a distinction between problem and solution is no longer possible. Another issue could be that problem and solution events are entered separately but are not linked explicitly. In this case, there is no linked solution to the problem, although a solution exists in the database. It is also possible to enter problems and solutions in the wrong category altogether, i.e., a problem could be assigned to the information category. Moreover, it is common in a production environment that a problem is only solved after a more extended period, e.g., one or two days later.

These issues lead to a decrease in the data quality in the log files, which limits the effective use of the collected knowledge in the industrial setting. Therefore, the log files' data quality must be improved to operate effective KM. Once sufficient data quality is available, the log files can be used as a database to train a model which suggests suitable solutions to emerging problems. Such a model can only be trained efficiently using historical data from the log files to learn how problems and solutions are related in order to propose solutions to problems. For this purpose, a data set must be prepared that contains only explicitly labeled linked and non-linked problems and solutions.

Linking solutions to problems can be viewed as a coreference resolution task, where a problem

is viewed as the antecedent of a solution. Coreference resolution is a well-studied research topic within Natural Language Processing (NLP), where good results have already been obtained in various domains and describes the task of resolving anaphoric expressions into their antecedents. While current state-of-the-art coreference resolution models focus on mentions on word level or spans of a couple of words [7], the log files typically contain problem and solution events on the level of sentences and paragraphs. In order to solve the problem with the time-separated problems and solutions within the framework of coreference resolution, it can be defined that a document is created for each shift, i.e., one log file corresponds to one document. With this definition, an approach can be made to link related problems and solutions with the help of cross-document coreference resolution.

1.2 Research Objective

To train a solution suggestion model, the data quality of the log files must be improved, i.e., problem and solutions must be linked explicitly. Motivated by developing a coreference resolution model to improve data quality in the database of an industrial setting, the following research question (RQ) was defined:

How can coreference resolution be used to improve data quality in the logs of daily operations, i.e., restore missing links between problem and solution records?

Therefore, following research tasks (RT) are derived:

1. Create an artificial dataset derived from the logs of daily operations;
2. Preprocess the data to get the data into the necessary input format for the models;
3. Identify state-of-the-art models for within-document and cross-document coreference resolution and develop a problem-solution linking model based on them;
4. Evaluate the developed models.

1.3 Outline

Chapter 1 introduced the task of linking problem and solution datasets (Section 1.1) and presented the proposed research objective for this thesis (Section 1.2).

Chapter 2 addresses the research tasks by providing fundamental background knowledge on coreference resolution (Section 2.1) and an overview of the main classical as well as novel approaches to coreference resolution (Section 2.2) for both within-document coreferences (Section 2.2.1) and cross-document coreferences (Section 2.2.2).

Chapter 3 defines the task in detail and presents the scope and limitations (Section 3.1) of this thesis. Then it describes the conducted preprocessing of the log files (Section 3.2). Next, the chapter explains the NeuralCoref model (Section 3.3) and the CDLM model (Section 3.4) in detail, starting with the specific preprocessing steps, continues with the explanation of the model

architecture of the original models and describes how the models are implemented for this thesis. Finally, the chapter concludes with a summary of the critical points (Section 3.5).

Chapter 4 presents the evaluation metrics used to evaluate a classification model as well as coreference resolution (Section 4.1), then shows the results and limitations of the trained NeuralCoref model (Section 4.2) and the CDLM model (Section 4.3), and closes with a summary (Section 4.5).

Chapter 5 describes which improvements for the NeuralCoref model (Section 5.1) and for the CDLM model (Section 5.2) can be considered in the future and presents an end-to-end prototype that represents a complete pipeline to solve coreferences including mention extraction in the log files (Section 5.3).

Chapter 6 concludes the thesis with a summary.

2 Background and Related Work

The background and related work chapter explains the basics of coreference resolution (Section 2.1), followed by an overview of several classic as well as novel state-of-the-art approaches to solve coreference resolution (Section 2.2), further categorized into within-document coreference resolution (Subsection 2.2.1) and cross-document coreference resolution (Subsection 2.2.2). Finally, the chapter concludes with a summary (Section 2.3).

2.1 Coreference Resolution

Knowing who is being talked about in a text is essential to natural language processing (NLP) [24]. Consider the following passage [30]:

> Barack Obama nominated Hillary Clinton as his secretary of state on Monday. He chose her because she had foreign affairs experience as a former First Lady.

Figure 2.1 Basic coreference resolution example.

The underlined words in this passage are used to refer to the same real-world entity. Linguistic expressions like "Barack Obama" or "his" are called *mentions*. A mention can be a noun phrase (NP), a named entity (NE), or a pronoun, which refers to an entity in the real world known as the referent [48]. Two or more mentions that refer to the same entity are said to *corefer* [24].

Linguistic expressions that refer back to another word or phrase are called *anaphora*. A word or phrase that gets its meaning from a preceding expression is an anaphor (like "his" and "He" in Figure 2.1). The preceding expression is called an *antecedent* ("Barack Obama" in Figure 2.1) [22]. Not every referring expression is an antecedent. An entity with only a single mention in a text is called a *singleton* [24].

Coreference resolution (CR) is the task of determining whether two mentions in a text refer to the same underlying real-world entity [37]. The set of coreferring mentions is called *coreference chain* or a *cluster*. Thus, the task of coreference resolution comprises two tasks: First, identify the mentions, and second, cluster them into coreference chains. Alternatively, expressed formally: "Given a text T, find all entities and the coreference links between them. The task is evaluated by comparing the predicted links with those in human-created gold coreference annotations on T" [24].

Referring to the coreference example in Figure 2.1, now using superscript for each cluster and subscript for individual mentions in the cluster, a CR model would need to extract and cluster the following mentions:

> $[\text{Barack Obama}]^1_a$ nominated $[\text{Hillary Clinton}]^2_a$ as $[\text{his}]^1_b$ $[\text{secratary of state}]^2_b$ on Monday. $[\text{He}]^1_c$ chose $[\text{her}]^2_c$ because $[\text{she}]^2_d$ had foreign affairs experience as a former $[\text{First Lady}]^2_e$.

Figure 2.2 Coreference example using superscript for mark each cluster, and subscript for individual mentions in each cluster.

A CR algorithm needs to find at least two cluster corresponding to the two entities in the text:

1. {Barack Obama, his, He}
2. {Hillary Clinton, secretary of state, her, she, First Lady}

Typically, CR requires a preprocessing pipeline comprising a variety of NLP tasks (e.g., tokenization, lemmatization, named entity recognition, and part-of-speech tagging). These tasks are addressed before training the coreference resolution model. Errors made during preprocessing impact the CR models' performance, assuming that the provided input is correct. Models trained with incorrect data are less accurate [15].

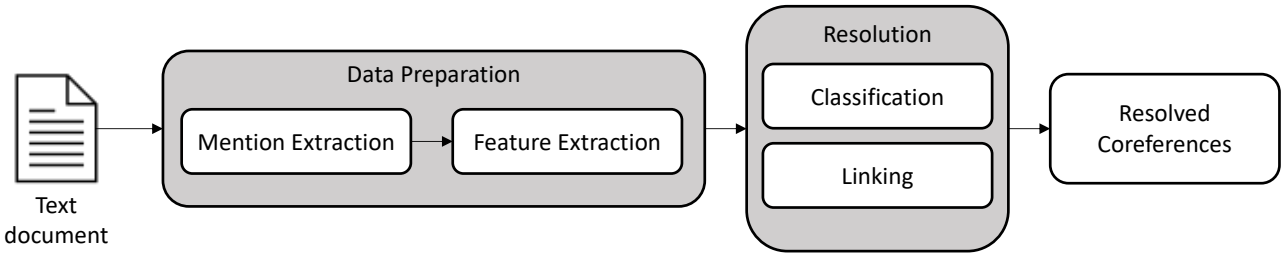


Figure 2.3 Basic architecture of a coreference resolution system. The input consists of raw text documents. In the data preparation stage the mentions are extracted as well as some features of the mentions. In the resolution stage mention-pairs are classified if they are coreferent or not, and linked to coreference chains. The output is a file containing the resolved coreferences [15].

The typical architecture of a CR system features data preparation and resolution phases, as seen in Figure 2.3. For most CR tasks, the input to the system is the raw text of articles. The data preparation pipeline consists of detecting mentions in the input text, followed by a feature extraction step, converting each data instance into an expressive feature vector. Extracted features are dependent on the specific task and could be document features, e.g., the length of the document or the total mention count; mention features, e.g., named entity type, or pronoun; or mention-pair features, e.g., mention head match, or mention distance [44]. The resolution phase consists of classifying these instances as coreferent or not and the linking of mentions into the final coreference chains [15].

Precisely what counts as mention and what links are annotated differs from task to task and dataset to dataset. For example, some coreference datasets do not label singletons. Some tasks use gold mention-detection (i.e., the system is given human-labeled mention boundaries, and the task is to cluster these gold mentions), which eliminates the need to detect and segment mentions from running text [24].

2.2 Approaches for Coreference Resolution

This section outlines the major machine learning approaches to CR. Classic coreference algorithms are distinguished based on whether they make each coreference decision in a way that is *entity-based* (i.e., representing each entity in a text), or only *mention-based* (i.e., considering each mention independently), and whether they use *ranking models* to directly compare potential antecedents [37]. Novel approaches in CR are based on end-to-end models, outperform the previous state-of-the-art models, and will be outlined after the classic approaches. Furthermore, the approaches are divided into CR for within-document (WD) (Subsection 2.2.1) and cross-document (CD) (Subsection 2.2.2) approaches.

2.2.1 Within-Document Coreference Resolution

Most CR research is conducted for within-document tasks. Particularly in the early approaches, only individual documents were considered. This subsection deals with classical approaches to CR that classically consider only single documents.

Soon et al. [49] proposed the *mention-pair* model, one of the simplest and most influential architectures [24]. It combines a binary classifier that, given a pair of mentions, a candidate anaphor, and a candidate antecedent, distinguishes whether it is a coreference or not, with fast heuristic procedures to merge the decision of the classifier in the decoding phase [52]. While earlier classifiers used hand-built features, more recent classifiers use neural representation learning [11]. Despite the advantage of simplicity, the mention-pair model has two significant weaknesses. (1) The classifier is not trained to decide between two likely antecedents, which is better, as it does not directly compare candidate antecedents to each other. (2) Each classifier decision is made entirely locally to the pair, without being able to take into account other mentions of the same entity, as the classifier is looking only at mentions, not entities.

Instead of considering the CR problem as a pairwise binary classification task between all possible antecedents, *mention-ranking* models directly score candidate antecedents to each other for their likelihood of coreference rather than comparing partial coreference clusters, choosing the highest-scoring antecedent for each anaphor [15]. The candidate antecedent may be any mention before a given target mention in the document or NA, indicating that the target mention has no antecedent [24]. This approach limits errors caused by the lack of global contextual information. However, during the classification phase, this approach may still encounter contradictions by considering each target mention and the corresponding anaphora separately. These contradictions must later be solved in the linking process.

Both the mention-pair and mention-ranking models make their decisions about mentions. By contrast, *entity-based* models link each mention not to a previous mention but to a previous *entity* (cluster of mentions). Generally, entity-based classifiers have the same features as mention-pair classifiers. The difference is that an entity feature's value considers the specific values of all mentions in a partial entity. Hence, this type of model can often overcome contradictions in the linking process [15]. A mention-ranking model can be turned into an entity-ranking model simply by having the classifier make its decisions over clusters of mentions instead of the individual mentions [42].

Cluster-ranking models can overcome local linking contradictions using entity-level information. The models take a pair of mention clusters instead of a pair of mentions as input. These partial entities are incrementally built of coreferring mentions. Each cluster starts with a single mention and iteratively builds coreferential clusters [15]. Wiseman et al. [55] used recurrent neural networks (RNNs) to extract entity-level representations of mentions. At each new link, the representation of a mention is reevaluated by running a RNN over the cluster mentions in order. Clark and Manning [14] further improved the performance of cluster-ranking models using a pooling-based cluster-pair encoder. Each mention from cluster *c1* pairs with each mention of cluster *c2*. The pairs are run through a mention-pair encoder, and the resulting matrix of mention-pair representations is max- and average-pooled for a final cluster-pair representation [15].

As with many other NLP tasks, the use of neural networks to learn non-linear models of CR is now a common practice. End-to-end models solve the subtasks of CR together in a single model. The end-to-end objective allows related subtasks to have a common optimization goal, leading to the models achieving better overall results than pipeline systems. The system is enabled to learn from pairwise interactions between tasks. Therefore, the system avoids cascading errors and provides more information for tasks traditionally at the beginning of the pipeline and would not have access to the results of later tasks [15].

Lee at al. [26] proposed an end-to-end neural model that jointly learns which text spans are mentiones and how to best link them. They produced vector embeddings of text spans utilizing a bidirectional long short-term memory (LSTM) network [21], capturing the spans' representation in the context of the whole sentence. The antecedent and mention scores for each span pair are computed from the same learned span representations, allowing for the error signal of both tasks to be used for one common optimization goal. As all spans (up to a maximum number of tokens) are considered potential mentions, this model is computationally intensive. The authors later improved their system by adapting a span-ranking architecture to capture better entity-level information [27]. Also, the system uses an antecedent pruning mechanism for reducing computational complexity, computing less expensive antecedents and cluster scores for longer-distance span links.

Joshi et al. [23] proposed a model based on the model by Lee at al. improved with BERT embeddings [17], both for base and large variants. BERT is a pre-trained transformer network [53], which set for various NLP tasks new state-of-the-art results, including question answering, sentence classification, and sentence-pair regression. They present two variants of their model: (1) the *independent* variant uses non-overlapping segments, each of which acts as an independent instance for BERT, and (2) the *overlap* variant splits the document into overlapping segments to provide the model with context beyond 512 tokens. When facing the 512 limit for context tokens for BERT, using independent BERT instances on non-overlapping segments proved to be the best-performing option, as opposed to using overlapping segments to provide the model with context beyond 512 tokens.

2.2.2 Cross-Document Coreference Resolution

Significant progress has been made in within-document coreference. However, the more significant problem of cross-document (CD) coreference has not received as much attention. This subsection outlines some of the progress made in CDCR in recent years.

Beltagy et al. [4] have proposed a long-range language model called Longformer to overcome the model's shortcomings by Joshi et al. The Longformer extends the capabilities of earlier transformers [53] to process long sequences using a sparse self-attention architecture. The model showed improved performance on both long-document and multi-document tasks. In the case of multiple documents, instead of encoding documents separately, the model allows concatenating the documents into a long sequence of tokens and encoding them jointly.

Barhom et al. [3] developed a model for cross-document coreference resolution (CDCR), as most CR research has focused on its within-document variant, despite the importance of the task. They propose an algorithm that incrementally constructs the final clustering configuration. A vector representation is computed for each mention, containing the mention span, the context surrounding the mention, and some additional features extracted from the given mention and text. The mention representations are fed to a mention pair scorer that predicts whether the mentions belong to the same cluster. Finally, they apply agglomerative clustering where the cluster merging score is based on the predicted pairwise mention scores. This model outperforms the previous state-of-the-art event coreference model on ECB+ while providing this corpus's first entity coreference result. Furthermore, they show that all the representation elements, including the mention span itself, its context, and the relation to other mentions, contribute to the model's success.

Caciularu et al. [7] improved the model developed by Barhom et al. They introduced a new pretraining approach geared for multi-document language modeling, incorporating two key ideas into the masked language modeling self-supervised objective. First, instead of considering documents in isolation, they pretrain over sets of multiple related documents, encouraging the model to learn CD relationships. Second, they improve over current long-range transformers by introducing dynamic global attention with access to the entire input to predict masked tokens. The CDLM (Cross-Document Language Model) is a new general language model for multi-document settings that can be quickly adapted to downstream tasks. They show through extensive analysis that both ideas are essential for the success of CDLM and work in synergy to set new state-of-the-art results for several multi-text tasks.

Cattan et al. [9] highlight the need for a more realistic evaluation of the CDCR task. In particular, existing work (e.g., Barhom et al. [3]) often reports performance using gold-annotated datasets instead of predicted mentions by the model. The contribution of mention prediction significantly impacts the model performance and should be considered during the evaluation. They argue that an end-to-end training approach is more realistic for real-world data.

2.3 Summary

CR is a well-established NLP research topic and describes the task of linking mentions in a text that corefer, i.e., refer to the same entity, resulting in a set of coreference clusters. To resolve coreferences, typically, two individual tasks must be performed: First, identifying the mentions, and second, clustering them into coreference chains. The major learning-based models for CR are mention-based, entity-based, mention-ranking, and cluster-ranking models. Novel coreference systems tend to be end-to-end architectures, as these models outperform the previous state-of-the-art models. The introduction of the transformer-based BERT model, in particular, has fundamentally changed the approaches taken to solve coreferences. While significant progress has been made in within-document coreference, the larger problem of CD coreference has not received as much attention. In recent years, some promising CDCR models have been presented. However, it must be taken into account that the results may be overestimated, as in most cases, they are evaluated with gold mentions instead of predicted mentions.

3 Methodology

While current state-of-the-art coreference resolution models are focused on mentions on word level or spans of a couple of words [7], the goal of this thesis is to develop a coreference resolution model that will be able to deal with larger spans of text, i.e., on the level of sentences and paragraphs. The state-of-the-art models cannot simply be adopted, and similar results expected. This thesis attempts to determine whether CR also works for sentences or paragraphs of log files.

The methodology chapter defines the research task and determines the scope and limitations of this thesis (Section 3.1). It then describes the data preprocessing conducted on the log files (Section 3.2), and then continues to explain the models applied to solve the task in detail (Section 3.3 and Section 3.4) and finally concludes with a summary of the key points (Section 3.5).

3.1 Scope and Limitation

This thesis uses log files from an industrial production environment as a database created in the German language. The log files are created at different machines, sites, and companies. They contain descriptions of problems, solutions, and additional information occurring in production. Problems that occur randomly during production must be entered manually into the log files by user input. Solutions must also be entered manually once a problem has been solved. Information reports can be various events, from instructions, and administrative notes, to quality remarks. All kinds of individual descriptions inside the log files are called *entry items*.

Ideally, the solution to a problem is linked manually to the solved problem when the solution entry item is created. Unfortunately, several circumstances cause solutions not always to be explicitly linked to the related problem. For example, problem and solution events can be entered together in one text field, so it is no longer possible to distinguish between problem and solution. Another issue is that the link between a problem and a solution is not made during user input. It is also possible that a problem or a solution is entered in the wrong category altogether, i.e., a problem could be assigned to the information category. In addition, every problem and every solution is of varying complexity, and there is no uniform standard in which form a description must be provided. Therefore, the descriptions vary in length, from single words to several sentences, i.e., a paragraph. All these issues lead to a decrease in the data quality in the log files.

With the help of these log files, the underlying assumption will be validated if such log files can be used to suggest solutions to emerging problems using historical data. This objective requires an improvement in the data quality of these log files. Data quality increases when related problems and solutions are extracted and then linked. Therefore, linking solutions to problems can be viewed as a CR task, where a problem is defined as an antecedent of a solution (the solution is the anaphora in this example). In such log files occur several combinations of problems and solutions. In the context of the CR task, the following definitions are made (compare Figure 3.1a).

