Jheronimus Academy of Data Science

Information extraction from homicide-related Dutch texts using BERT

Master's Thesis

Bart Haak

Supervisors:

Dr. D.A. Kolkman
Drs. B.J. Butijn

July 1, 2020

Abstract

An abundance of crime-related news articles gets published every day. Consequently, it is infeasible to quickly extract relevant information from all the texts. Therefore, the Dutch police are planning to deploy a knowledge base that contains all the relevant data concerning homicide cases in the Netherlands. Importantly, to attain accurate information in this homicide knowledge base, the process of information extraction (IE) needs to be optimized. Hence, this thesis focuses on IE on a self-constructed homicide corpus. This corpus allows for the extraction of information through named entity recognition (NER) and semantic role labeling (SRL). Additionally, I create a homicide-specific ontology for both these tasks to aid the extraction of victims, suspects, and homicide locations. Furthermore, this thesis implements the Bidirectional Encoder Representations Transformer (BERT), which has boosted performance on many natural language processing (NLP) tasks such as NER and SRL. Accordingly, I propose two BERT-driven IE systems.

The first system uses NER to identify all tokens (e.g. words, punctuation marks) that belong to relevant named entities. Subsequently, based on the token-level extractions, the system applies a rule-based algorithm (Algorithm 1) to infer the entities at a case level. To maximize the performance of this system, I experiment with eighteen BERT configurations for NER. The most successful configuration uses BERT as features with two Bidirectional Long Short Term Memory (BiLSTM) layers, a Softmax classification layer, and a dropout rate of 0.5. This model achieves a F1-score of 0.847 for NER on the homicide corpus. Furthermore, the entire NER system accurately predicts the case-level entities 87.4% of the time. Hence, these results demonstrate that homicide-related named entities can be extracted relatively successfully.

The second system applies SRL to discover the semantic roles within the homicide texts at a token level. Subsequently, a rule-based algorithm (Algorithm 2) obtains the case-level semantic roles. In contrast to NER, I apply two BERT-based models for SRL. The fine-tuned BERT with a Conditional Random Field (CRF) classification layer achieves the best performance (F1-score of 0.714). By using this model, the predictions of the case-level roles have an accuracy of 69%. The results imply that SRL is not at the same level as NER for now. However, real-world applications of SRL have been mostly unexplored. Hence, the proposed SRL system provides useful insights that future research can investigate more extensively.

In conclusion, the results show that there is potential for BERT-based models that conduct crime-specific NER and SRL tasks. Even though the proposed systems should not be used in practice yet, this thesis presents the first step towards real-world applicable information extraction systems.

Acknowledgements

For the past ten months, I dedicated a large part of my time reading academic literature, discussing ideas, scraping data, implementing machine learning algorithms, and writing my thesis. I am thankful that I have had this opportunity to acquire new knowledge and to develop my skills.

First of all, I would like to thank my first supervisor, Daan Kolkman, who helped me to formulate my research and to make adjustments when necessary. Next, I wish to thank my second supervisor, Bert-Jan Butijn, for guiding me through the technical approach and the later stages of my thesis. Third, I would like to pay special regards to Peter de Kock from Pandora Intelligence. He initiated the project that this thesis is a part of and provided support when necessary. Additionally, I wish to express my gratitude to Susanne Brok and Stan Duijf from the Dutch National Police. Both of them showed interest in my research and let me experience how the police force operates. This motivated me to improve my work and helped me to understand the potential practical implications of the thesis.

Finally, I want to thank my family and friends who supported me throughout the project. In particular, I would like to show my gratitude to fellow student Martin Kirilov, who worked together with the police and Pandora Intelligence as well. He always was available to provide some feedback, discuss ideas, or to just have a laugh with.