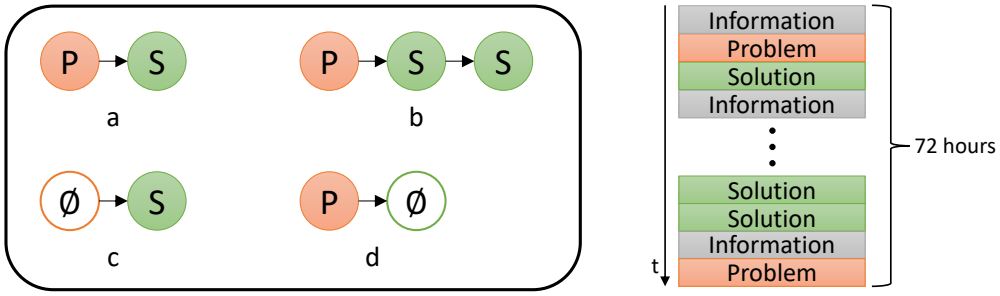
a. The simplest case is when there is only one problem with one linked solution. In this case, it is defined that a problem is an antecedent of a solution, i.e., a coreference chain of size 2.

b. A problem can also have multiple solutions, i.e., anaphora. This case implies that a

solution can have both problems and solutions as antecedents. In the industrial production environment, a problem has been solved after N attempts, i.e., a coreference chain of size N.

c. A solution can also have no antecedent, i.e., the solution has no linked problem. No problem was reported in this case, but there is still a solution in the log files. Defined in the context of a CR task, the solution without a linked problem is equivalent to a singleton.

d. Similar to the previous case, there can also be the reverse scenario, i.e., a problem does not have an anaphora. This case means that a problem has not been solved or the solution has not been reported. In the CR setting, this problem is also a singleton.



a Possible combinations of problems and solutions in a log file.

b Schematic representation of a log file over a period of 72 hours.

Figure 3.1 Linking solutions to problems: a coreference resolution task. Figure 3.1a shows a schematic overview of what kind of coreference chains can be formed with the problems and solutions of a log file. ø means there is no linked problem or solution. Figure 3.1b shows a schematic overview of a log file which are defined as documents for the task.

Figure 3.1b illustrates a schematic representation of a log file over 72 hours. The period of 72 hours is defined as the maximum time difference between problems and solutions that are tried to be linked. Section 3.2 explains why the time limit of 72 hours is chosen.

This thesis aims to develop a pipeline that uses CR as a tool that can be used either as a preprocessing step to link problems and solutions retrospectively or as a standalone application that searches for not linked records in a given time frame and links them.

For this thesis, the problem, solution, and information labels are already given. Problems and solutions are defined as mentions. Information texts are considered as the context of the mentions. As the labels are already given, the first step of a typical CR pipeline, i.e., the mention extraction, is out of the scope of this thesis.

The general idea of the CR pipeline is visualized in Figure 3.2. The input to the pipeline consists of the raw log files. First, some preprocessing is conducted, like formatting and setting the cluster ids of coreferent problems and solutions. No mention extraction has to be conducted as the mentions are already given. Second, in the data preparation stage, the features of the mentions are extracted. Mention features are, for example, the creation time of the mention or where in the document the mention occurs. The feature extraction is either conducted during

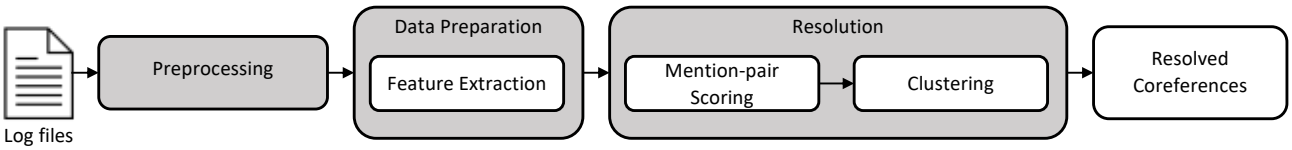


Figure 3.2 Conceptual visualization of the coreference resolution pipeline. The input consists of the raw log files. Fist, some general preprocessing is conducted. Second, as the mentions are already given, no mention extraction has to be done. Hence, only the feature extraction is conducted. Third, in the resolution stage, depending on the score assigned by a mention-pair scoring model the cluster are formed. Finally, the output are the resolved coreferences, i.e., linked solutions and problems.

preprocessing or as the first step of the CR model. Third, in the resolution stage, mention pairs are formed. Mention-pairs are possible combinations of all problems and solutions. Hence, a mention-pair could also consist of two problems or two solutions. The mention-pairs are scored using a mention-pair scoring model. The model's score is the likelihood that two mentions are coreferent. The clustering of the mentions is also part of the resolution stage. Depending on the score of the mention pairs, the mentions are clustered or not. Finally, the pipeline's output is a file with the resolved coreferences, i.e., the linked problems and solutions of the log files.

Two models are developed to solve the research task. First, a classic state-of-the-art algorithm for coreference resolution by Clark and Manning ([13, 14]) is adapted. In particular, the *NeuralCoref* model by *HuggingFace* is adapted [57], as it can be modified with relatively little effort to suit the research task. NeuralCoref is designed to strike a good balance between accuracy and speed/simplicity, using a constrained number of features and a simple feed-forward neural network. The model is explained in detail in section 3.3 and is referred to as the *NeuralCoref model* in the further course of the thesis.

Second, a model based on the transformer-based cross-document language model (CDLM) by Caciularu et al. [7] is implemented. The CDLM is currently state-of-the-art for cross-document coreference resolution and is an improvement of the Event-Entity-CDCR by Barhom et al. [3]. CDLM is trained over sets of multiple documents instead of considering documents in isolation, using a long-range transformer and dynamic global attention that has access to the entire input to predict masked tokens. The model is explained in detail in section 3.4 and referred to as the *CDLM model*.

3.2 Preprocessing

The dataset for this thesis consists of log files generated using the *Shiftconnector*. Shiftconnector is a software developed by *eschbach GmbH* for logging events in an industrial production environment. These log files originate from three different companies and a total of nine different sites. The data was collected between 17/02/2014 and 08/06/2022. All entry items are assigned to three categories problem, solution, and information. The category information includes everything that does not represent a problem or a solution. In total there are 14.411 problem, 22.932 solution, 287.056 information entry items. The related problem and solution pairs comprise a total of 14.396 clusters. A cluster is either a simple problem-solution pair or a combination of

several problems and solutions, i.e., a coreference chain.

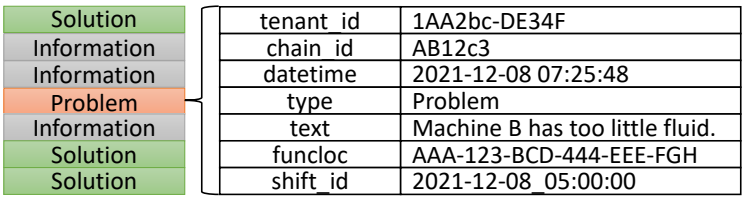


Figure 3.3 Schematic structure of the log file entries. Each entry item has additional meta information included. To create the gold annotated datasets the *type* and the *chain_id* are used.

Figure 3.3 illustrates the available meta data for each entry item (i.e., problem, solution and information) of the log files: tenant_id, chain_id, datetime, type, text, funcloc, and shift_id.

- The **tenant_id** is used to uniquely identify the site where the entry item was made. Problem-solution pairs are only possible when they have the same tenant_id, i.e., recorded at the same site.
- The **chain_id** is the cluster id of the coreference chain. Each problem-solution pair (or cluster) has the same chain_id assigned. The chain_id extracted from the log files is modified from a combination of letters and digits to a numerical value during preprocessing.
- The **datetime** is the exact time the entry item was saved in the log files. This value is later used to determine the time difference between two mentions.
- The **type** is the assignment of the categories problem, solution, or information. These categories would need to be recognized and assigned in the mention extraction phase.
- The **text** contains the user-written text of the individual entry items. These are the specific problem and solution descriptions that would need to be extracted in the mention extraction phase. The content varies in length from single words to several sentences.
- The **funcloc** describes the functional location of the entry item. A functional location is an organizational unit that structures objects of an enterprise according to functional, process-oriented, or spatial aspects. Thus, the functional location is used to describe where, for example, a problem has occurred on a machine.
- The **shift_id** assigns each entry item to a specific shift. All entries with the same shift_id belong to the same shift. A shift is defined as a document in the context of this thesis.

Figure 3.4 shows the essential preprocessing pipeline for this thesis. First, there are for each customer two files. These files were previously extracted from a SQL database and provided as CSV files. One of these files contains all problem and solution entry items, and one contains all information entry items. All these individual files are concatenated and sorted by site and in ascending order by date of creation. This results in a large table where the problem and solution entry items are mixed with the separate information entry items. They are mixed in that they are brought from separate documents into the form similar in which they were created.

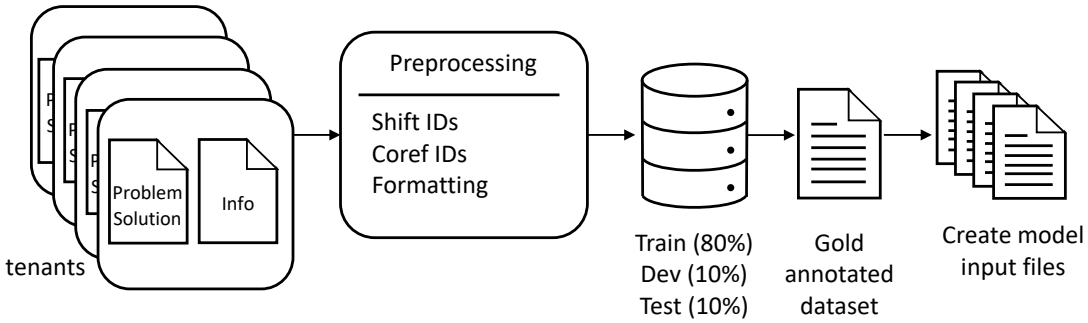


Figure 3.4 General overview of the preprocessing pipeline. First, the individual tenant files are concatenated. Second, preprocessing of the master data is conducted. Third, the dataset is split to training/validation/test datasets. Fourth, gold annotated datasets are created, containing all marked mentions and coreference cluster. Fifth, depending on the model, different input files for the models are created.

Second, some preprocessing is conducted. The preprocessing begins with cleaning the data by removing entry items with no text content or if they are in a form that cannot be processed. The cluster ids are modified from a combination of numbers and letters to simple numerical values. The shift id is determined and added for problem and solution entry items where the shift id is missing.

Third, three datasets are created for train, development, and test, with a distribution of 80%, 10%, and 10%, respectively. Each dataset's distribution is taken from each site using the tenant_id. This results in the log files of the different locations being distributed as evenly as possible across the three data sets.

Fourth, gold annotated files are created for each dataset based on the table containing all the information. These files contain information such as the mentions, the context, the documents, and the cluster id, in the form in which it is needed for further processing and evaluation.

Fifth, the specific input files required for the two models are created. These files contain the same information as the log files, only in a format that the models can use for further processing. With the help of these files, some mention-pair features are also extracted. Mention-pair features are based on features of the mentions that are compared to each other. The functional location code and the creation date time are the two features to be compared.

The functional location code determines if two mentions were created for the exact location in an industrial production environment. The more similar the functional location codes are, the closer the locations of the mentions are spatial. To determine how close two mentions are spatially, the overlap of the functional locations is obtained. Starting with the first character, the more characters that match, the greater the proximity. The functional location codes are structured so that they gradually become more specific, starting from an initial rough description. To normalize the overlap, the number of overlapping characters is divided by the longer functional location code of the mentions. Therefore, the overlap $o(f_i, f_j)$ of two functional location codes f_i and f_j is calculated by:

$$o(f_i, f_j) = \frac{\text{number of overlapping characters}}{max(len(f_i), len(f_j))} \tag{3.1}$$

When considering following pseudo functional location codes

$$f_1 = \text{AAA-123-BBB-444} \qquad f_3 = \text{AAA-999-ABC}$$
$$f_2 = \text{AAA-123-BBB-987} \qquad f_4 = \text{ZZZ-123-BBB}$$

the overlap in some of the possible combinations results in

$$o(f_1, f_2) = \frac{12}{15} = 0.80 \qquad o(f_1, f_3) = \frac{4}{15} = 0.27 \qquad o(f_1, f_4) = \frac{0}{15} = 0.00$$

This indicates that the functional locations of $o(f_1, f_2)$ are relatively close to each other, $o(f_1, f_3)$ are further apart and $o(f_1, f_4)$ are spatially absolutely separated. The normalized values of this feature is in the range of 0.0 (there is no overlap) to 1.0 (the whole functional location code overlaps).

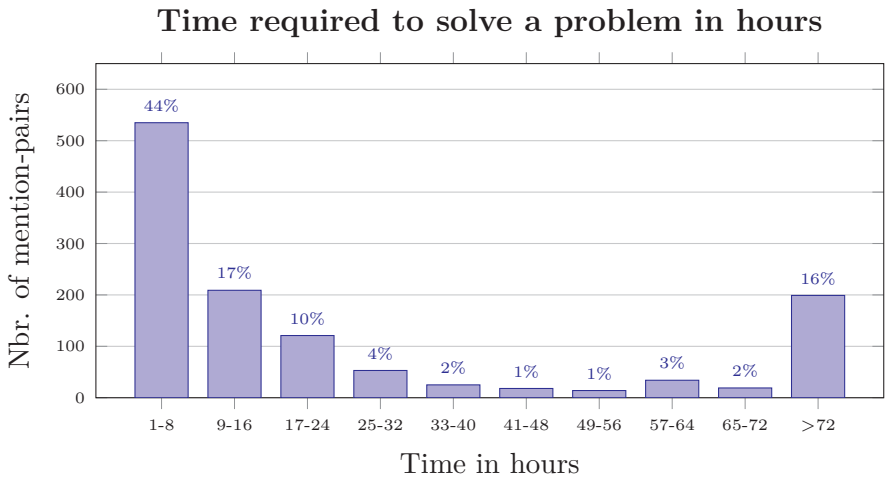


Figure 3.5 Time required to solve a problem in hours. 44% of all problems are solved after 1-8 hours, i.e., in the time frame of approximately one shift. 71% of all problems are solved within a day. Only 16% of problems are solved after more than three days (72 hours).

Figure 3.5 shows the results of an analysis of the linked problem-solution pairs. A comparison of the creation date times shows that most problems are solved relatively quickly after their occurrence. 44% of all problems are solved within a time frame of one to eight hours. Since the duration of a shift is usually eight hours, it can be assumed that a significant amount of problems are solved within one shift. Already 71% of all problems are solved within 24 hours (three shifts). This amount again underlines the assumption that most problems are solved relatively quickly. The more time passes, the fewer problems remain to be solved. A further 13% of all problems are solved between 25 and 72 hours after a problem has occurred. Only 16% of all occurring problems take longer than 72 hours to be solved.

It should be noted that the number of problem solutions after 72 hours is probably lower. This is because, in some cases, the solutions are only entered into the log file at a delayed time, which leads to a certain distortion of the data. To overcome this, for mention pairs with a time difference of more than 72 hours, a time difference between 1 and 72 hours is assigned

randomly. There is also the reverse case, where the time difference between a problem and a solution is 0 hours. This case occurs when the problem and solution descriptions are reported in one entry item. To avoid bias, since it can be assumed that such cases are always coreferent, a time difference of 1 to 72 hours is also randomly assigned for such mention-pairs.

Since a significant part of problems are solved in a time frame of 1 to 8 hours, it is more likely that a problem will be coreferent to a solution if the time difference is 3 hours instead of 50 hours. To normalize the feature value, the time difference is divided by 72 (hours). The time difference is divided by 72, as this time difference has been set as the maximum distance. The normalized values for this feature are in the range of 0.014 (the smallest time difference is set to one hour) and 1.0 (the maximum time difference of 72 hours).

Furthermore, two dataset variants are created. These variants represent different filter settings of the unprocessed log files. The specific filtering of these variants is conducted at the beginning of the preprocessing pipeline. This means that the remaining preprocessing of the two variants does not differ. The two variants differ in the following aspects.

1. The variant **without funcloc overlap** filters out cases where problem-solution pairs were reported under the same entry item with the same functional location code. This is to ensure that the results are not skewed toward these cases. There are cases where problems are reported under a central functional location, such as a reactor. However, then a solution is reported about a part of that functional location that was actually incorrect.

2. The variant **with funcloc overlap** does not exclude any entry items for the reason above.

Since the two distinct models require different input formats, the preprocessing differs in a few aspects. These are described in the following section 3.3 and section 3.4 for the NeuralCoref model and the CDLM model respectively.

3.3 NeuralCoref

As a state-of-the-art algorithm for within-document coreference resolution, the mention-ranking NeuralCoref model by HuggingFace is adapted for this thesis. This section describes the specific characteristics of the NeuralCoref model. It starts with the preprocessing for this model (Section 3.3.1), then explains the underlying architecture of the model in detail (Section 3.3.2) and finally describes the implementation and training of the model (Section 3.3.3).

3.3.1 Preprocessing

As previously mentioned, the unprocessed log files contain several entry items without an assigned shift id. Artificial shift ids for the NeuralCoref model are created as follows. A table is created for each tenant that contains only the entry items that belong to the corresponding tenant. A tenant corresponds to a specific site. Since linked problem-solution pairs are only valid if they belong to the same tenant, the shift id must reflect the assignment to a specific tenant. The affiliation of the entry items is determined using the tenant_id. The tables are sorted by ascending date. For each tenant, the first and the last date are determined. A loop is conducted from the first date to

the last date in steps of three days. For example, with the first date being 2020-02-14, the next step (i.e., day) of the loop is 2020-02-17, then 2020-02-20, etc., until the last extracted date of the specific tenant is reached. All entry items between two dates (e.g., 2020-02-14 and 2020-02-17) are assigned the same shift id. The artificial shift id is a combination of part of the tenant_id and the extracted date. A tenant_id 1AA2bc-DE34F and a date 2020-02-14 results in the shift id 1AA2bc_2020-02-14. This ensures that each entry item in the dataset has an assigned shift id.

It is possible that after the shift ids are created and assigned, there are shifts that do not have problem and solution entry items (i.e., mentions) but only information entry items. Since these shifts do not add any value in later processing, as no mention-pairs can be formed, these shifts are removed from the dataset. In addition, shifts that contain only a single problem or solution description are also removed from the dataset. This is because these shifts cannot be preprocessed in the subsequent steps due to the architecture of the NeuralCoref model.

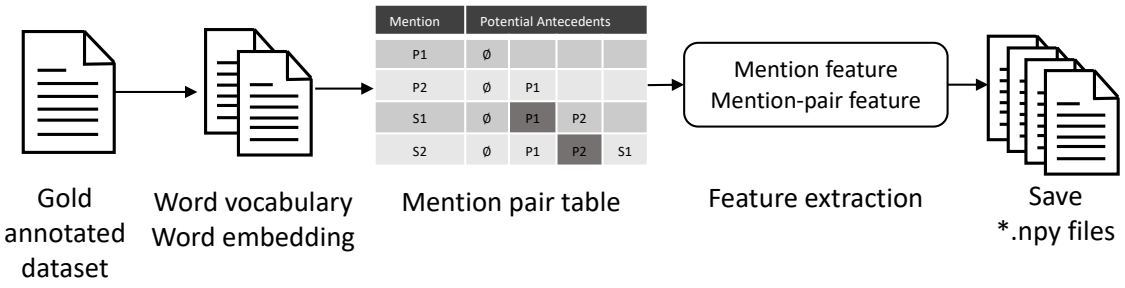


Figure 3.6 NeuralCoref preprocessing pipeline.

Figure 3.6 illustrates the NeuralCoref preprocessing pipeline. Each word of the dataset is saved in a word vocabulary text file. Word embeddings are created for all these words, using the embedding provided by *fastText* for the German language [5]. fastText is an open-source library developed by the Facebook AI Research Lab. The focus is on scalable solutions for the tasks of text classification and representation with fast and accurate processing of large amounts of data. The model is based on the skip-gram model, which uses a central word and tries to predict words within a specific range before and after the current word. The skip-gram objective function sums the log-likelihoods of the surrounding n words to the left and right of the target word w_t to obtain the following objective [32]:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n,\neq 0}\log p(w_{j+1}|w_t) \tag{3.2}$$

Instead of n words, fastText uses bags of character n-grams to represent each subword. n-grams are continuous sequences of words or characters in a document. fastText uses between 3 and 6 characters for each subword, as the best results were achieved with these parameters. Each character n-gram is assigned a vector representation. The words are represented as the sum of these representations. The authors of fastText found that using character n-grams is more useful in morphologically rich languages such as German. In the case of out-of-vocabulary (OOV) words, fastText can generate vectors by summing the vectors for the individual characters n-grams, provided at least one of the characters n-grams was present in the training data. The

fact that the model can also process OOV words is essential for creating word vectors for the vocabulary of the log files. Since the log files are created in complex industrial environments, some words, such as machine names or specific technical terms, occur only once in the entire corpus. This also explains why pre-trained word vectors cannot simply be used. The technical terms are too specific to occur in other datasets. Added to this is the complexity of manual user input in the form of unfamiliar abbreviations for words, as well as several spelling errors. Word vectors based only on more general texts with correct spelling would not be able to provide word vectors for such cases.

Mention	**Potential antecedents**					
P1	ø					
P2	ø	P1				
S1	ø	P1	P2			
S2	ø	P1	P2	S1		
P3	ø	P1	P2	S1	S2	
S3	ø	P1	P2	S1	S2	P3

Table 3.1 A generic example of NeuralCorefs mention-pair table. Coreferring mention-pairs are highlighted in grey [57].

Table 3.1 illustrates the next preprocessing step of creating a mention-pair table of all mentions within a document. All mentions of a shift, i.e., problem and solution entry items, are gathered in a table of mention-pairs to highlight the coreferring pairs. Each coreferent mention pairs are assigned a positive label (1), and non-coreferent mention pairs are assigned a negative label (0). In the table, ø means that a mention does not corefer with any previous mention. A given mention can have more than one coreferring antecedent, creating clusters of coreferring mentions. In this case, several grey boxes would appear in a single line.

There are two things to consider when training a model on such data. First, each mention has a different number of potential antecedents, which complicates the batching. The size of the mention-pairs vector ranges from 0 to N, where N is the total number of mentions in a shift. Second, the table of mention pairs scales with cN, where c is the average number of mentions in each shift of the dataset.

Category	**Mention features**			
Location	Mention norm location			
Mention	Root head embedding			
Span vector	Mention	Before	After	Doc

a Extracted features during preprocessing for single mentions for the NeuralCoref model.

Category	**Mention-pair features**
Funcloc	Overlap
Date time	Time difference
Location	Mention-pair distance

b Extracted features during preprocessing for mention-pairs for the NeuralCoref model.

Table 3.2 Overview of mention features and mentions-pairs features that are extracted for the NeuralCoref model.

The following preprocessing step is to extract the features of the single mentions and the

features of all mention-pairs. Table 3.2a shows all extracted features of the single mentions and Table 3.2b shows all extracted mention-pair features of all mention-pairs.

For the single mentions, there are a total of six features that are extracted. Five are word embeddings, and one is a float value feature. The float value feature is the mention norm location, i.e., the index of the mention, counting only the mentions within a document, divided by the total number of mentions in the document. This feature provides information about where a mention is located in the document.

The other single mention features are word embeddings. The root head embedding feature extracts the root head word of the mention and creates a word embedding for this word. The root head word is the most critical in a sentence and is used as a feature because it may provide valuable information to link a problem to a solution. For example, if a problem and a solution have the same root head word, the mentions are more likely to be coreferent.

The next feature is the average embedding of the mention. For each word of the mention, the word embedding is extracted. All these word embeddings are added and divided by the total number of words in the mention. This feature provides a general representation of the mention itself.

Two further average word embeddings are created with the context before and after a mention. The context of a mention is the entry items surrounding the mention. One average embedding is created with the two entry items before a mention. For each word in these two entry items, the word embeddings are extracted, added, and divided by the total number of words in these two entry items. The same procedure is followed for the two entry items after the mention. The last feature is the average embedding of the whole document. With these features, the context of the mention is represented.

For the mention-pairs, the following three features are extracted. The first one is the overlap of the functional location codes. The functional location describes where a problem or solution occurs in the industrial environment. The overlap is determined by identifying the overlapping characters of the functional location codes, starting with the first character of both codes. The overlaps are divided by the more extended functional location code length to normalize the feature. The more similar the functional location codes are, the closer the locations of the mentions are to each other. If there is a complete match, it can be assumed that both mentions can be assigned to the same place.

The second one is the time difference between the mentions. The time difference is determined using the creation date time. The smaller the time difference, the more likely a mention-pair is coreferring. To normalize the feature, the time difference is divided by 72.

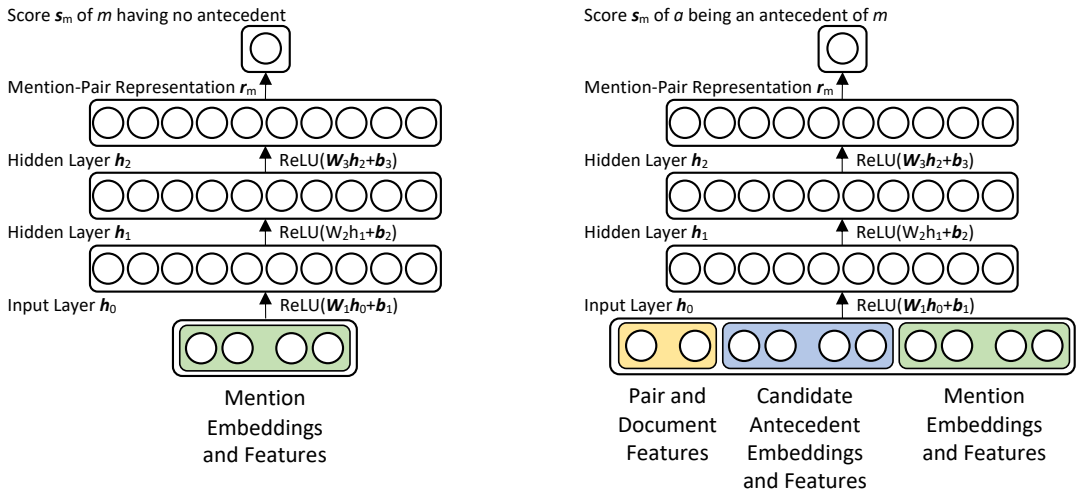
The third feature is the mention-pair distance. The mention-pair distance is determined by counting the mentions between the mention-pair. Two neighboring mentions have a distance of 0. The maximum distance is $N - 1$, where N is the number of mentions in a document.

After extracting all single mention and mention-pair features, the final step of preprocessing is to save all these features in several **.npy* files for further processing.

3.3.2 Model

The NeuralCoref is a CR model based on the neural mention-ranking model developed by Clark and Manning ([13, 14]) for within-document CR. Mention-ranking models rank pairs of mentions by the probability of coreference rather than comparing partial coreference clusters. The NeuralCoref model is designed to strike a good balance between accuracy and speed/simplicity. To train the NeuralCoref model, various mention features and mention-pair features are extracted, which form the input to two parallel feed-forward neural networks (FFNNs). Coreference clusters are built successively depending on the score (the higher, the better).

Figure 3.7 illustrates the architecture of the NeuralCoref model. The model consists of a common embedding layer that transforms mentions into vectors and feeds them into two parallel FFNNs, one for single mentions and one for mention-pairs.



a NeuralCoref FFNN for single mentions.

b NeuralCoref FFNN for mention-pairs.

Figure 3.7 Visualization of the parallel feed-forward neural networks of NeuralCoref. The architectures for the FFNNs are identical, except the input vector [14].

The scoring and the inputs to the parallel FFNNs differ in the following aspects:

- The **single mention** FFNN calculates the probability that a mention has no coreference, i.e., that there is no antecedent for the mention. The input vector for the single mention FFNN consists of several mention features, e.g., the position and size of the mention in the document, the average embedding spans of the mention itself, and the context before and after the mention.
- The **mention-pair** FFNN calculates the probability that a pair of mentions is coreferring. The input vector for the mention pair FFNN consists of the single mention vectors of the two mentions and some additional mention-pair features, e.g., whether the speaker is the same, whether the mentions match exactly (string match), and the distance between the mentions.

Table 3.3 shows all extracted single mention features and Table 3.4 shows all extracted mention-pair features of the NeuralCoref model. The NeuralCoref model has only a small number of features that are extracted for each mention and each mention-pair. In comparison, some other CR systems use up to 120 features (cf. [31, 34]).

Category	**Mention features**							
Type	Type (noun/...)			Nested?		Doc (talk/news/...)		
Location/Size	Location (int)				Length (int)			
Word indices	Head	0	Last	+1	+2	-1	-2	Root head
Span vectors	Mention		Before		After	Sentence		Doc

Table 3.3 Extracted features for single mentions of the NeuralCoref model [57].

Category	**Mention-pair features**		
Speakers	Same?	Speaker name in mention/antecedent?	
String match	Exact match?	Relaxed match?	Head match?
Locations	Distance	Sentence distance	Overlapping

Table 3.4 Extracted features for mention-pairs of the NeuralCoref model [57].

The extracted features are a mixture of real-valued vectors (e.g., the span vectors that average over word vectors), integers (e.g., word indices in a dictionary and categorical indices), and boolean features (e.g., "nested?" indicates whether one pair mention is contained within the other). All these features are combined into a single array, and the integer and boolean values are converted into floating-point values.

The feature vectors are concatenated to produce an I-dimensional vector $\boldsymbol{h}_0$, which is the input to the neural network. If the mention m has no antecedent, i.e. $a = \text{NA}$, the mention-pair features are not considered. In this case, the FFNN for single mentions is trained to produce anaphoric scores. Except for the input layers, the architectures of the FFNNs are identical. The FFNNs each consist of three hidden layers of rectified linear units (ReLU) [36] activation functions.

Each unit in a hidden layer is fully connected to the previous layer and is represented by

$$\boldsymbol{h}_i(a, m) = max(0, \boldsymbol{W}_i \boldsymbol{h}_{i-1}(a, m) + \boldsymbol{b}_i) \tag{3.3}$$

where $\boldsymbol{W}_1$ is a $M_1 \times I$ weight matrix, $\boldsymbol{W}_2$ is a $M_2 \times M_1$ matrix, and $\boldsymbol{W}_3$ is a $d \times M_2$ matrix. The output of the last hidden layer is the vector representation for the mention-pair:

$$\boldsymbol{r}_m(a, m) = \boldsymbol{h}_3(a, m). \tag{3.4}$$

All scores are compared to determine whether a mention has an antecedent and which one it should be, with the highest score being decisive. The NeuralCoref model assigns scores $s_m(a, m)$ to a mention m and an antecedent candidate a, representing the probability that the mention m has no antecedent or their compatibility for coreference. The score is generated by applying a

single fully connected layer of size one to the representation $\boldsymbol{r}_m(a, m)$ generated by the mention pair-encoder:

$$s_m(a, m) = \boldsymbol{W}_m \boldsymbol{r}_m(a, m) + b_m \tag{3.5}$$

where $\boldsymbol{W}_m$ is a $1 \times d$ weight matrix. At test time, the mention-ranking model links each mention to the highest-rated antecedent candidate.

Figure 3.8 illustrates the three training phases conducted all pairs, top pairs and ranking. A straightforward training schedule is set up to make the training fast. Each time the metrics on the development set stop improving, the training moves to the next phase.

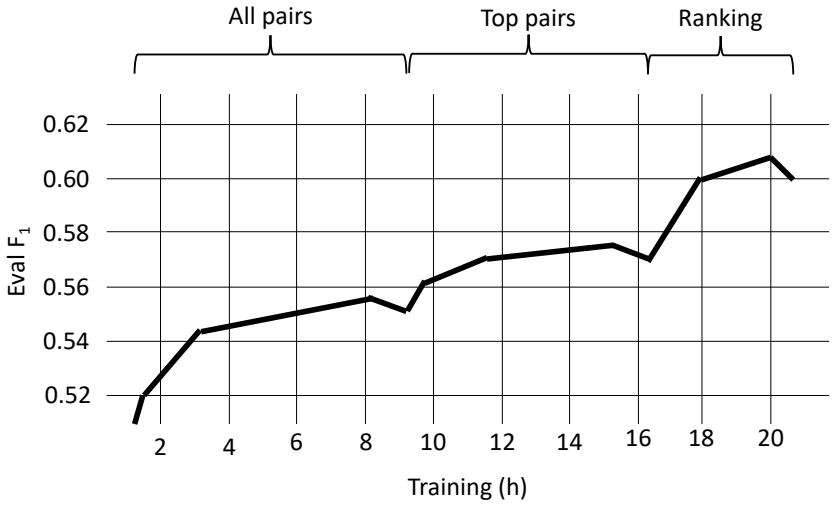


Figure 3.8 Example of NeuralCoref evaluation metric during training with the three successive training phases all pairs, top pairs, and ranking [57].

A probabilistic loss (cross-entropy) is used in the first two phases, while a slack-rescaled ranking loss is used in the last phase. The loss for each training phase is calculated as follows:

$$\text{All pair loss:} \quad -\sum_{i=1}^{N} \left[\sum_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \sum_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i)) \right] \tag{3.6}$$

$$\text{Top pair loss:} \quad -\sum_{i=1}^{N} \left[\max_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \min_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i)) \right] \tag{3.7}$$

$$\text{Ranking loss:} \quad \sum_{i=1}^{N} \max_{a \in \mathcal{A}(m_i)} \Delta(a, m_i) \left(1 + s_m(a, m_i) - \max_{t \in \mathcal{T}(m_i)} s_m(t, m_i) \right) \tag{3.8}$$

Where $\mathcal{T}(m_i)$ denotes the set of true antecedents of m_i (i.e., mentions that precede m_i and are coreferent with it, or NA if m_i has no antecedent), $\mathcal{F}(m_i)$ is the set of false antecedents of m_i, $\mathcal{A}(m_i)$ denotes the set of candidate antecedents of mention m_i (ie, mentions preceding m_i or NA) and $p(a, m_i) = sigmoid(s(a, m_i))$. And finally, $\Delta(a, m_i)$ is the error-specific cost function

$$\Delta(a, m_i) = \begin{cases} \alpha_{\text{FN}} & \text{if } a = \text{NA} \wedge \mathcal{T}(m_i) \neq \{\text{NA}\} \\ \alpha_{\text{FA}} & \text{if } a \neq \text{NA} \wedge \mathcal{T}(m_i) = \{\text{NA}\} \\ \alpha_{\text{WL}} & \text{if } a \neq \text{NA} \wedge a \notin \mathcal{T}(m_i) \\ 0 & \text{if } a \in \mathcal{T}(m_i) \end{cases}$$

for "false new", "false anaphoric", "wrong link", and correct coreference decision. The individual error penalties can adjust the CR model towards making more or fewer coreference links.

The *all pair loss* is a standard cross-entropy loss on the complete set of mention-pairs. The *top pairs loss* is also cross-entropy but is restricted to the (currently) top-scoring true and false antecedents of a mention. Finally, the *ranking loss* is a max-margin loss with a slack rescaled cost Δ. The slack-rescaled max-margin training objective from Wiseman et al. [56] encourages separation between the highest-scoring true and false antecedents of the current mention.

The training setup of the NeuralCoref model follows in large parts these as Clark and Manning. The error penalties are set to $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$. The word embeddings are initialized with 50-dimensional ones produced by *word2vec* [33] on the OntoNotes 5.0 dataset. The hidden layer sizes are set to $h_1 = 1000$, $h_2 = h_3 = 500$ and the training objective is minimized using root mean squared propagation (RMS-Prop) [20]. RMS-Prop is a gradient optimization method that deals with the problem of vanishing and exploding gradients. The normalization balances the step size, decreasing the step size for large gradients to avoid exploding and increasing the step size for small gradients to avoid vanishing. L2 regularization is applied to the model weights, and dropout [19] with a rate of 0.5 on the word embeddings and the output of each hidden layer. L2 regularization prevents overfitting by making the weights small but not exactly 0. After training (200, 200, 200) epochs each for all pairs, top pairs, and ranking, the NeuralCoref model achieves a CoNLL F_1 of about 61.2.

3.3.3 Implementation and Training

Table 3.5 lists the training, development, and test split statistics regarding documents, mentions, and coreference clusters for the dataset variants with and without overlapping functional location codes. The data split is about 80%/10%/10% for training/validation/test, respectively.

Funcloc Overlap		Train	Validation	Test
no	Docs (Shifts)	1,078	171	244
	Mentions	13,384	1,767	2,623
	Cluster	5,353	706	1,049
yes	Docs (Shifts)	1,572	229	282
	Mentions	30,059	2,878	4,348
	Cluster	12,023	1,151	1,739

Table 3.5 Datasets used to train the Neuralcoref model.

The fundamental pipeline of the NeuralCoref model remains the same as in the original. The extracted mention features and the mention-pair features are combined to produce an I-dimensional vector $\boldsymbol{h}_0$, which is the input to the neural network. If mention m has no antecedents,

i.e., $a = \text{NA}$, the mention-pair features are not included. In this case, the FFNN for single mentions is trained to produce anaphoric scores. The model assigns a score $s_m(a, m)$ to a mention m and a candidate antecedent a to determine their likelihood to be coreferent. The training is conducted in three stages all pairs, top pairs, and ranking, and moves to the next stage as soon as the metrics on the development set stop increasing.

Mention	**Potential antecedents**					
P1	ø					
P2	ø	P1				
S1	ø	P1	P2			
S2	ø	P1	P2	S1		
P3	ø	P1	P2	S1	S2	
S3	ø	P1	P2	S1	S2	P3

Table 3.6 A generic example of a table of coreferring mention-pairs. Coreferring candidate antecedents are highlighted grey [57].

The training set comprises all potential antecedents of a mention within one document. Referring to Table 3.6, sampled instances are (P1, ø), (P2, ø), (P2, P1), (S1, ø), (S1, P1), etc. This results in six positive instances (grey fields in the table) and 15 negative instances (white fields in the table). When compiling the mention-pairs of the preprocessed datasets as described, the distribution of positive and negative instances results in as shown in Table 3.7.

Funcloc		Instances	
Overlap	Dataset	Positive	Negative
no	Train	5,101	151,799
	Validation	617	14,567
	Test	938	25,299
yes	Train	18,166	503,333
	Validation	1,652	26,276
	Test	2,616	44,155

Table 3.7 Distribution of the positive and negative instances for both variants used to train NeuralCoref.

The NeuralCoref model of HuggingFace extracts 18 features and span vectors for single mentions and eight for mention-pairs. Six features and span vectors are extracted for single mentions and three features for mention-pairs for the implemented model. Features of the single mention that are no longer extracted include the type of mention (e.g., noun) or whether the mention is nested within another mention, as this is not the case in the current setting. Individual word indices are no longer extracted, as CR is not performed at the level of words but at the level of sentences and paragraphs. Similar to the original model are the averaged span vectors. However, instead of the (averaged) span vector of individual words or word pairs, the average span vectors of the entire mention, i.e., the entire sentence or paragraph, are used. In addition, the average span vectors of the two entry items in the log files are averaged before and after the mention.

For the mention-pairs, the features for the model are derived as follows. The only mention-pair feature that is still retained is the mention-pair distance. All other original mention-pair features are no longer used because the datasets and the problem definition are different. The three mention-pair features extracted are the overlap of the functional location codes, the time difference between two mentions, and the distance between the mentions in the log files. The first feature is the functional location code describes the location where, for example, a problem occurred at a machine. Therefore, if a problem's functional location matches the solution's functional location, both mentions have been created for the same location on a machine. The feature is normalized by dividing the number of overlapping characters by the length of the longer functional location code.

The second mention-pair feature is the time difference between two mentions. Since a majority of problems are solved in a time frame of 1 to 8 hours, a problem is more likely to be coreferent with a solution if the time difference is between 1 and 8 hours instead of 50 hours. To normalize the feature value, the time difference is divided by 72.

The third feature is the mention distance, i.e., the number of mentions between the mention-pair. If the two mentions are adjacent, the mention distance is 0. The maximum value is $N - 1$, where N is the number of mentions in the document.