Bart Haak
Eindhoven, 2020

Contents

List of Figures

List of Tables

List of Acronyms

ACE	Automatic Content Extraction
Arg0	PropBank Argument 0
Arg1	PropBank Argument 1
Arg2	PropBank Argument 2
Arg3	PropBank Argument 3
Arg4	PropBank Argument 4
ArgM-DIR	PropBank Argument Modifier for Direction
ArgM-LOC	PropBank Argument Modifier for Location
ArgM-NEG	PropBank Argument Modifier for Negation
ArgM-TMP	PropBank Argument Modifier for Time
BERT	Bidirectional Encoder Representations from Transformers
BILOU	Begin, Inner, Last, Outer, Unit (annotation scheme)
BiLSTM	Bidirectional Long-Short Term Memory network
BIO	Begin, Inner, Outer (annotation scheme)
BoW	Bag of Words
CBOW	Continuous Bag of Words
[CLS]	Classification Token
CoNLL	Conference on Natural Language Learning
CRF	Conditional Random Fields
EE	Event Extraction
ELMo	Embeddings from Language Models
ESC12	all the 12 Elementary Scenario Components
FN	False Negatives
FP	False Positives
GloVe	Global Vectors for Word Representations
GPT-2	Generative Pre-trained Transformer 2
HMM	Hidden Markov Models

IDF	Inverse Document Frequency
IE	Information Extraction
lr	learning rate
LSTM	Long-Short Term Memory network
MEANTIME	Multilingual Event ANd TIME (corpus)
MEMM	Maximum Entropy Markov Model
MLM	Masked Language Modeling
MLP	Multi-layer Perceptron
MRQ	Main Research Question
MUC	Message Understanding Conference
NER	Named Entity Recognition
NLTK	Natural Language Toolkit
NLP	Natural Languange Processing
NSP	Next Sentence Prediction
POS	Part-of-Speech
PP	Prepositional Phrase
RE	Relation Extraction
RNN	Recurrent Neural Network
RoBERTa	Robustly Optimized BERT Pre-Training Approach
[SEP]	Separation Token
seq2seq	sequence-to-sequence
SoNaR	Stevin Nederlandstalig Referentie (corpus)
SRL	Semantic Role Labeling
SRQ	Sub Research Question
SVM	Support Vector Machine
TF	Term Frequency
TiMBL	Tilburg Memory Based Learner
TN	True Negatives
TP	True Positives
TwNC	Twente News Corpus

Chapter 1

Introduction

When a crime occurs, the media is eager to report on it. As a consequence, an abundance of crime-related news articles gets published every day. These crime texts can contain information that is valuable to law enforcement agencies. However, due to the large volume, it is almost infeasible to read everything in detail. Moreover, Srinivasa and Thilagam (2019) mention that the manual extraction of crime-related information is prone to errors and requires a lot of time and effort. Hence, the procedure can be improved by extracting the relevant information automatically. According to Ku et al. (2008), the automatic extraction and presentation of the information can help to quickly comprehend the crime case. Besides, Dasgupta et al. (2017) suggest that the storage of this information in a multi-region knowledge base aids cross-region communication of information. Hence, the Dutch police are investigating the creation of such a crime knowledge base. They partnered with Pandora Intelligence, which is an intelligence company that uses data science and analytics for safety and security initiatives. This thesis is conducted in collaboration with both these parties to aid the investigation.

The ultimate goal of the police is to create a homicide knowledge base consisting of both recent and historical homicides. Subsequently, the police can query the knowledge base to discover relevant information about homicide cases. However, before such a knowledge base can be beneficial for law enforcement, the stored information needs to be accurate and relevant. Therefore, the task of information extraction is crucial for knowledge-base creation. Accordingly, in this thesis, I apply automatic information extraction on homicide texts. As a result, I propose two information extraction systems to retrieve the victims, suspects, and the homicide locations (referred to as arenas) from a self-constructed homicide corpus. This corpus constitutes a collection of textual descriptions of homicide cases in the Netherlands. Hence, this thesis provides the initial step towards the desired homicide knowledge base for the Dutch police.