Overall, the single mentions put more emphasis on the word embeddings than on the extracted features (there are five word embeddings on only one feature). For the mention-pairs, only three features are extracted, as well as the distances between mentions. Still, the word embeddings outweigh the features by five to four.

Instead of 50-dimensional word2vec pretrained word embeddings, 300-dimensional word embeddings pretrained by fastText for the German language are used. By using higher dimensional word embeddings, it is expected that the individual words and, thus, the meaning of the descriptions will be better represented.

The training setup of the adapted NeuralCoref model largely follows that of Clark and Manning. In the same way, the authors of the NeuralCoref model have adapted most of the training structure. The error penalties are set to $(\alpha_{\mathrm{FN}}, \alpha_{\mathrm{FA}}, \alpha_{\mathrm{WL}}) = (0.8, 0.4, 1.0)$. The word embeddings are initialized with 300-dimensional ones pretrained by fastText. The hidden layer sizes are set to $h_1 = 1000$, $h_2 = h_3 = 500$ and the training objective is minimized using RMS-Prop. The regularize the network, L2 regularization is applied to the model weights and dropout with a rate of 0.5 on the word embeddings and the output of each hidden layer. The training is conducted for 20/20/20 epochs for all pairs/top pairs/ranking.

3.4 CDLM

As state-of-the-art model for cross-document coreference resolution the transformer-based cross-document language model by Caciularu et al. is built upon. CDLM introduces a new pretraining approach geared towards multi-document language modeling. Pretraining is performed over sets of related documents and introduces dynamic global attention that has access to the entire input to predict masked tokens. The CDLM uses the Longformer as a base but can also be built on other similarly efficient transformer models. While most NLP research addresses single texts, typically at the sentence or document level, the CDLM is a new general language model

developed for the multi-document domain [7].

This section describes the specific characteristics of the CDLM model. It starts with the preprocessing of the log files conducted for this model (Section 3.4.1), then explains the underlying architecture of the model in detail (Section 3.4.2) and finally describes the implementation and training of the model (Section 3.4.3).

The implemented model makes use of additional features beyond the Transformer architecture. Since it differs from the original architecture of the CDLM and to distinguish the model by its name, the implemented version will be called *PR-CDLM* from now on.

3.4.1 Preprocessing

As previously described, the unprocessed log files contain a number of entry items without an assigned shift id. Figure 3.9 provides a schematic overview of how missing shift ids are determined for the PR-CDLM model.

type	creation date time	shift_id before		shift_id after
Information	2021-12-08_05:25:48	2021-12-08_05:00:00		2021-12-08_05:00:00
Problem	2021-12-08_05:32:29	nan		2021-12-08_05:00:00
Solution	2021-12-08_06:11:07	2021-12-08_05:00:00	①	2021-12-08_05:00:00
Information	2021-12-08_08:46:48	2021-12-08_05:00:00		2021-12-08_05:00:00
Information	2021-12-08_08:49:13	nan		nan
Problem	2021-12-08_10:08:16	2021-12-08_05:00:00		2021-12-08_05:00:00
Information	2021-12-14_17:27:36	nan		2021-12-14_17:00:00
Solution	2021-12-14_17:30:51	nan		2021-12-14_17:00:00
Solution	2021-12-14_17:34:32	nan	②	2021-12-14_17:00:00
Information	2021-12-14_18:15:02	nan		2021-12-14_17:00:00
Problem	2021-12-14_19:02:54	nan		2021-12-14_17:00:00

Figure 3.9 Schematic representations of how missing shift ids are determined. Top section shows how missing shift ids are determined using shift ids from entry items with the same creation date. Bottom section shows how missing shift ids are approximated when no shift id could be determined. The date and the full hour from the first problem or solution are used to fill the gaps.

The process for determining the shift ids is divided into two stages. First, an attempt is made to assign missing shift ids using existing shift ids of entry items with the same date and the same tenant_id. If a problem or solution entry item is found, that does not have a shift id, the creation date is determined from this entry item using the creation date time. The extracted date is used to check whether there is an entry item that was created on the same day and has an assigned shift id. If such an entry item is found, the shift id is copied from the found entry item. The copied shift id is then assigned to each problem and solution entry item with the same date in case the respective entry item does not have a shift id. For example, the first problem entry item in the top section of Figure 3.9 has no assigned shift id. The date 2021-12-08 is extracted from the creation date time. This date is used to check whether there are any entry items with the same date and an assigned shift id. The first information entry item has the exact date and an assigned shift id. Consequently, the shift id from the information entry item is used for the problem entry item. In this case, information items are not considered to leave the data as accurate to the original as possible.

Second, if no shift id could be determined via the first stage, an artificial shift id is created. The artificial shift id is generated from the creation date time and corresponds to the date and the rounded-off full hour. For example, the first solution entry item in the bottom section of Figure 3.9 has no assigned shift id. In this case, none of the other entry items with the same creation date have an assigned shift id. Hence, a new shift id must be created. Using the creation date time 2021-12-14 17:30:51 of the solution entry item results in the shift id 2021-12-14_17:30:51. The newly created shift id is also used for all entry items with the same creation date that do not have a shift id. This creates a new artificial shift, including problem, solution, and information entry items.

It is possible that after the artificial shift ids are created and assigned, there are shifts that do not have problem and solution entry items but only information entry items. Since these shifts do not add any value in later processing, as no mentions can be processed, these shifts are removed from the dataset.

The preprocessing of the PR-CDLM model is finalized by creating two files. One file contains only all mentions, including further information such as cluster id, funcloc, and creation date time. Moreover, one file contains all shift entry items, including the information entry items. Both files are saved as **.json*.

3.4.2 Model

The original CDLM is a transformer-based CR model developed for cross-document CR. Existing language models (LMs) (cf. [18, 28, 41]) are pretrained with variants of the masked language modeling (MLM) self-supervised objective. These LMs provide powerful representations for internal text structure (cf. [12, 47]), which are also beneficial for various multi-document tasks (cf. [59, 63]).

Instead of pretraining over individual documents, the CDLM is trained over groups of related documents in which informative cross-textual relationships are abundant. The model is intended to learn to consider and represent such relationships, as they provide valuable signals for optimizing the goal of language modeling.

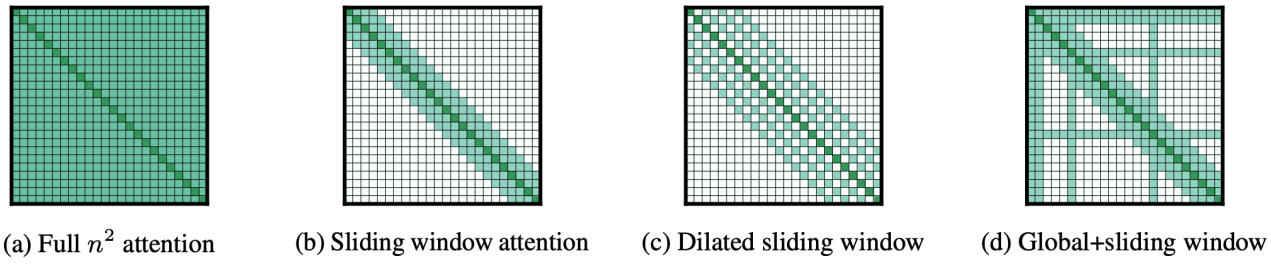


Figure 3.10 Comparing the full self-attention pattern and the configuration of attention patterns of the Longformer [4].

Pretraining and finetuning across multiple documents requires a model handling large amounts of text. Long-range LMs such as the Longformer [4] were designed to extend the capabilities of earlier transformers [53] to process longer sequences of text. The CDLM is based on the

Longformer, which sparsifies the full self-attention matrix in transformers by combining a localized sliding window (called local attention) and a global attention pattern at some specific input locations (Figure 3.10).

Two key elements are introduced in pretraining the CDLM: First, pretraining over sets of related documents that contain overlapping information, and second, pretraining with a dynamic global attention pattern over masked tokens to reference the entire cross-textual context.

The CDLM approach to cross-document language modeling is based on pretraining the model on document sets describing the same topic. For the CDLM language modeling, the ECB+ corpus [16] is used because it contains several related documents that are readily available. In this way, the model is encouraged to learn cross-text mapping and alignment capabilities that can be used for improved unmasking. This pretraining strategy causes the model to use cross-document information and helps with several downstream cross-document tasks.

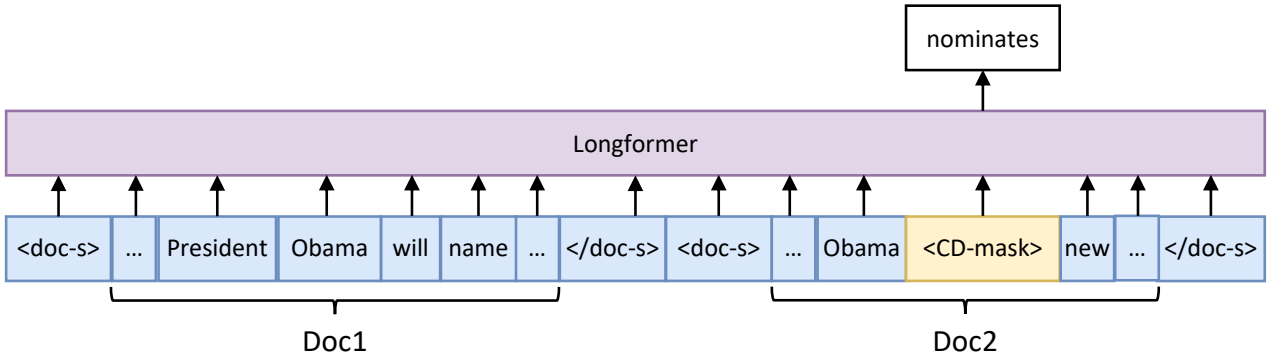


Figure 3.11 Visualization of the CDLM pretraining [7]: The input consist of concatenated documents, separated by special document separator tokens. The masked (unmasked) token colored in yellow (blue) represents global (local) attention. The goal is to predict the masked token *nominates*, based on the global context, i.e., the entire set of documents.

Figure 3.11 illustrates the CDLM pretraining procedure. The CDLM is based on the Longformer model to support the contextualization of information across multiple documents, as an efficient Transformer model is needed that scales linearly with input length. However, the setup of CDLM is general and can be applied to other similar efficient Transformers. To concatenate related documents, new special document separator tokens, ⟨`doc-s`⟩ and ⟨`/doc-s`⟩, are used to mark document boundaries. A masking procedure similar to BERT is used: For each training example, a sample of tokens (15%) is randomly chosen to be masked. The masked tokens are attempted to be predicted considering the entire document set by assigning global attention to them using the global attention weights. In this way, the Longformer contextualizes information both across documents and across wide-ranging dependencies within a document. The non-masked tokens use local attention by using the local attention weights.

Figure 3.12 illustrates the setup for the cross-document CR pairwise scoring model of the CDLM. The documents containing the mentions are concatenated using the special document separators and encoded using the CDLM together with the token `[CLS]` at the beginning of the sequence. Only the single input document is used with a set of document separators for within-document coreference candidates. Additionally, the candidate mentions are marked as inspired

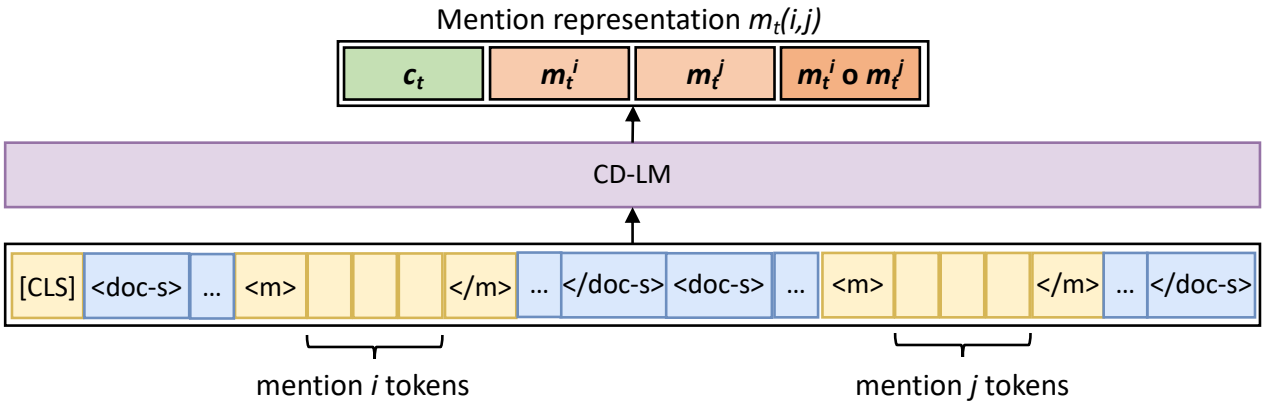


Figure 3.12 Representation of the cross-document coreference resolution pairwise mention scorer of the CDLM. m_t^i, m_t^j and c_t are the cross-document contextualized representation vectors for mentions i and j, and of the `[CLS]` token, respectively. $m_t^i \circ m_t^j$ is the element-wise product between m_t^i and m_t^j. $m_t(i,j)$ is the final produced pairwise-mention representation. The tokens colored in yellow represent global attention, and tokens colored in blue represent local attention [7].

by Yu et al. [60]. Each mention is wrapped with special tokens ⟨m⟩ and ⟨/m⟩ to direct the model to pay attention to the candidate representations specifically. Global attention is assigned to ⟨m⟩, ⟨/m⟩, `[CLS]`, and the mention tokens themselves. The final pairwise-mention representation is formed like Zeng et al. [61] and Yu et al. [60]: The cross-document contextualized representation vector for the t^{th} sample is:

$$m_t(i,j) = \left[c_t, m_t^i, m_t^j, m_t^i \circ m_t^j\right] \tag{3.9}$$

where $[\cdot]$ denotes the concatenation operator, c_t is the cross-document contextualized representation vector of the `[CLS]` token, and each of m_t^i and m_t^j is the sum of candidate tokens of the corresponding mentions (i and j). The pairwise scorer is trained according to the suggested finetuning scheme.

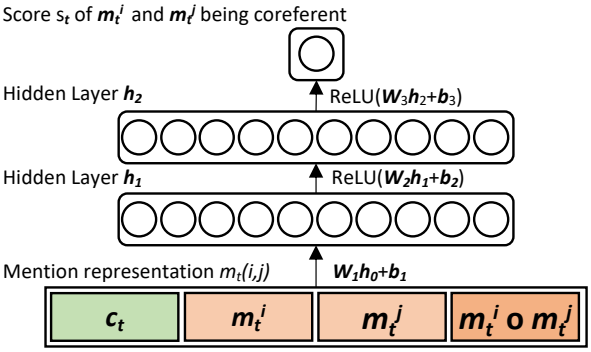


Figure 3.13 Visualization of the mention-pair scoring model of CDLM.

Figure 3.13 shows the mention representation vector $m_t(i,j)$, which is also the I-dimensional

input vector $\boldsymbol{h}_0$ for the neural network. The input to hidden layer $\boldsymbol{h}_1$ is represented by

$$\boldsymbol{h}_1(i,j) = \boldsymbol{W}_1\boldsymbol{h}_0(i,j) + \boldsymbol{b}_1 \tag{3.10}$$

where $\boldsymbol{W}_1$ is a $M_1 \times I$ weight matrix. The inputs to hidden layer $\boldsymbol{h}_2$ is represented by

$$\boldsymbol{h}_2(i,j) = max(0, \boldsymbol{W}_2\boldsymbol{h}_1(i,j) + \boldsymbol{b}_2) \tag{3.11}$$

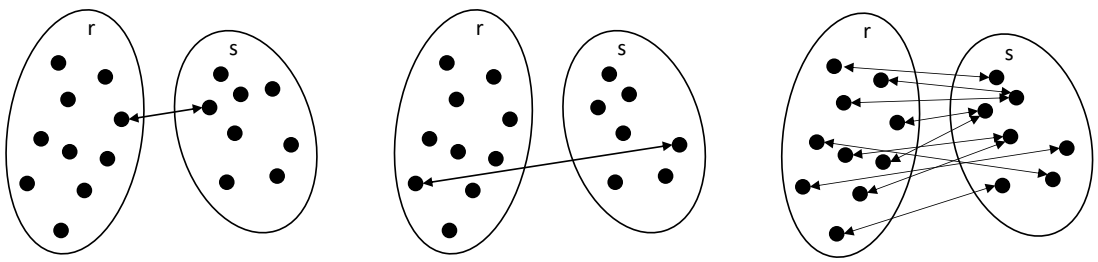
where $\boldsymbol{W}_2$ is a $M_2 \times M_1$ matrix. The output of the CDLM model is a score $\boldsymbol{s}_t(i,j)$ of the likelihood that a mention-pair, m_t^i and m_t^j, is coreferring. This is produced by applying a single fully connected layer of size one to the hidden layer $\boldsymbol{h}_2(i,j)$ produced by:

$$\boldsymbol{s}_t = max(0, \boldsymbol{W}_t\boldsymbol{h}_2(i,j) + b_t) \tag{3.12}$$

where $\boldsymbol{W}_t$ is a $1 \times d$ weight matrix.

At test time, agglomerative clustering is applied to merge the most similar cluster pairs. Agglomerative clustering is a type of hierarchical clustering that is a general family of clustering algorithms that form nested clusters by merging or splitting them sequentially [35]. Agglomerative clustering performs hierarchical clustering with a bottom-up approach: Each observation starts in its own cluster, and the clusters are merged one after the other. In agglomerative clustering, small clusters are combined into large clusters until a termination criterion is met. There are different linkage criteria to determine the metric used for the merging strategy (see Figure 3.14):

- **Single linkage** minimizes the distance between the closest observation of cluster pairs and is calculated by $L(r,s) = \min(D(x_{ri}, x_{sj}))$.
- **Complete linkage** minimizes the maximum distance between observations of cluster pairs and is calculated by $L(r,s) = \max(D(x_{ri}, x_{sj}))$.
- **Average linkage** minimizes the average of the distance between all observations of cluster pairs and is calculated by $L(r,s) = \frac{1}{n_r n_s}\sum_{i=1}^{n_r}\sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$.



a Single linkage. **b** Complete linkage. **c** Average linkage.

Figure 3.14 Visualization of different agglomerative clustering linkage methods.

The training set for the CDLM model is composed of positive instances, which consist of all the mention pairs that belong to the same coreference cluster, while negative examples are randomly sampled. The resulting features vector is passed through a multi-layer perceptron

(MLP) pairwise scorer where $h_1 = 3072$ and $h_2 = 768$. The finetuning process for cross-document coreference resolution took ∼28 and ∼45 hours per epoch, for event coreference and entity coreference, respectively. The results on event and entity CDCR on the ECB+ test set are CoNLL F_1 scores of 85.6 and 82.9, respectively, outperforming the previous state-of-the-art of +1.2 (event) and +7.6 (entity).

3.4.3 Implementation and Training

Table 3.8 lists the statistics on training, validation, and testing distribution of documents, mentions, and clusters for both dataset variants, the one with and the one without the overlap of functional location codes. That data split is about 80%/10%/10% for training/validation/test, respectively.

Funcloc Overlap		Train	Validation	Test
no	Docs (Shifts)	2,150	341	525
	Mentions	13,222	1,451	2,285
	Cluster	5,379	775	1,105
yes	Docs (Shifts)	3,833	576	729
	Mentions	30,089	2,501	3,877
	Cluster	12,149	1,162	1,624

Table 3.8 Datasets used to train the CDLM model.

The original CDLM model was trained on the ECB+ corpus. This corpus contains several topics, each described with several documents. The model is trained on topics, i.e., across multiple documents, to learn cross-document relationships. Therefore, it is a cross-document coreference model. The preprocessed dataset in the implemented setup is structured differently. In the dataset, each topic consists of only one document. Nevertheless, only pairs of mentions are formed from mentions that belong to the same topic, which in the current setting means that the mentions come from only one document. Therefore, the implemented PR-CDLM model must be defined as a within-document coreference resolution model, as no cross-document relationships are considered.

Funcloc		Instances	
Overlap	Dataset	Positive	Negative
no	Train	12,186	12,186
	Validation	1,000	1,000
	Test	1,565	1,565
yes	Train	21,171	21,171
	Validation	2,187	2,187
	Test	3,809	3,809

Table 3.9 Distribution of the positive and negative instances for both dataset variants used to train CDLM.

Table 3.9 shows the actual number of positive and negative instances for training and validating the PR-CDLM model. The training set comprises all positive mention-pairs on a document and a random sample of negative mention-pairs from the same document of the same size. For this purpose, all possible combinations of mention pairs within a document are formed. Then the number of positive instances is determined, which in turn equals the number of randomly selected negative instances from the formed mention-pairs of the same document. The datasets thus consist of 50% each of positive and negative instances. In all three data sets, the variant with the overlap of functional location codes has about twice as many instances as the variant without the overlap. This is to be expected as the variant with overlap consists of twice as many documents, mentions, and clusters.

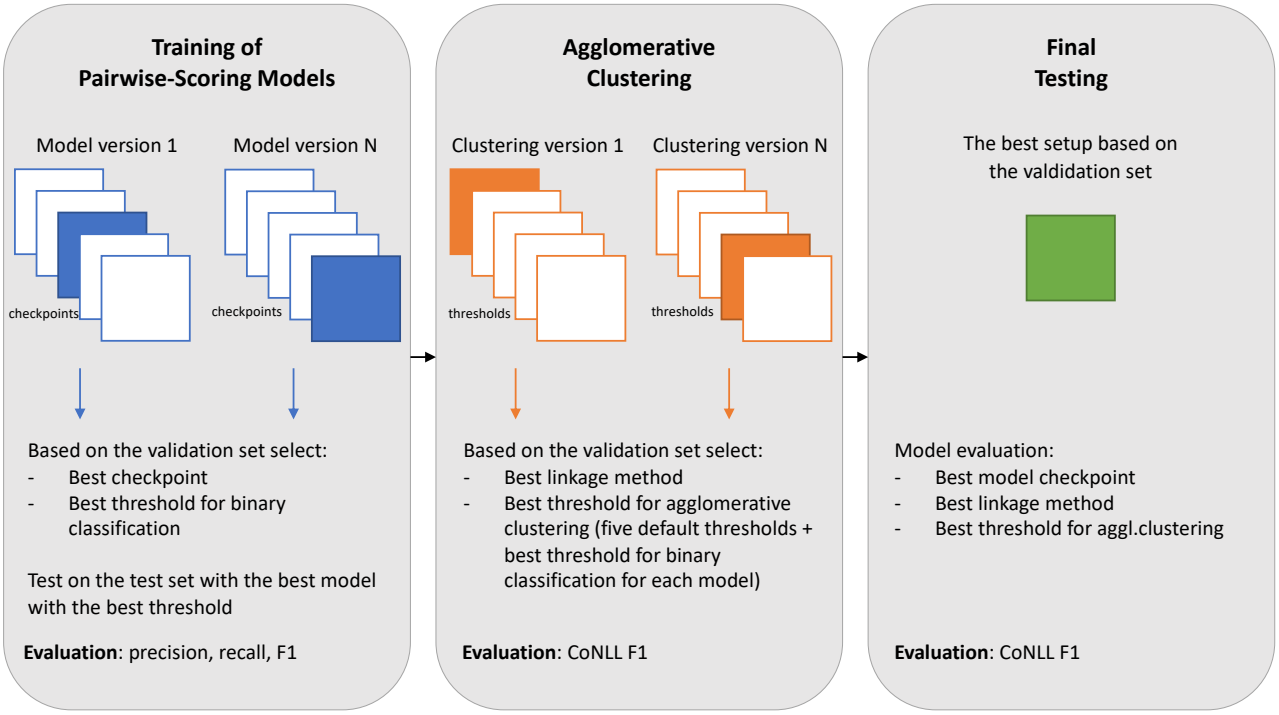


Figure 3.15 Schematic overview of the implemented training pipeline of the PR-CDLM model. The pipeline consists of three major stages. First, the training of the pairwise-scoring model. Different thresholds are tested to classify instances as positive or negative. Second, agglomerative clustering is conducted to form the final cluster, using different linkage methods and thresholds. Third, the best performing model, based on the F_1-score in the second stage, is tested on the test set.

Figure 3.15 visualizes the training pipeline of the implemented PR-CDLM model. The pipeline consists of three major stages. The first stage is training the pairwise-scoring model. The best threshold for binary classification (coreferring or not) is determined at this stage. The second stage is agglomerative clustering. Mention-pairs are merged into cluster based on their score from the pairwise-scoring model, which forms the coreference chains. The best agglomerative clustering method is determined using different linkage methods and several thresholds. The third stage, the final testing, tests the best-performing model identified in the agglomerative clustering stage. The best model is determined by the best F_1 result of the agglomerative clustering phase.

The model is evaluated after each epoch using the validation set to train the pairwise-scoring model. One checkpoint is conducted after one executed training epoch, i.e., after training for one epoch on the training set, the current model modifications are used to test them on the validation set. For each checkpoint, three different thresholds are tested. The pairwise classification of whether two mentions corefer is a binary classification task. The decision whether a pair of mentions is coreferring or not depends on the score of the pairwise scoring model. By default, the deciding threshold is 0.5, i.e., if the assigned score s is between $0.0 \leq s < 0.5$, the mention-pair is classified as non-coreferent, and if the score is between $0.5 \leq s \leq 1.0$ the mention-pair is classified as coreferent. Since the task setting is relatively complex, it is assumed that the scoring of the pairwise-scoring model is more uncertain. Therefore, different thresholds are used to check if there is a better threshold than the default value of 0.5. The additional thresholds are determined using the threshold tuning method and the precision-recall method. The methodologies for identifying the best threshold are explained in Section 4.1.1. To complete the training of the pairwise scored model, the model with the best performance, based on the highest achieved F_1 score, is tested on the test set.

The next stage is agglomerative clustering, where the final clusters are formed successively. Although the best pairwise scored model was identified during training, agglomerative clustering is performed for each model test point to create the final mention cluster. The rationale is that just because a model performed best in binary classification does not necessarily mean that it is the best model for clustering. For each agglomerative clustering, three linkage methods are tried: single, average, and complete linkage. Single linkage minimizes the average distance between the closest observation of cluster pairs. Average linkage minimizes the average distance between all observations of cluster pairs. Complete linkage minimizes the maximum distance between observations of cluster pairs. For each linkage method, five different default distance thresholds (0.3, 0.4, 0.5, 0.6, 0.7) are applied, and the best determined threshold of the mention scoring model to check whether this threshold is also beneficial for the agglomerative cluster model. The agglomerative clustering is evaluated using the CoNLL F_1 score.

Finally, the best setup based on the CoNLL F_1 score of the validation set is tested on the test set. Therefore, the test model is based on the checkpoint, linkage method, and agglomerative clustering threshold of the model with the highest score. The setup is evaluated using the CoNLL F_1 metric.

Model	Transformer	Truncation	Special Tokens
CDLM	Longformer	4096	`[CLS]`, ⟨m⟩, ⟨/m⟩, ⟨doc-s⟩, ⟨/doc-s⟩
PR-CDLM	SBERT-all-Mini-L12-v2	256	`[CLS]`, `[SEP]`

Table 3.10 Comparison of the Transformers on which the original CDLM model is built upon and on which the implemented version is built upon.

Table 3.10 highlights the differences between the Transformers used for the original CDLM model and the implemented PR-CDLM model. In contrast to the original CDLM model, no pretraining is conducted as part of this thesis. The implemented PR-CDLM is, therefore, not based on the Longformer, as there is no pretrained Longformer in German. Instead, the model builds on the Sentence-BERT (SBERT) model `all-MiniLM-L12-v2` [46], as a pretrained version is available in German. The SBERT model is intended to be used as a sentence and

short paragraph encoder and maps the input to a 384-dimensional dense vector space for further processing. By default, input texts longer than 256 tokens are truncated. Consequently, only much shorter texts can be used as input for the model than in the original CDLM model, which is based on the Longformer and is only truncated if the input text is longer than 4096 tokens. Also, special tokens such as ⟨m⟩ and ⟨/m⟩ cannot be used to delimit mentions or ⟨doc-s⟩ and ⟨/doc-s⟩ to mark document boundaries because the meaning of the tokens is not known to the model without additional pretraining.

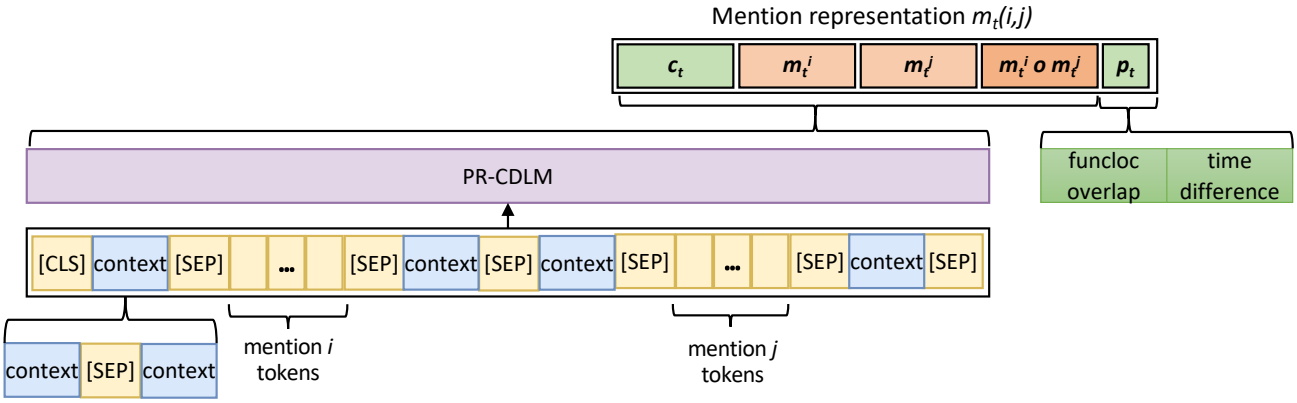


Figure 3.16 Representation of the implemented CR pairwise mention scorer. m_t^i, and m_t^j and c_t are the document contextualized representation vectors of the mentions i and j, and of the `[CLS]` token, respectively. $m_t^i \circ m_t^j$ is the element-wise product between m_t^i and m_t^j. p_t is the representation of the mention-pair features. $m_t(i,j)$ is the final produced mention-pair representation. The tokens colored in yellow represent global attention, and the tokens colored in blue represent local attention (adapted from [7]).

Figure 3.16 illustrates the CR pairwise scoring setup. The mentions i and j are enclosed by the context of each mention, i.e. the two entry items before and after each mention are appended to each mention and separated by the token `[SEP]`. In this sense, the model is more similar to the previous state-of-the-art of Zeng et al. [61], and Yu et al. [60] who train the pairwise scorer with the local context of the mention pairs. The local context represents the sentences in which the mentions occur. In the implemented PR-CDLM model, the local context thus consists of the two entry items before and after a mention. The whole mention pair is encoded using PR-CDLM together with the token `[CLS]` at the beginning of the sequence and the sentence separation tokens `[SEP]`.

The encoded mention-pair is concatenated with two additional mention-pair features. The first feature is the overlap of the functional location code. The functional location determines where in a plant or machine, for example, a problem has occurred. If a problem and a solution have the same functional location, they are more likely to be coreferent because they occur in the same spatial location. The feature value is the number of overlapping characters, starting with the first character, divided by the length of the longer technical place code of the two mentions. Therefore, the feature is normalized to values between 0.0 (no overlap) to 1.0 (complete functional location code overlaps).

The second feature is the time difference between the two mentions. As most problems are solved in 1-8 hours (i.e., one shift), it is more likely that a problem is coreferring with a solution when the time difference is, e.g., 3 hours instead of 50 hours. To normalize the feature value, the time difference is divided by 72 (hours). The time difference is divided by 72 because most problems are solved within 72 hours. For all mention-pairs where a time difference of more than 72 hours is extracted, a new time difference between 1-72 hours is randomly assigned. If a problem is entered simultaneously with the solution, the time difference is 0. In this case, a new time difference between 1-72 hours is also randomly added. Therefore, the feature is normalized to values between 0.014 (the smallest time difference is set to 1 hour) and 1.0 (the maximum time difference of 72 hours).

The final pairwise-mention representation is formed similarly to the original CDLM except for the additional mention-pair features. The contextualized representations vectors are concatenated for the t^{th} sample:

$$m_t(i,j) = \left[c_t, m_t^i, m_t^j, m_t^i \circ m_t^j, p_t\right], \tag{3.13}$$

where $[\cdot]$ denotes the concatenation operator, c_t is the contextualized representation of the `[CLS]` token, each of m_t^i and m_t^j is the sum of candidate tokens of the corresponding mentions (i and j), and p_t is the mention-pairs feature representation.

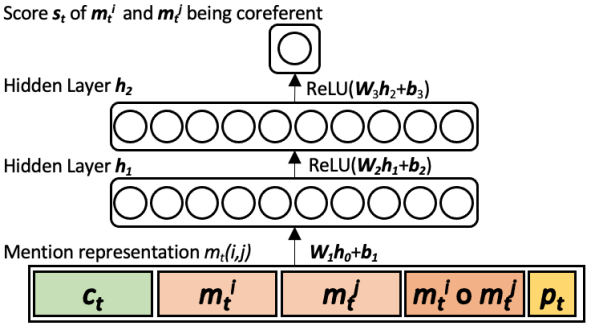


Figure 3.17 Visualization of the neural network of CDLM.

Figure 3.17 shows the mention representation $m_t(i,j)$, which is also the input vector $\boldsymbol{h}_0$ to the pairwise-scoring model. The FFNN consists of two hidden layers with ReLU activation where $h_1 = 1538$ and $h_2 = 384$. The training is conducted for five epochs.

3.5 Summary

This thesis aims to develop a CR model that can process larger text spans, i.e., at the level of sentences and paragraphs. As a database, the log files from a processing industry are taken. These log files contain three descriptions: problems, solutions, and information. These log files' problem and solution entry items are defined as mentions for the CR task. Since the mentions and labels are already known, the first step of a typical CR pipeline, the mention extraction, is not part of this thesis.

Two state-of-the-art CR models are identified to solve this task. The first is HuggingFace's NeuralCoref, an adaptation of Clark and Manning's mention ranking model. The second is a transformer-based CDLM model by Caciularu et al. developed for cross-document CR.

This section summarises the differences in the preprocessing of the log files for the two models (Section 3.5.1), as well as the most critical aspects of the model architectures (Section 3.5.2).

3.5.1 Summary of Preprocessing

In order to be able to use the data of the log files generated in an industrial environment, some preprocessing steps have to be performed. The unprocessed log files must be combined, and irrelevant data must be dropped. The text types of the entry items are limited to the three categories problem, solution, and information, and are already given. Problems and solutions are defined as mentions. The mention extraction is out of scope for this thesis.

Although most of the data preprocessing is the same for both models, the NeuralCoref model and the PR-CDLM model require different formats of model input data. A significant difference is that in the NeuralCoref model, the mention and mention-pair features are already extracted during preprocessing. For the PR-CDLM model, feature extraction is not part of the preprocessing.

Another critical difference is how the missing shift ids are generated. While the PR-CDLM model uses a complex two-stage procedure to determine the missing shift ids, the NeuralCoref model generates the shift ids in a simple way.

Individual datasets are created for both models. Furthermore, there are two variants of the datasets for both models. One includes the entry items with the overlap of the functional location codes and one variant without the overlap of the functional location codes. In the variant without overlapping functional location codes, cases are filtered out in which problem-solution pairs were reported under the same entry item with the same functional location code. This is to ensure that the results are not biased toward these cases.

Table 3.11 shows both models' final datasets after preprocessing. The most striking difference between the NeuralCoref and the PR-CDLM dataset is the number of documents (the documents correspond to the shifts). When looking at the variants with and without overlapping functional location, it is noticeable that NeuralCoref has less than half as many documents as PR-CDLM. This is due to the different ways the missing shift ids are generated. However, the number of mentions and clusters remains roughly the same for both datasets (concerning the specific variants). Since the number of mentions and clusters is about the same, but there are only about half as many documents for the NeuralCoref dataset, it can be assumed that the NeuralCoref documents contain, on average, twice as many mentions and clusters as the PR-CDLM documents.

Another significant effect is caused by the different variants with and without overlap. Looking at the training dataset, it is noticeable that the difference in size between the variants is over 50%. The number of mentions and clusters in the variant without overlap of NeuralCoref and PR-CDLM are each less than half of the variant with overlap. For the validation and test datasets, the difference is only about 40%. Since the datasets were created in such a way that, from the original, time-sorted dataset, the first 80% were used for training, the next 10% for validation, and the last 10% for testing, it can be concluded that the effect of filtering is most likely to affect

	Funcloc Overlap		Train	Validation	Test
Neuralcoref	no	Docs (Shifts)	1,078	171	244
		Mentions	13,384	1,767	2,623
		Cluster	5,353	706	1,049
	yes	Docs (Shifts)	1,572	229	282
		Mentions	30,059	2,878	4,348
		Cluster	12,023	1,151	1,739
PR-CDLM	no	Docs (Shifts)	2,150	341	525
		Mentions	13,222	1,451	2,285
		Cluster	5,379	775	1,105
	yes	Docs (Shifts)	3,833	576	729
		Mentions	30,089	2,501	3,877
		Cluster	12,149	1,162	1,624

Table 3.11 Datasets after the preprocessing of the log files. The different approaches of generating shift ids results in the Neuralcoref dataset having less than half as many documents (shifts) as the PR-CDLM dataset. The number of mentions and clusters nevertheless remains about the same for both datasets.

the earlier entries in the log files.

3.5.2 Summary of Models

Two state-of-the-art models of CR are built upon to solve the research task. The first model is the NeuralCoref model, an adaptation of Clark and Manning's neural mention ranking model. The model's heart consists of two parallel FFNNs, one for single mentions and one for mention pairs. The input to the FFNNs comprises various mention features and mention-pair features and spans comprising of the mention and the candidate antecedent, as well as the context of the mentions. The output of the FFNNs are scores of how likely it is that a mention m and a candidate antecedent a are coreferent. The higher the score, the higher the likelihood. Coreferences are resolved for within-document relations. Neuralcoref reaches a CoNLL F_1 score of 61.2.

The second model is the transformer-based cross-document language model by Caciularu et al. This model is developed for cross-document CR and is therefore trained over multiple documents covering the same topic. To be able to process large amounts of text, the CDLM is built on the Longformer. The input to the model comprises the complete documents containing the mention-pairs. Special tokens are introduced to direct the model to specifically pay attention to the candidate representations, using global attention. The CDLM is developed to resolve coreferences for cross-document relations and achieves for entity CDCR a CoNLL F_1 score of 82.9.

Model	Scope	Algorithm	CoNLL F1
Neuralcoref	within-document	neural mention-ranking	61.2
CDLM	cross-document	transformer-based	82.9

Table 3.12 Comparison of the key data of Neuralcoref and CDLM.

Table 3.12 compares the two CR models. The most critical data, such as the scope (within-document/cross-document) to resolve coreferences and the respective algorithms on which the models are based, are listed. CDLM outperforms Neuralcoref by 21.7 points on the CoNLL F_1 score metric. However, it must be noted that the mention extraction is part of the Neuralcoref model, whereas CDLM uses gold-annotated mentions. Therefore, a direct evaluation of the results is only possible to a limited extent.

4 Evaluation

The evaluation chapter presents the metrics to evaluate binary classification models and coreference resolution models (Section 4.1). Next, it shows the obtained results and highlights the limitations of the NeuralCoref (Section 4.2) and CDLM (Section 4.3) models. Finally, the chapter concludes with a summary of the critical findings (Section 4.5).