1.1 Related work

The automatic extraction of information in texts is referred to as information extraction (IE), an important task in the field of natural language processing (NLP). According to Russell and Norvig (2010) the goal of information extraction is to acquire knowledge by extracting objects of interest from a text and identifying the relations among them. Hence, information extraction can be applied in a broad spectrum of domains, such as politics (Bamman and Smith, 2015), finance (Costantino et al., 1997) or, as discussed earlier, crime (Arulanandam et al., 2014; Dasgupta et al., 2017). Existing literature presents various tasks that are useful for information extraction. In this thesis, I apply two tasks: named entity recognition (NER) and semantic role labeling (SRL). The NER task extracts the named entities such as person names, organization names. or location names from a text. On the other hand, SRL helps to extract relations in the sentence based on the semantics and the verb-predicate. Basically, as described by Màrquez et al. (2008), SRL determines who did what to whom, where, when, and how. Moreover, the SRL task is not restricted to the extraction of names. Hence, it allows the identification of more generic descriptions (e.g. "the 25-year old man") as well. However, despite this apparent useful information, real-world applications of SRL are rare (Màrquez et al., 2008).

In contrast, NER is one of the most prevalent IE tasks that has had some practical applications. For instance, Arulanandam et al. (2014) use named entity recognition to extract crime locations from web-scraped news articles. Furthermore, according to Singh (2018), NER is an important stepping stone towards more complex tasks such as knowledge base creation. Dasgupta et al. (2017) and Srinivasa and Thilagam (2019) illustrate this in their work. They apply NER, coreference resolution, and relation extraction techniques to find names of victims, perpetrators, dates, etc. As a result, the researchers from both studies are able to construct a crime knowledge base. Nonetheless, the authors imply that the quality of the retrieved information is insufficient, and hence the applicability of the knowledge base is limited. Similarly, Schraagen et al. (2017) apply NER on fraud reports that have been submitted to the Dutch police. The authors acknowledge that their performance is insufficient and that future research should explore other approaches to improve NER on fraud reports as well as other crime-related texts (e.g. legal texts).

Still, NER has successfully been applied in research that is unrelated to crime. Recently, the introduction of pre-trained contextual word representations models such as the Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2018), Embeddings from Language Models (ELMo) (Peters et al., 2018), and Flair (Akbik et al., 2018) improved the state-of-the-art for NER and various other NLP tasks. These pre-trained models are advantageous to alternative models that need to learn contextual representations from scratch since the pre-trained models need less task-specific data to train on (Sun et al., 2019). Especially BERT has had a substantial impact in the NLP-field. Accordingly, BERT proves to be successful on IE tasks like NER (Devlin et al., 2018; Straková et al., 2019), coreference resolution (Joshi et al., 2019) and SRL (Shi and Lin, 2019).

However, prior research applies BERT in many different configurations. For instance, Devlin et al. (2018) propose a fine-tuned and a feature-based version of BERT. The authors show that fine-tuned BERT achieves better results for NER on the English CoNLL-2003 (Conference on Natural Language Learning from 2003) corpus (Tjong Kim Sang and De Meulder, 2003). Interestingly, however, Straková et al. (2019) achieve state-of-the-art performance for English, Spanish, and

Dutch named entity recognition using feature-based BERT. In a similar vein, results form prior research are inconsistent with respect to the classification approach. Devlin et al. (2018) implement a Softmax classification function in their NER model, whereas (Souza et al., 2019) improve their NER results by implementing a Conditional Random Fields (CRF) classification layer instead. Hence, even though BERT has shown to be successful for information extraction, existing literature shows no consensus on the best BERT configuration. Moreover, successful prior works on BERT do not always experiment with hyperparameter settings (Souza et al., 2019), or simply do not report on the impact of hyperparameter tuning (Devlin et al., 2018; Peters et al., 2019).