4.1 Metrics

Machine learning classification tasks are typically evaluated using precision, recall, and F_1-scores. Coreference resolution, however, is usually evaluated by the CoNLL F_1-score, which combines the three metrics MUC, B^3, and CEAF to translate the scores better of a system into real-world performance.

This section explains the metrics for binary classification and coreference resolution models.

4.1.1 Metrics for Classification Models

The task of predicting whether a mention-pair is coreferent or not is a binary classification task. A classification model predicts the probability that an instance belongs to one class or another. Metrics to evaluate how well the model performs include accuracy, precision, recall, and F_1-score.

To calculate these metrics following categories of predictions are used:

- **True Positive (TP):** True positive measures the extent to which the model correctly predicts the positive class and indicates how well the model performs on positive instances.
- **False Positive (FP):** False positive measures the extent to which the model wrongly predicts the positive class to instances that are actually negative and is the proportion of all negative examples that are predicted as positive.
- **True Negative (TN):** True negatives are the outcomes that the model correctly predicts as negative.
- **False Negative (FN):** False negative measures the extent to which the model predicts an instance as negative when it is actually positive.

These metrics can be visualized with a confusion matrix as shown in Figure 4.1. Such a matrix helps with determining the performance of a classification model.

The precision score measures the proportion of positively predicted labels that are correct. Precision can be thought of as a measure of exactness or quality. If the goal is to minimize false negatives, a model with high precision should be chosen. The precision score is useful for predicting success when the classes are imbalanced. Mathematically, it represents the ratio of true positives to the sum of true positives and false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.1}$$

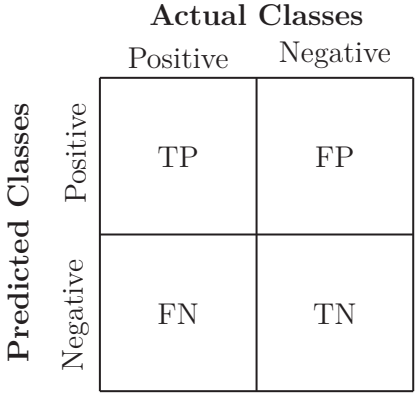


Figure 4.1 Overview of a 2x2 confusion matrix. The predicted classes are compared with the actual classes. The comparison shows directly where possible weaknesses lie of a classification model.

The recall score represents the ability of the model to predict the positives out of actual positives. Recall is also known as sensitivity or the true positive rate. A high recall score indicates that the model is good at identifying positive examples. Mathematically, it represents the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

Accuracy is a classification model performance metric defined as the ratio of true positives and negatives to all positive and negative observations. Accuracy indicates how often the classification model will predict the correct label from all made predictions. Mathematically, it represents the ratio of the sum of true positives and true negatives out of all predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \tag{4.3}$$

The main problem with accuracy is that it hides the details of the model's performance. Also, the accuracy metric is unreliable for models trained on imbalanced datasets. For a dataset with a 95% imbalance, a classifier could predict the major class in all cases and archive an accuracy of 95%, which is not a useful evaluation in this case.

The F_1-score represents the model score as a function of precision and recall. F_1-score is a performance metric that gives equal weight to precision and recall for measuring the model's performance in terms of accuracy. It is useful in the scenario where the classification model is to be optimized for precision or recall. Mathematically, it represents the harmonic mean of precision and recall.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.4}$$

The threshold of a binary classification model to decide if a prediction is negative or positive is, by default, 0.5. If the predicted score $\boldsymbol{s}$ is between $0.0 \leq \boldsymbol{s} < 0.5$, the instance is classified as negative, and if the score is between $0.5 \leq \boldsymbol{s} \leq 1.0$ the instance is classified as positive. When dealing with imbalanced classification problems or a task in which predictions are made with higher uncertainty, it may be useful to try different thresholds, such as the default threshold of

0.5. As such, a straightforward approach to improving the performance of a classifier is to tune the threshold.

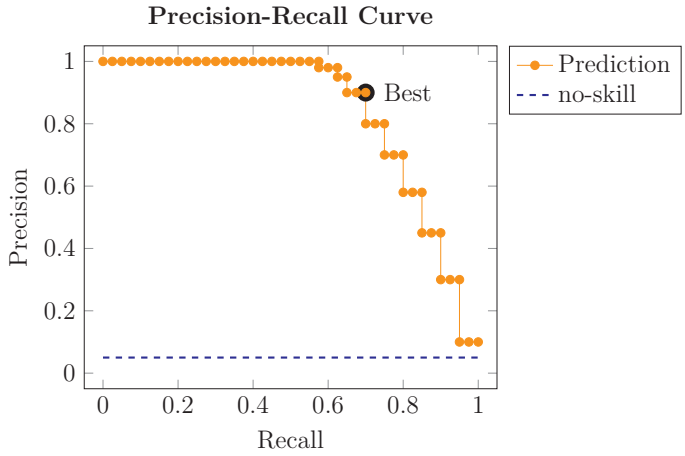


Figure 4.2 Example precision-recall curve for an imbalanced dataset. The orange line represents various precision and recall values for different thresholds. The dashed blue line represents the no-skill line, i.e., a classifier which cannot make useful predictions. The best value is calculated by the highest F_1-score of all the precision and recall combinations.

One approach to identifying the best threshold is the precision-recall method. A precision-recall curve is created to visualize the method by calculating the precision and recall scores for a set of thresholds. Different thresholds lead to different outcomes of predicted positive and negative labels. The lower the threshold, the more positive labels are assigned, and vice versa. A line plot is created for the thresholds in ascending order, with recall on the x-axis and precision on the y-axis. The applied thresholds correspond to all distinct scores by the mention-pair scoring model. A no-skill model is represented by a horizontal line with a precision that is the ratio of positive examples in the dataset. A perfect classifier has full precision and recall and, therefore, is represented in the coordinate (1,1) of the precision-recall curve. For example, the precision-recall curve of an imbalanced dataset with 95% negative and 5% positive labels might look like Figure 4.2.

The best threshold is determined by calculating the F_1-score for each combination of the precision and recall scores. The highest F_1-score determines the best threshold.

Another way to identify the best threshold is using the threshold tuning method. In this approach, the F_1-score is calculated directly for various predetermined thresholds. A plot is created with ascending thresholds on the x-axis and F_1-scores on the y-axis. A threshold tuning curve could, for example, look like illustrated in Figure 4.3.

The precision-recall and threshold tuning methods are based on the same calculation. The F_1-score is calculated for several thresholds using the determined precision and recall values for both methods. The difference is that with the precision-recall method, the thresholds depend on the previously determined probabilities of the mention-pair model. In contrast, the thresholds for the threshold tuning method are predefined. Depending on the granularity of the predefined thresholds for the threshold tuning method, both methods should result in the same optimal threshold value as the same underlying calculation is performed.

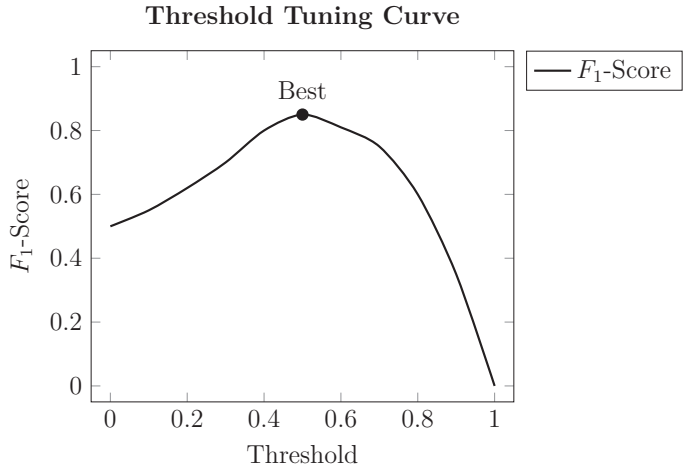


Figure 4.3 Example of a threshold tuning curve. On the x-axis are the thresholds from 0 to 1 for a binary classifier and on the y-axis are the according F_1-scores calculated using precision and recall.

4.1.2 Metrics for Coreference Resolution

Machine learning classification tasks are typically evaluated using precision, recall, and F-scores. Coreference resolution, however, is usually evaluated by the CoNLL F_1 score, which combines three metrics: MUC, B^3, and CEAF. These metrics have been designed to better translate a system's scores into real-world performance.

The input to the CoNLL scorer consists of two files. One is the *key* file containing gold annotated mentions and coreferences, i.e., the perfect mentions and coreferences. Moreover, one is the *response* file containing the predicted mentions and coreferences generated by an entity linking algorithm. The closer the predicted mentions and coreferences are to those in the gold file, the better the rating. To illustrate the different metrics, the example from Pradhan et al. [39] is adapted.

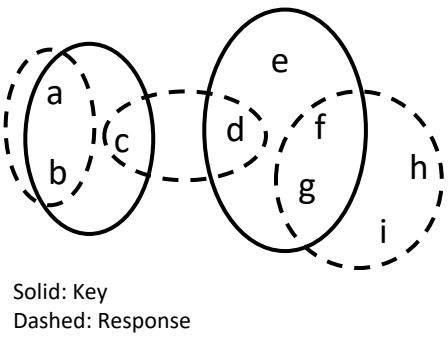


Figure 4.4 Visualization of key and response cluster based on Pradhan et al. [39]

As seen in Figure 4.4 the key (K) contains two cluster with $\{a, b, c\}$ and $\{d, e, f, g\}$ and the

response (R) contains three cluster with mentions $\{a,\ b\}$, $\{c,\ d\}$ and $\{f,\ g,\ h,\ i\}$:

$$K = \overbrace{\{a, b, c\}}^{K_1}\overbrace{\{d, e, f, g\}}^{K_2} \tag{4.5}$$

$$R = \overbrace{\{a, b\}}^{R_1}\overbrace{\{c, d\}}^{R_2}\overbrace{\{f, g, h, i\}}^{R_3} \tag{4.6}$$

Mention e is missing in the response and mentions h and i are incorrect in the response. The following sections use R to denote recall and P for precision.

The *MUC* scoring algorithm is a linked-based evaluation scheme that compares the equivalence classes defined by the gold standard links and the predicted links by the classifier. Hence, it determines how many links need to be added to obtain the correct clustering [54]. However, this metric has two shortcomings: (1) as it is linked-oriented, it disregards singleton mentions, and (2) its link-based F-measure inherently benefits systems producing fewer mentions [29].

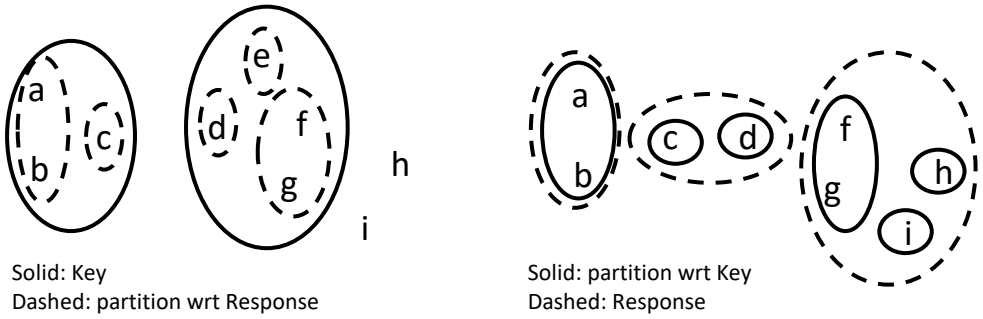


Figure 4.5 Visualization with the partitions of the key and response cluster based on Pradhan et al. [39]

MUC applied to the previously mentioned example yields the partitions for the key and the response, respectively, as shown in Figure 4.5. Based on the partitions, the MUC score is computed by:

$$Recall = \frac{\sum_{i=1}^{N_k}(|K_i| - |p(K_i)|)}{\sum_{i=1}^{N_k}(|K_i| - 1)} \tag{4.7}$$

$$= \frac{(3-2)+(4-3)}{(3-1)+(4-1)} = 0.40 \tag{4.8}$$

$$Precision = \frac{\sum_{i=1}^{N_r}(|R_i| - |p'(R_i)|)}{\sum_{i=1}^{N_r}(|R_i| - 1)} \tag{4.9}$$

$$= \frac{(2-1)+(2-2)+(4-3)}{(2-1)+(2-1)+(4-1)} = 0.40 \tag{4.10}$$

K_i is the i^{th} key entity and $p(K_i)$ is the set of partitions created by intersecting K_i with response entities. R_i is the i^{th} response entity and $p'(R_i)$ is the set of partitions created by intersecting R_i with key entities. Compare Figure 4.5 left and right respectively. N_k is the number of key entities, and N_r is the number of response entities. In this example, the MUC F_1 score is 0.40.

The B^3 scoring algorithm [2] was proposed to address the shortcomings of MUC. This metric is mention-based, i.e., the overall recall and precision are computed based on the recall and precision of the individual mentions. Since the calculations of B^3 are based on mentions, singletons are considered. However, the major shortcoming of this metric is that B^3 assumes that the mentions of the gold standard and the classifier's mentions are identical. Therefore, the algorithm needs to be extended to solve the problem of mismatched mentions [50].

Applied to the example, the B^3 recall and precision scores are calculated as follows:

$$Recall = \frac{\sum_{i=1}^{N_k} \sum_{j=1}^{N_r} \frac{|K_i \cap R_j|^2}{|K_i|}}{\sum_{i=1}^{N_k} |K_i|} \tag{4.11}$$

$$= \frac{1}{7} \times \left(\frac{2^2}{3} + \frac{1^2}{3} + \frac{1^2}{4} + \frac{2^2}{4} \right) = \frac{1}{7} \times \frac{35}{12} \approx 0.42 \tag{4.12}$$

$$Precision = \frac{\sum_{i=1}^{N_k} \sum_{j=1}^{N_r} \frac{|K_i \cap R_j|^2}{|R_j|}}{\sum_{i=1}^{N_r} |R_j|} \tag{4.13}$$

$$= \frac{1}{8} \times \left(\frac{2^2}{2} + \frac{1^2}{2} + \frac{1^2}{2} + \frac{2^2}{4} \right) = \frac{1}{8} \times \frac{4}{1} = 0.50 \tag{4.14}$$

Terms with 0 value are omitted. The B^3 F_1 score is 0.46.

The $CEAF$ [29] metric computes the alignment between gold and system entities. It measures the similarity of each mention cluster to determine the value of each possible alignment. The best alignment is then used to calculate CEAF precision, recall, and F-measure.

Similarity between a key entity K_i and a response entity R_j is denoted as $\phi(K_i, R_j)$ and is defined as [29]:

$$\phi(K_i, R_j) = \frac{2 \times |K_i \cap R_j|}{|K_i| + |R_j|} \tag{4.15}$$

Applied to the example, CEAF recall and precision are:

$$Recall = \frac{\phi(K_1, R_1) + \phi_4(K_2, R_3)}{N_k} = \frac{\frac{(2\times2)}{(3+2)} + \frac{(2\times2)}{(4+4)}}{2} = 0.65 \tag{4.16}$$

$$Precision = \frac{\phi(K_1, R_1) + \phi_4(K_2, R_3)}{N_r} = \frac{\frac{(2\times2)}{(3+2)} + \frac{(2\times2)}{(4+4)}}{3} \approx 0.43 \tag{4.17}$$

The CEAF F_1 score is 0.52.

The B^3 and CEAF metrics are criticized for inflating a system's score by overly rewarding singleton mentions [43]. Virtually all coreference resolution systems developed since 2011 adopt the CoNLL metric: the unweighted average of MUC, B^3, and CEAF scores [40].

Therefore, the CoNLL F_1 score for the applied example is:

$$CoNLL\ F_1 = \frac{MUC\ F_1 + B^3\ F_1 + CEAF\ F_1}{3} = \frac{0.40 + 0.46 + 0.52}{3} = 0.46. \tag{4.18}$$

In the scope of this thesis, the macro-averaged F_1 score is used to evaluate the coreference metrics. The macro-F_1 score is calculated by [25]

$$\text{Macro-F}_1 = \frac{1}{Q}\sum_{j=1}^{Q} 2 \times \frac{P_j \times R_j}{P_j + R_j} \tag{4.19}$$

where R_j and P_j are the j-th recall and precision scores, and Q is the number of different classes. The macro-F_1 score will be called F_1-score for the following sections.

4.1.3 Summary of Metrics

The evaluation metrics of machine learning models depend on the type of model trained. A binary classification model is typically evaluated by accuracy, precision, recall, and F_1-score. These metrics are calculated with the help of the true positive, false positive, true negative, and false negative rates. To visualize these rates, the confusion matrix is typically used. If the model is trained on an imbalanced dataset or on a task in which predictions are made with higher uncertainty, it may be useful to tune the threshold to improve the model performance. Approaches to threshold tuning are, e.g., the precision-recall method and the threshold tuning method, which both use the same underlying calculation to determine the best threshold with the help of the best F_1-score.

CR is evaluated by the CoNLL F_1 score, which combines the three metrics MUC, B^3 and CEAF. These metrics have been designed to better translate the scores of a CR system into real-world performance.

The metrics for a binary classification model will be used to evaluate the pairwise-mention scoring models. Moreover, the CoNLL F_1 score will be used to evaluate the resolved coreferences of developed models. .

4.2 NeuralCoref

This section presents and discusses the results (Section 4.2.1) and highlights the limitations (Section 4.2.2) of the NeuralCoref model.

4.2.1 Results

Table 4.1 shows the results after training the NeuralCoref model. The variant without overlap reaches a CoNLL F_1 score of 55.8, and the variant with overlap has a score of 3.4, which is lower than the score of HuggingFace's NeuralCoref of 61.2. Unfortunately, it is not clear why the results for the variant with overlap are so low in comparison with the other variant.

As the scores for the B^3 and CEAF metrics are higher than the MUC metric, it can be assumed that the model outputs many singletons. This observation is in line with the critique of B^3 and CEAF that they inflate the system's score by overly rewarding singleton mentions.

One possible explanation for the performance is that the extracted features and spans of the mentions are not meaningful enough for a good representation. As previously described, only

Funcloc	MUC			B^3			CEAF			CoNLL
Overlap	R	P	F_1	R	P	F_1	R	P	F_1	F_1
no	36.1	25.6	28.8	87.3	67.5	74.0	59.4	74.5	64.5	55.8
yes	0.9	0.5	0.6	5.3	5.9	5.1	4.4	6.5	5.0	3.6

Table 4.1 Results for Neuralcoref after training.

six instead of 18 single mentions features and only three instead of 8 mention-pair features are extracted.

Another explanation could be that the averaged embeddings of the mentions are not a good representation of how they are formed. By averaging whole sentences and giving equal weight to each word, keywords lose meaning. A sentence consists, in general, of more stop words than keywords. Therefore, the averaging portion of stop words weighs more than the portion of the keywords. Thus, mention embeddings become more homogeneous due to the higher proportion of stop words in sentences. The stop words could be excluded from the embeddings to overcome this issue.

To assign each word a specific weight the *term frequency-inverse document frequency* (TF-IDF) measure could be used [58]. TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. Typically, the TF-IDF weight is composed of two terms. The first computes the normalized term frequency (TF), i.e., the number of times a word appears in a document, divided by the total number of words in that document. The second term is the inverse document frequency (IDF), computed as the logarithm of the number of documents in the corpus divided by the number of documents where the specific term appears. The terms and the overall weight are calculated by:

$$\mathrm{TF}(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \tag{4.20}$$

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}}\right) \tag{4.21}$$

$$\text{TF-IDF}(t) = \mathrm{TF}(t) \times \mathrm{IDF}(t) \tag{4.22}$$

Overall, the current word embedding representation with a few accompanying features is insufficient to find solutions to problems.

Another consideration is that the model can learn from too few positive instances. There are only approximately 3% positive instances in the datasets. This distribution is because each mention has too many antecedents, leading to a highly unbalanced dataset.