1.2 Research questions

According to prior research (Dasgupta et al., 2017; Schraagen et al., 2017; Srinivasa and Thilagam, 2019), crime-related information extraction is insufficient for real-world applications at the moment. On the contrary, existing research has shown promising results in the field of IE by using BERT (Devlin et al., 2018; Shi and Lin, 2019; Straková et al., 2019). Therefore, this research applies BERT to perform information extraction on the self-constructed Dutch homicide corpus. Consequently, the main research question (MRQ) that needs to be answered is the following:

MRQ: *To what extent can BERT be leveraged for information extraction on homicide-related Dutch texts?*

The main IE task of this research is named entity recognition. This task has been used to extract crime-related entities (Arulanandam et al., 2014; Dasgupta et al., 2017), and appears in BERT research (Devlin et al., 2018). Therefore, this thesis investigates whether BERT can successfully be implemented for NER on the homicide corpus. Nevertheless, there is no consensus on the optimal BERT configuration, since NER results for BERT vary across studies (Devlin et al., 2018; Souza et al., 2019; Straková et al., 2019). Therefore, the following two sub research questions (SRQ) are essential for this thesis:

SRQ1: *To what extent can BERT be leveraged for* ***named entity recognition*** *on homicide-related Dutch texts?*

SRQ2: *What BERT configuration is favorable with respect to named entity recognition on homicide-related Dutch texts?*

Finally, the application of semantic role labeling is experimented with. Potentially, SRL can discover crime information more successfully than NER. Namely, it is not confined to the extraction of names and it should be able to find semantic relations within the sentence. Moreover, Shi and Lin (2019) show promising results when using BERT for SRL. Therefore, this thesis explores BERT for SRL on the homicide texts. Which leads to the following question:

SRQ3: *To what extent can BERT be leveraged for* ***semantic role labeling*** *on homicide-related Dutch texts?*

The answers to these research questions provide insights concerning the usefulness of different BERT configurations for information extraction on the homicide corpus. This understanding is crucial for the creation of the homicide knowledge base.

1.3 Motivation and contributions

The motivation to conduct the current research and to ask the aforestated research questions is fuelled by gaps in the existing literature.

First of all, existing studies on crime-related information extraction achieve results that are insufficient for real-life applications (Dasgupta et al., 2017; Schraagen et al., 2017; Srinivasa and Thilagam, 2019). For example, Dasgupta et al. (2017) explain that named entity recognition in crime texts is challenging due to various types of name mentions (e.g. abbreviations, nicknames). Hence, there is room for improvement for crime-related NER and information extraction in general. This observation motivates the decision to study the MRQ and SRQ1.

Secondly, existing research shows that BERT-based models are successful for regular NER (Devlin et al., 2018). However, since prior studies apply different configurations of BERT (Devlin et al., 2018; Souza et al., 2019; Straková et al., 2019), this research investigates which BERT configuration works best on the homicide corpus. Furthermore, the lack of consensus motivates the decision to investigate the configurations more extensively by asking SRQ2. Hence, this thesis contributes to the NER literature by comparing BERT performances using configurations involving feature-based and fine-tuned BERT, CRF and Softmax layers, and several hyperparameters.

In addition, the lack of real-world applications of SRL (Màrquez et al., 2008) motivates the decision to apply SRL for a real-life cause. Accordingly, this thesis uses SRL to extract homicide-related information for the Dutch police. Furthermore, there are fewer studies that examine the application of BERT for SRL compared to NER. Nevertheless, Shi and Lin (2019) use BERT to achieve state-of-the-art performances for SRL. Consequently, the implementations of BERT for SRL are just scratching the surface of possibilities. Hence, this motivates the decision to examine the question posed by SRQ3. Consequently, the thesis explores BERT for SRL on the homicide corpus. This contributes to the SRL literature since it is a real-world application for SRL (i.e. crime knowledge base creation).