Due to the simple way of creating the shift ids for the Neuralcoref model, pairs of mentions will have been separated. As a result, mentions are assigned to shifts in which they do not belong to any other cluster. Consequently, for each shift a mention has been assigned to, $N - 1$, additional negative instances are created, where N is the number of mentions within a shift. Therefore, a more sophisticated method for generating the shift ids for the Neuralcoref model needs to be developed.

4.2.2 Limitations

The NeuralCoref model is trained for one setup only (on both dataset variants, with and without overlap of the functional location codes). Therefore, no statement can be made about how performance would be affected if the context of mentions (e.g., only one entry item before and after a mention as context instead of two) or the time frame is changed (more or fewer hours than the current 72 hours).

Furthermore, a statement about the model's performance cannot be made for the variant with overlap since an implementation error is expected at an undetermined point.

4.3 CDLM

This section first explains additional implemented baselines, which are modifications of the CDLM model (Section 4.3.1), presents and discusses the results (Section 4.3.2) and finally highlights the limitations of this model for the research task (Section 4.3.3).

4.3.1 Baselines

In addition to the pairwise-scoring model of the PR-CDLM, two further model modifications are implemented to be used as additional baselines. The motivation is to check whether simpler pairwise-scoring models are sufficient to solve the task of linking solutions to problems. An overview of the model differences is shown in Table 4.2.

Model internal name	Model head	Uses mentions' context	Mention pairwise vector
NSP	One dense layer	no	no
NSP+CDLM	Multilayer	no	no
PR-CDLM	Multilayer	yes	yes

Table 4.2 Variants of the mention-pair scoring model. Additional models *NSP* and *NSP+CDLM* are used as additional baselines and to check whether simpler models are sufficient to solve the task of linking solutions to problems.

The modifications only affect the scoring of the mention pairs. The datasets used for training and testing are still the same, i.e., the number of positive and negative mention-pairs is distributed equally to 50% each. Also, three different thresholds are tested for the classification to classify if mention-pairs are coreferring or not during training. The thresholds are the default threshold of 0.5 and two additional thresholds, determined with the precision-recall method and the threshold tuning method. Agglomerative clustering is conducted with the same default thresholds plus the best determined threshold of the classification model. Finally, dependent on the best scoring model, evaluated with the CoNLL F_1-score, the best performing model is tested on the test set to determine the final scoring. Both modifications are built upon the SBERT model `all-MiniLM-L12-v2`.

The two additional mention-scoring models are as follows. Figure 4.6 shows the architecture of both of the mention-scoring models. The first *NSP* model is a simple implementation where,

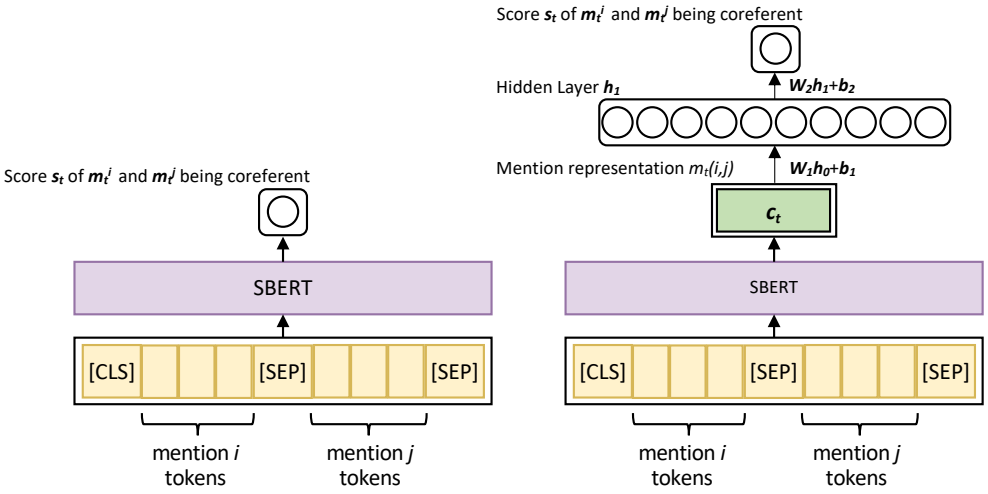


Figure 4.6 The architectures of the additional mention-scoring models. Both models take as input the encoded mention pair along with the BERT tokens `[CLS]` and `[SEP]`. On the left side is the *NSP* model where only one dense layer determines the likelihood of coreference of two mentions. On the right side is the *NSP-CDLM* model, where $\boldsymbol{c}_t$ is the representation vector of the `[CLS]` token and is also the mention representation $m_t(i,j)$. The score $s_t(i,j)$ is produced with one hidden layer of size 384 and a single fully connected layer of size one.

for the SBERT Transformer, the model architecture pretrained for classification is retrieved. The model is retrieved with the *AutoModel* class of HuggingFace. The mention representation consists only of the tokenized mentions i and j, meaning the is no additional context and no pairwise mention features. The model head consists of only one dense layer, which outputs the likelihood that mentions i and j are coreferent. This modification is tested to compare to what extent a basic model architecture can solve coreferences on the level of sentences.

The second *NSP+CDLM* model combines the simple NSP model with the top layer of the CDLM model. The mention-pair is concatenated and encoded using the model along with the `[CLS]` token at the beginning of the sequence. There are also no additional contexts or pairwise mention features for this model. The mention representation consists of the contextualized representation vector of the `[CLS]` token and is the input to the mention-scoring model. The neural network has one fully connected hidden layer $\boldsymbol{h}_1$ of size 384. The output is a score $\boldsymbol{s}_t(i,j)$ of the likelihood that a mention pair is coreferring. The score is produced by applying a single fully connected layer of size one to the output neuron $\boldsymbol{s}_t$.

Figure 4.7 illustrates two different ways two in which the context of the mentions is created for the PR-CDLM model. The first way, the *simple* context, corresponds to the classical approach of wrapping the mentions with context. The two neighboring entries in the log files are used for each mention as context. No distinction is made whether the context is an information, problem, or solution entry item.

The second way, the *advanced* context, distinguishes between information entry items and problem and solution entry items (which are the mentions). Only the information entry items are used to create the context of a mention. Suppose there is a problem or solution description within the scope of two entry items before or after the current mention. In that case, these

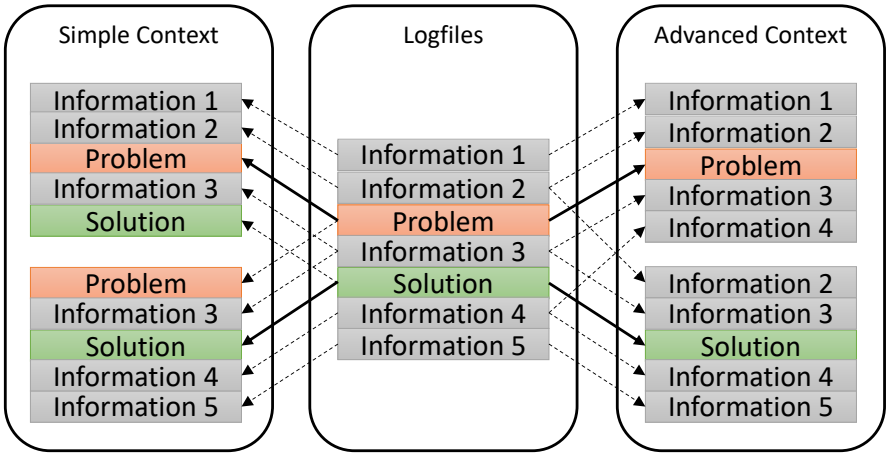


Figure 4.7 Extraction of the *simple* and *advanced* context for the CDLM model.

descriptions are skipped, and the next entry item (if it is an information item) is used as the context for the mention. The motivation for the advanced context is the assumption that the surrounding information entry items can better capture the industrial environment and thus contribute more to linking the problem-solution pairs.

Model	Funcloc Overlap	Context
NSP	no	-
	yes	-
NSP+CDLM	no	-
	yes	-
PR-CDLM	no	advanced
	yes	advanced
	no	simple
	yes	simple

Table 4.3 Listing of all dataset and context modifications of the CDLM model.

Table 4.3 lists all dataset and context modifications of the CDLM model. A total of eight different configurations are tested, including three different mention-scoring models. The NSP and NSP+CDLM mention-scoring models are trained with the filtered and unfiltered variants of the dataset and without additional context. The complete CDLM model is also trained for two different types of context, the simple context, and the extended context, in addition to the two dataset variants.

4.3.2 Results

Table 4.4 shows the results after training all modifications of the CDLM model. Besides the model modifications, dataset variants, context type, and results, the table also lists the best thresholds and linkage types determined for agglomerative clustering.

Model	Funcloc Overlap	Context	Thresh.	Linkage	MUC			B^3			CEAF			CoNLL
					R	P	F_1	R	P	F_1	R	P	F_1	F_1
NSP	no	-	0.3	complete	31.2	37.4	32.6	84.8	91.7	86.6	84.9	78.7	**80.5**	66.6
	yes	-	0.7	complete	51.5	58.0	52.2	79.5	88.2	81.0	78.6	71.4	72.6	68.6
NSP+CDLM	no	-	0.3	complete	31.5	39.5	33.5	79.7	93.0	84.0	84.4	73.5	77.1	64.9
	yes	-	0.7	complete	51.4	62.0	53.7	74.5	90.1	78.7	78.8	66.4	69.6	67.3
PR-CDLM	no	advanced	0.5	complete	36.2	47.4	39.3	83.2	95.0	86.5	85.7	76.3	79.1	68.3
	yes	advanced	0.6	complete	60.2	**72.0**	**63.7**	82.4	95.5	86.3	**86.0**	75.5	78.7	76.2
	no	simple	0.5	average	37.0	51.5	40.6	78.0	**96.4**	83.3	84.3	70.6	74.7	66.3
	yes	simple	0.5	complete	**60.7**	68.7	62.6	**85.2**	93.5	**87.1**	85.5	**78.8**	80.4	**76.7**

Table 4.4 Test results of all modifications of the CDLM model.

The lowest scoring modification is the NSP+CDLM model, trained on the dataset without overlap, with a CoNLL F_1 score of 64.9. The highest scoring modification is the complete PR-CDLM model, trained on the variant with overlap of the dataset and the simple context of the mentions, with a CoNLL F_1 score of 76.7. Thus, the difference between the worst and the best model is 11.8.

All modifications perform better on the dataset variant with overlap than on the variant without overlap. For the NSP model and the NSP+CDLM model, the CoNLL F_1 score differences for the variant without and with overlap are 2.0 and 2.4, respectively. For the PR-CDLM model, the differences for the datasets are 7.9 for the advanced context and 10.4 for the simple context.

Looking at the MUC scores of all model modifications without overlap, the F_1 scores are pretty low, ranging from 32.6 for the NSP model to 40.6 for the PR-CDLM model with simple context. In contrast, the B^3 and CEAF scores are pretty high for the variant without overlap. The uneven distribution suggests that many singletons are produced for the variant without overlap. This can be assumed because B^3 and CEAF inflate the system's scoring when there are a large number of singletons.

The MUC scores for the variants with overlap are significantly higher than those for the variant without overlap. In contrast, the B^3 and CEAF scores for the variant with overlap are slightly lower than those without overlap. So the models trained on the variant with overlap probably produce fewer singletons and more correct coreference chains.

The NSP and NSP+CDLM models have similar overall performance across all metrics (depending on the variants with and without overlap). The variant without overlap of the full PR-CDLM is also in the range of the two previously mentioned models. The PR-CDLM model benefits most from the unfiltered dataset, as this variant has the best performance, regardless of the type of context.

When analyzing the results of agglomerative clustering, it can be seen that the best linkage method is complete linkage. Complete linkage produced the best-performing cluster for all modifications, except for the PR-CDLM model for the variant without overlap and the simple context, for which average clustering produced the best results. The best thresholds for the

NSP and NSP+CDLM model are for the variant without overlap, the lowest tested threshold of 0.3, and for the variant with overlap, the highest tested threshold of 0.7. For the PR-CDLM model, the best thresholds are 0.5 for all modifications, except the variant with overlap with advanced context, for which a threshold of 0.6 yields the best results. For no modification, the best-performing threshold of the mention-pair scoring model is the best-performing threshold for agglomerative clustering.

The main findings of the training with different model modifications and dataset variants are as follows. First, simpler mention-pair scoring models cannot solve the coreference task at the level of sentences and paragraphs in the current setup. The results show that the NSP and NSP+CDLM models perform worse on average than the full PR-CDLM model.

Second, filtering out cases where a problem-solution pair is entered under the same entry item with the same functional location code does not lead to an improvement. One reason for this, however, may be that the dataset without overlap variants contains only about half as many documents, mentions, clusters, and instances. Thus, the advantage of the variant with overlap is that more training data is available. In order to be able to draw a definite conclusion about the effect of filtering, the size of the dataset with overlap must be limited to that of the dataset without overlap.

Third, the best linkage method for agglomerative clustering is complete linkage. Except for one model modification, the best results were obtained with this linkage method. Complete linkage minimizes the maximum distance between observations of cluster pairs.

Fourth, the best-determined threshold for the mention-scoring model is not also the best threshold for the agglomerative clustering task. Worth mentioning is that the best threshold values determined for the NSP and NSP+CDLM models are the minimum and maximum values tested. For the variant without overlap, the minimum threshold of 0.3 is determined. And for the variant with overlap, the maximum value of 0.7. Due to this fact, it is not sure that the best threshold for agglomerative clustering is in the range of 0.3 to 0.7. To conclude, a more comprehensive range of thresholds must be tested for the NSP and NSP+CDLM models.

The PR-CDLM model's best thresholds for agglomerative clustering are 0.5 (three times) and 0.6 (once). These thresholds ensure that for this model, the best threshold for the agglomerative clustering task is in the tested range.

	MUC			B^3			CEAF			CoNLL
	R	P	F_1	R	P	F_1	R	P	F_1	F_1
Barhom et al. [3]*	81.0	80.8	80.9	66.8	75.5	70.9	62.5	62.8	62.7	71.5^+
Cattan et al. [8]*	85.7	81.7	83.6	70.7	74.8	72.7	59.3	67.4	63.1	73.1^+
Allaway et al. [1]*	83.9	84.7	84.3	74.5	70.5	72.4	70.0	68.1	69.2	75.3^+
Caciularu et al. [7]*	**88.1**	**91.8**	**89.9**	82.5	81.7	82.1	81.2	72.9	76.8^+	**82.9**$^+$
PR-CDLM	60.7	68.7	62.6	**85.2**	**93.5**	**87.1**	**85.5**	**78.8**	**80.4**	76.7^-

Table 4.5 Best performing model in comparison with the current state-of-the-art for CDCR. Models with (*) are tested on the ECB+ corpus. Results are including($^+$)/excluding($^-$) singletons in the evaluation, indicating that the PR-CDLM might perform similar to the state-of-the-art as singletons inflate the results [9].

Table 4.5 compares the best performing implemented model with the current-state-of-the art

models for CDCR. However, this comparison can only be evaluated to a limited extent. The four state-of-the-art models were trained and evaluated on the ECB+ corpus. The ECB+ corpus was developed for the (CD)CR task to create a possibility to compare different CR models. Nevertheless, the comparison is made here to get an approximate assessment of the performance of the PR-CDLM model.

According to the results, PR-CDLM performs on average as well as the other CDCR models. These models were trained and evaluated on a simpler dataset and on the setup with singletons. Evaluation on the dataset with singletons has proven to inflate the results [9]. For Barhom et al. [7], reevaluation on the setup without singletons yielded -4pp, so it can be assumed that this pattern affects all other models. To sum up, on the setup without singletons and with a more complex dataset, domain-specific PR-CDLM could ve considered competitive to the other CDCR models of a general domain.

4.3.3 Limitations

Currently, the PR-CDLM model is only trained as a within-document CR model. The datasets have to be created differently to use the cross-document component of the original CDLM. A topic must not simply be equal to a shift and thus include only one document (as one shift equals one document). Instead, a topic must be defined as a period of, e.g., one to three days. Moreover, each shift in the period (one shift equals eight hours) equals one document. Thus, a topic would comprise three to nine documents.

Furthermore, only one set of mention-pair features is tested. A statement about the effect of different feature settings cannot be made.

4.4 Results

Table 4.6 shows all results of each modification of the Neuralcoref model and the CDLM model.

	Funcloc				MUC			B^3			CEAF			CoNLL
Model	Overlap	Context	Thresh.	Linkage	R	P	F_1	R	P	F_1	R	P	F_1	F_1
Neuralcoref	no	-	-	-	36.1	25.6	28.8	**87.3**	67.5	74.0	59.4	74.5	64.5	55.8
	yes	-	-	-	0.9	0.5	0.6	5.3	5.9	5.1	4.4	6.5	5.0	3.6
NSP	no	-	0.3	complete	31.2	37.4	32.6	84.8	91.7	86.6	84.9	78.7	**80.5**	66.6
	yes	-	0.7	complete	51.5	58.0	52.2	79.5	88.2	81.0	78.6	71.4	72.6	68.6
NSP+CDLM	no	-	0.3	complete	31.5	39.5	33.5	79.7	93.0	84.0	84.4	73.5	77.1	64.9
	yes	-	0.7	complete	51.4	62.0	53.7	74.5	90.1	78.7	78.8	66.4	69.6	67.3
PR-CDLM	no	advanced	0.5	complete	36.2	47.4	39.3	83.2	95.0	86.5	**85.7**	76.3	79.1	68.3
	yes	advanced	0.6	complete	60.2	**72.0**	**63.7**	82.4	95.5	86.3	86.0	75.5	78.7	76.2
	no	simple	0.5	average	37.0	51.5	40.6	78.0	**96.4**	83.3	84.3	70.6	74.7	66.3
	yes	simple	0.5	complete	**60.7**	68.7	62.6	85.2	93.5	**87.1**	85.5	**78.8**	80.4	**76.7**

Table 4.6 Results on all variants of the NeuralCoref and CDLM models for coreference resolution.

Overall, the transformer-based CDLM modifications score higher than the feature-based NeuralCoref model. Neuralcoref's best CoNLL F_1 value of 55.8 is 9.1 points lower than the worst CDLM modification, which achieves a CoNLL F_1 value of 64.9. Except for the NeuralCoref model, the results for the models trained with the variant with overlap are always higher. However, it

should be noted that the implemented NeuralCoref model trained on the variant with overlap may contain an error.

All modifications have the issue that they create a large number of singletons. This can be assumed because the B^3 and CEAF values are relatively high compared to the MUC values. Therefore, the scores of the systems are artificially inflated (cf. [45, 62]).

4.5 Summary

Two distinct CR pipelines are implemented to resolve coreferences on the level of sentences and paragraphs. The first pipeline is based on the feature-driven NeuralCoref model, developed for within-document CR. The second model is a transformer-based model developed for cross-document CR. The pipelines consist of each a mention-scoring stage and a resolution stage. In the mention-scoring stage, the likelihood of a mention-pair being coreferring or not is determined. For the NeuralCoref model, only the individual scores for each pair of mentions are assigned. For the CDLM models, it is already decided in this stage if two mentions are coreferring or not. The decision is dependent on an identified best-performing threshold. The performance of this stage is for the CDLM models evaluated using the precision, recall, and F_1 metrics.

The mentions are merged into clusters during the resolution stage depending on their previously assigned scores. As the NeuralCoref model is a mention-ranking model, the highest-scoring antecedent is merged with the current mention. This way, the clusters are built incrementally. For the CDLM model, agglomerative clustering is used to form the cluster. Agglomerative clustering is a type of hierarchical clustering. Several distance thresholds and linkage methods are tested with the result that complete linkage combined with a threshold of 0.5 is the best setting for the agglomerative clustering for this task. The coreference resolution is evaluated using the CoNLL F_1 score, a combination of the three metrics MUC, B^3, and CEAF.

Overall, the transformer-based CDLM modifications achieve higher scores than the feature-based NeuralCoref model. Nevertheless, both models have the issue that they produce a high number of singletons which inflate the results. Hence, the final results may look better than they are.

5 Future Work

The future work chapter shows possible improvements and extended approaches that can be implemented for the models. The chapter begins with improvements that apply to both model variants. Then it moves on to specific problems and improvements for the NeuralCoref model (Section 5.1) and on to the PR-CDLM model (Section 5.2). Finally, it presents an end-to-end prototype (Section 5.3).

As previously mentioned, both models have the problem of generating too many singletons and inflating the evaluation results. In order to not artificially inflate the results with a large number of singletons, the singletons should be excluded from the evaluation to obtain a more realistic picture of the model's performance. Alternatively, two parallel evaluations can be conducted, one with and one without the singletons, to assess the impact of the singletons.

Also relevant to both models is that a comprehensive ablation study needs to be conducted to determine the usefulness of the currently defined setups and pairwise features. It needs to be assessed whether the context of two entry items before and after a mention reflects the context well or whether a different number of entry items leads to better performance. It also needs to be examined whether the 72-hour time frame is reasonable or whether a different limit would yield better results. It must also be verified whether it makes sense to randomly generate a time difference between 1 and 72 hours if it exceeds 72 hours. In addition, it must be checked whether the log files contain other valuable information besides the time difference and the overlap of the functional location code that can be used.

5.1 NeuralCoref Model

As discussed in section 4.2, the implemented NeuralCoref model has some issues and possible improvements. First, it must be investigated why the model does not work with the dataset variant with the overlapping functional location codes, as it only reached a CoNLL F_1 score of about 3.6. Possible errors are that either the dataset was not preprocessed correctly, the mentions and mention-pair features were extracted incorrectly, or there is a bug in the code.

As a further improvement, the shifts for the NeuralCoref model need to be generated more sophisticatedly. In order to have a uniform definition, the shifts should be analogous to those of the PR-CDLM model. An approach must be found that all coreferent mentions occur within a shift for within-document CR, or all coreferent mentions are part of one topic for cross-document CR. At the very least, it must be verified whether the defined time frame of 72 hours for a shift is reasonable or whether a longer or shorter period provides improved results.

As previously discussed, the NeuralCoref model is trained on an imbalanced dataset. The distribution is 3% positive instances to 97% negative instances. With so few positive instances, the model does not know where to look and what it has to learn. With such a distribution, the task would need to be defined as an anomaly detection task. The distribution of the dataset needs to be balanced to be able to train a CR model. There are several ways to create a more balanced dataset. First, the number of antecedents per mention can be limited when creating the mention-pair table. Right now, a mention-pair is created for every antecedent of a mention within the scope of a document. This means that for a document with 100 mentions, 100 pairs

of mentions are created for the 100th mention (99 for each antecedent + one for NA, i.e., the mention has no antecedent). If this 100th mention has only one coreferring mention in the document, this yields 99 additional negative instances and only one positive instance. By limiting the mention-pair distance to, e.g., five, there would only be four negative instances and one positive instance (if the coreferring antecedent is within the defined distance). The positive instances can be created regardless of the defined distance to avoid losing any positive instances. This will lead to a more balanced dataset the model can be trained on. Second, after creating all mention-pairs, the number of negative instances can be analogous to the PR-CDLM instances preparation, limited to the number of positive instances. This way, there would be a distribution of 50% each. After creating such a dataset, the model must be re-trained, and the results must be compared with the previously obtained results.

A further improvement discussed earlier is to assign weights to the word embeddings using the TF-IDF measure. The assigned TF-IDF weights eliminate uninformative stop words when creating the average word embeddings of the mentions and context text. Thereby, more meaningful vector representations can be created, better representing the texts' content. Due to a better representation, it can be assumed that the task of linking problems and solutions can be solved more effectively.

5.2 CDLM Model

Specific to the PR-CDLM model, the definition of documents and topics must be addressed as discussed in section 4.3. Currently, each shift is defined as a topic and a document. Since the PR-CDLM is trained on topics, in the current setup, this results in the training only being conducted on single documents. To address this shortcoming, a topic should not be defined as a shift but as a time frame of, e.g., three days. Consequently, each shift belongs to a topic within a specific time frame. A topic would, therefore, consist not only of one shift (one document) but of up to nine shifts (documents) for three days. A shift lasts, on average, eight hours, so three shifts per day and nine shifts within three days. With such a setup, the cross-document component of the CDLM can be properly utilized. This can also result in additional positive instances, as in such a setup, the cases are considered when related problems and solutions appear in adjacent shifts.

The CDLM is a general model, and other similar Transformers can be used. Therefore, another fundamental research question is whether other pre-trained language models are better suited to solve the task. Since the log files are created in German, Transformers like German BERT or other pretrained Transformer in the German language[1] should be tested as underlying models of the CDLM. So the question here is which architecture and training task is most suitable to build the CDLM on top to link problems and solutions in the log files.

[1] Overview of HuggingFace pretrained Transformers in German: https://bit.ly/3D6sJNy

5.3 End-to-End Prototype

This thesis contributes to an end-to-end pipeline that identifies problems and solutions (mentions) in a log file and then uses CR to link related problems and solutions. The objective is to have a revised log file in which all problems and solutions are tagged, and it is indicated which problems and solutions are related to each other. Figure 5.1 illustrates a high-level overview of the pipeline.

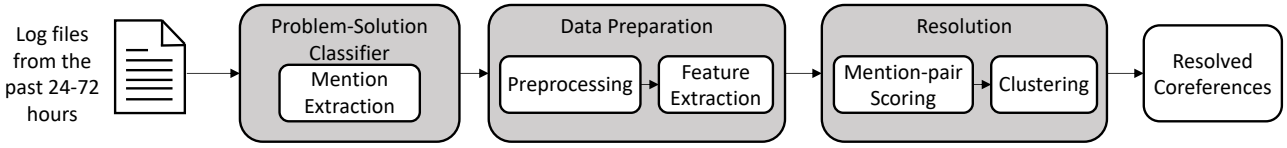


Figure 5.1 End-to-end problem-solution linking pipeline. The input to the pipeline are the log files from the past 24-72 hours. The events are classified from a Problem-Solution classifier to the classes: problem, solution, information. Then, during the data preparation phase the data is preprocessed and features are extracted from the mentions, i.e., the events that previously were classified as problem or solution. Next, mention-pair scores are calculated, which are used to cluster the mentions to coreference chains. Finally, a file with the resolved coreferences is saved.

The input to the pipeline are the log files from the past 24 to 72 hours. With the help of a problem-solution classifier, the problems and solutions in these log files are identified. In a CR pipeline, this is the equivalent of the mention extraction step. Therefore, instead of gold annotated log files, actually extracted mentions are used for this pipeline. These identified mentions are the input to the model developed in this thesis. As the current best model is a PR-CDLM implementation, the data preparation is conducted as described in section 3.4.1, and the resolution is as described in Section 3.4.3. The output of the pipeline are the resolved coreferences of the past 24 to 72 hours.

This pipeline helps to improve the data quality of the log files. Thus, problems can be specifically searched for, and solutions to them can be identified.

6 Conclusion

This master thesis aimed to determine how coreference resolution can be used to improve the data quality in the logs of daily operations, i.e., restore missing links between problem and solution records. Log files from daily operations in the processing industry were used as the database in the German language. These log files contain various entry items, such as problem and solution descriptions, and other entry items, such as various information descriptions.

The thesis contributes to a larger project of the solution recommender system, i.e., given a problem description, a system suggests solutions based on previously solved similar problems. These solution suggestions can be learned from the historical data of the log files. To learn correct solution suggestions presupposes that the data quality of the log files is high enough for such a model to learn the correct solution suggestions. The issue, however, is that such log files contain some incorrect entry items. Such issues are, for example, that related problems and solutions are not connected. As a result, there are fewer instances to learn which problems and solutions need to be connected. Another problem is that the condition of the entry is not clearly defined. Therefore, the entry items vary in complexity and can range from one word to several sentences.

Related problems and solutions must be linked to increasing data quality in the log files. This task is defined as a coreference resolution task in this thesis. Coreference resolution is a well-studied research topic in the field of NLP, where attempts are made to link anaphoric expressions with their antecedents. In this thesis's scope, the log files' problems and solutions are defined as mentions. The information entry items form the context of the mentions. A shift is defined as a document. The mention extraction was not part of this thesis. Therefore, it was attempted to only resolve the coreferences in the context of a classical coreference pipeline.

The thesis proved that the CR problem definition is applicable to the current domain problem by performing the following steps:

1. The task of problem-solution linking was defined as a coreference resolution task which involved mapping the general definitions of general CR and CDCR to the domain-specific applied task. As the log files contain problem and solution descriptions in the form of sentences, coreference resolution was applied on the level of sentences and paragraphs. Next, the data was transformed and preprocessed according to the derived definitions of data structures used in (CD)CR.

2. Two diverse models were evaluated, selected, and adjusted: 1) the feature-driven neural network model for CR proposed by Clark and Manning ([13, 14]), which has been used as state-of-the-art for a long time, and 2) the transformer-based neural model for CDCR by Caciularu et al. [7], which is currently state-of-the-art.

3. The CDLM model was extended by 1) using a German transformer model, 2) adding a domain-specific feature vector to the mentions pairwise representation, and 3) a thorough evaluation of parameters for agglomerative clustering. In addition, as an ablation study, two simpler model modifications were implemented to test whether simpler models can solve the coreferences.

4. The evaluation has shown that the transformer-based PR-CDLM model with additional features performs best in the current domain (CoNLL $F_1 = 76.7$) and yields roughly the same score as the current state-of-the-art models (which achieve an average CoNLL F_1 score of about 75.75). PR-CDLM demonstrates a significant advantage over pure transformer-based models, without additional features and added context (best CoNLL $F_1 = 68.6$), and over pure feature-driven models (CoNLL $F_1 = 55.8$).

Bibliography

[1] Allaway, E.; Wang, S., and Ballesteros, M., "Sequential cross-document coreference resolution", *CoRR*, vol. abs/2104.08413, 2021.

[2] Bagga, A. and Baldwin, B., "Algorithms for scoring coreference chains", in *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, 1998, pp. 563–566.

[3] Barhom, S.; hwartz, V.; Eirew, A.; Bugert, M.; Reimers, N., and Dagan, I., "Revisiting joint modeling of cross-document entity and event coreference resolution", in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4179–4189.

[4] Beltagy, I.; Peters, M. E., and Cohan, rman, "Longformer: The long-document transformer", *CoRR*, vol. abs/2004.05150, 2020.

[5] Bojanowski, P.; Grave, E.; Joulin, A., and Mikolov, T., *Enriching word vectors with subword information*, 2016.

[6] Bosua, R. and Venkitachalam, K., "Aligning strategies and processes in knowledge management: A framework", *Journal of Knowledge Management*, vol. 17, May 2013.

[7] Caciularu, A.; Cohan, A.; Beltagy, I.; Peters, M. E.; Cattan, A., and Dagan, I., "Cross-document language modeling", *CoRR*, vol. abs/2101.00406, 2021.

[8] Cattan, A.; Eirew, A.; Stanovsky, G.; Joshi, M., and Dagan, I., "Streamlining cross-document coreference resolution: Evaluation and modeling", *ArXiv*, vol. abs/2009.11032, 2020.

[9] ——, "Realistic evaluation principles for cross-document coreference resolution", in *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, Association for Computational Linguistics, Aug. 2021, pp. 143–151.

[10] Chua, A., "The dark side of successful knowledge management initiatives", *Journal of Knowledge Management*, vol. 13, pp. 32–40, Jul. 2009.

[11] Clark, K., "Neural coreference resolution", 2015.

[12] Clark, K.; Khandelwal, U.; Levy, O., and Manning, C. D., "What does BERT look at? an analysis of BERT's attention", in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 276–286.

[13] Clark, K. and Manning, C. D., "Deep reinforcement learning for mention-ranking coreference models", in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, 2016, pp. 2256–2262.

[14] ——, "Improving coreference resolution by learning entity-level distributed representations", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, 2016, pp. 643–653.

[15] Cruz, A. F.; Rocha, G., and Cardoso, H. L., "Coreference resolution: Toward end-to-end and cross-lingual systems", *Information*, vol. 11, p. 74, 2020.

[16] Cybulska, A. and Vossen, P., "Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution", in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 4545–4552.

[17] Devlin, J.; Chang, M.; Lee, K., and Toutanova, K., "BERT: pre-training of deep bidirectional transformers for language understanding", *CoRR*, vol. abs/1810.04805, 2018.

[18] Devlin, J.; Chang, M.-W.; Lee, K., and Toutanova, K., "BERT: Pre-training of deep bidirectional transformers for language understanding", in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[19] Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I., and Salakhutdinov, R., "Improving neural networks by preventing co-adaptation of feature detectors", *CoRR*, vol. abs/1207.0580, 2012.

[20] Hinton, G. and Tielemann, T., "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", COURSERA: Neural Networks for Machine Learning, 2012.

[21] Hochreiter, S. and Schmidhuber, J., "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] Huang, Y. *et al.*, *Anaphora: A cross-linguistic approach*. Oxford University Press on Demand, 2000.

[23] Joshi, M.; Levy, O.; Zettlemoyer, L., and Weld, D., "BERT for coreference resolution: Baselines and analysis", in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5803–5808.

[24] Jurafsky, D. and Martin, J. H., *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009.

[25] Kafrawy, P. E.; Mausad, A., and Esmail, H., "Experimental comparison of methods for multi-label classification in different application domains", *International Journal of Computer Applications*, vol. 114, pp. 1–9, 2016.

[26] Lee, K.; He, L.; Lewis, M., and Zettlemoyer, L., "End-to-end neural coreference resolution", in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 188–197.

[27] Lee, K.; He, L., and ettlemoyer, L., "Higher-order coreference resolution with coarse-to-fine inference", in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 687–692.

[28] Liu YinhanOtt, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L., and Stoyanov, V., *Roberta: A robustly optimized bert pretraining approach*, 2019.

[29] Luo, X., "On coreference resolution performance metrics", in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, Vancouver, British Columbia, Canada: Association for Computational Linguistics, Oct. 2005, pp. 25–32.

[30] Manning, C. "Cos 484: Natural language processing - coreference resolution". (2019), (visited on 09/12/2022).

[31] Marneffe, M.-C. d.; Recasens, M., and Potts, C., "Modeling the lifespan of discourse entities with application to coreference resolution", *Journal of Artificial Intelligence Research*, vol. 52, pp. 445–475, 2015.

[32] Mikolov, T.; Chen, K.; Corrado, G., and Dean, J., "Efficient estimation of word representations in vector space", *CoRR*, vol. abs/1301.3781, 2013.

[33] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G., and Dean, J., "Distributed representations of words and phrases and their compositionality", in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13, Lake Tahoe, Nevada: Curran Associates Inc., 2013, 3111–3119.

[34] Moosavi, N. S. and Strube, M., "Search space pruning: A simple solution for better coreference resolvers", in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1005–1011.

[35] Müllner, D., *Modern hierarchical, agglomerative clustering algorithms*, 2011.

[36] Nair, V. and Hinton, G. E., "Rectified linear units improve restricted boltzmann machines", in *ICML 2010*, 2010, pp. 807–814.

[37] Ng, V., "Supervised noun phrase coreference research: The first fifteen years", in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 1396–1411.

[38] Olubunmi, F., "Knowledge management as an important tool in organisational management: A review of literature", Apr. 2015.

[39] Pradhan, S.; Luo, X.; Recasens, M.; Hovy, E.; Ng, V., and Strube, M., "Scoring coreference partitions of predicted mentions: A reference implementation", in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 30–35.

[40] Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O., and Zhang, Y., "CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes", in *Joint Conference on EMNLP and CoNLL - Shared Task*, Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 1–40.

[41] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W., and Liu, P. J., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", *Journal of Machine Learning Research*, vol. 21, no. 140, p. 1,

[42] Rahman, A. and Ng, V., "Supervised models for coreference resolution", in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, ser. EMNLP '09, Singapore: Association for Computational Linguistics, 2009, 968–977, ISBN: 9781932432626.

[43] Recasens, M. and Hovy, E., "Blanc: Implementing the rand index for coreference evaluation", *Nat. Lang. Eng.*, vol. 17, no. 4, 485–510, 2011, ISSN: 1351-3249.

[44] Recasens, M. and Hovy, E., "A deeper look into features for coreference resolution", in *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution Colloquium on Anaphora Processing and Applications*, ser. DAARC '09, Goa, India: Springer-Verlag, 2009, 29–42, ISBN: 9783642049743.

[45] Recasens, M.; Màrquez, L.; Sapena, E.; Martí, M. A.; Taulé, M.; Hoste, V.; Poesio, M., and Versley, Y., "SemEval-2010 task 1: Coreference resolution in multiple languages", in *Proceedings of the 5th International Workshop on Semantic Evaluation*, Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 1–8.

[46] Reimers, N. and Gurevych, I., "Sentence-bert: Sentence embeddings using siamese bert-networks", *CoRR*, vol. abs/1908.10084, 2019.

[47] Rogers, A.; Kovaleva, O., and Rumshisky, A., "A primer in BERTology: What we know about how BERT works", *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020.

[48] Sapena, E.; Padró, L., and Turmo, J., "A constraint-based hypergraph partitioning approach to coreference resolution", *Computational Linguistics*, vol. 39, no. 4, pp. 847–884, Dec. 2013.

[49] Soon, W. M.; Ng, H. T., and Lim, D. C. Y., "A machine learning approach to coreference resolution of noun phrases", *Computational Linguistics*, vol. 27, no. 4, pp. 521–544, 2001, ISSN: 0891-2017.

[50] Stoyanov, V.; Gilbert, N.; Cardie, C., and Riloff, E., "Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art", in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 656–664.

[51] Teece, D. J.; Pisano, G., and Shuen, A., "Dynamic capabilities and strategic management", *Strategic Management Journal*, vol. 18, no. 7, pp. 509–533, 1997.

[52] Uryupina, O. and Moschitti, A., "A state-of-the-art mention-pair model for coreference resolution", in *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 289–298.

[53] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L., and Polosukhin, I., "Attention is all you need", *CoRR*, vol. abs/1706.03762, 2017.

[54] Vilain, M.; Burger, J.; Aberdeen, J.; Connolly, D., and Hirschman, L., "A model-theoretic coreference scoring scheme", in *Proceedings of the 6th Conference on Message Understanding*, ser. MUC6 '95, Columbia, Maryland: Association for Computational Linguistics, 1995, 45–52.

[55] Wiseman, S.; Rush, A. M., and Shieber, S. M., "Learning global features for coreference resolution", in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 994–1004.

[56] Wiseman, S.; Rush, A. M.; hieber, S., and Weston, J., "Learning anaphoricity and antecedent ranking features for coreference resolution", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1416–1426.

[57] Wolf, T. "State-of-the-art neural coreference resolution for chatbots". (Jul. 7, 2017), [Online]. Available: `https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30` (visited on 09/16/2022).

[58] Wu, H. C.; Luk, R. W. P.; Wong, K.-F., and Kwok, K.-L., "Interpreting tf-idf term weights as making relevance decisions", *ACM Trans. Inf. Syst.*, vol. 26, 13:1–13:37, 2008.

[59] Yang, L.; Zhang, M.; Li, C.; Bendersky, M., and Najork, M., "Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for document matching", *CoRR*, vol. abs/2004.12297, 2020.

[60] Yu, X.; Yin, W., and Roth, D., "Paired representation learning for event and entity coreference", *CoRR*, vol. abs/2010.12808, 2020.

[61] Zeng, Y.; Jin, X.; Guan, S.; Guo, J., and Cheng, X., "Event coreference resolution with their paraphrases and argument-aware embeddings", in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 3084–3094.

[62] Zhekova, S. K. andDesislava, "Singletons and coreference resolution evaluation", in *RANLP*, RANLP 2011 Organising Committee, 2011, pp. 261–267.

[63] Zhou, X.; Pappas, N., and Smith, N. A., "Multilevel text alignment with cross-document attention", *CoRR*, vol. abs/2010.01263, 2020.