Finally, there are not many crime corpora available for information extraction. Although crime corpora for NER do exist (Arulanandam et al., 2014; Dasgupta et al., 2017; Schraagen et al., 2017), they are not specifically designed for homicide-related information extraction. Furthermore, the regular entities (e.g. person, organization, location) that NER distinguishes do not convey relevant crime information (e.g. victim name). To solve this, Dasgupta et al. (2017) apply regular NER to obtain the person entities first. In their next step, they use another algorithm to classify victim and perpetrator names (Dasgupta et al., 2017). Similarly, Arulanandam et al. (2014) apply an algorithm succeeding regular NER to distinguish crime locations and non-crime locations. In contrast, this thesis creates a homicide-specific ontology that can be used for the IE tasks of NER and SRL directly. The proposed ontology distinguishes between the classes "victim", "suspect" and "arena" (i.e. homicide location). Consequently, this removes the need for an additional algorithm that translates regular NER or SRL labels into crime-specific labels. In a similar vein, existing literature illustrates NER applications on custom ontologies (Cimiano and Völker, 2005; Fleischman and Hovy, 2002). However, the ontology as proposed in this thesis is novel. Therefore, this thesis contributes to the literature on crime-related information extraction by applying a homicide specific ontology (i.e. victim, suspect, arena) for NER and SRL on a newly constructed homicide corpus.

1.4 Research approach

To answer the research questions I construct two information extraction systems (i.e. a NER system and a SRL system) for the self-constructed homicide corpus. Each system contains an IE module (NER module or SRL module) and a rule-based algorithm (Algorithm 1 or Algorithm 2). The IE modules predict which tokens (e.g. words, punctuation marks) convey relevant information. I experiment with several implementations concerning the IE modules. Especially the NER module is thoroughly experimented with by applying different BERT configurations. These NER experiments help to answer SRQ1 and SRQ2. Furthermore, to answer SRQ3, this thesis explores the application of BERT for semantic role labeling on the homicide corpus as well. However, since the SRL module is exploratory, only a subset of the BERT configurations is used for the SRL experiments.

In addition to the IE tasks, two rule-based algorithms (i.e. Algorithm 1 and Algorithm 2) are implemented succeeding NER and SRL. These rule-based algorithms translate the token-level named entities and semantic roles into case-level entities (Algorithm 1) and case-level roles (Algorithm 2). The case-level entities and roles present the victim, suspect, and homicide location of an entire homicide case. Hence, the evaluation of these case-level results allows me to answer the main research question.

1.5 Outline

Aside from this introduction, the thesis consists of five main chapters. First, in chapter 2 I elucidate the technical concepts and elaborate on related prior work. Subsequently, chapter 3 describes the methodology of the current thesis. In the methodology, the creation of the homicide corpus is explained. Furthermore, it delineates the experiments for the NER and SRL modules and defines Algorithm 1 and Algorithm 2. Then, in chapter 4 the results of the experiments are presented. Additionally, this chapter evaluates the performance of the complete NER and SRL systems based on the results from Algorithm 1 and Algorithm 2. In succession to the results, chapter 5 discusses the findings and reflects on the limitations of this thesis. Furthermore, this chapter proposes possibilities for future research. Finally, chapter 6 concludes the thesis with a summary of the preceding chapters.

Chapter 2

Literature Review

Automatic information extraction from publicly available texts became a popular research practice after the introduction of corpora provided by MUC-6 (Sixth Message Understanding Conference) (Grishman and Sundheim, 1996) and ACE (Automatic Content Extraction) (Doddington et al., 2004). MUC-6 introduced the "named entity", and hence provides the foundation of subsequent research in the field of named entity recognition. Besides, ACE uses its own benchmark for named entity research and adds new IE tasks such as coreference resolution and relation extraction.

Today, information extraction consists of many subtasks such as named entity recognition, coreference resolution, named entity linking, and the identification of relations (Singh, 2018). As explained by many (like Singh (2018) or Bethard and Yadav (2018)), named entity recognition is a crucial component in natural language processing. The task has been applied for more complex NLP tasks such as knowledge base construction, question answering systems, or text summarization (Singh, 2018). Due to the many opportunities that NER enables, the NER task can be valuable in the crime domain as well. For instance, prior research use NER to construct a crime knowledge base (Dasgupta et al., 2017; Srinivasa and Thilagam, 2019).

Typical entities in a NER task constitute a named mention of a person, location or organization (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). Hence, expressions such as "a couple of gang members" or "the woman" are discarded in NER. Therefore, it is essential to implement additional techniques when the extraction of descriptions and references are desirable. One such a relevant technique could be coreference resolution. Coreference resolution identifies links textual references such as pronouns and entities with each other. Hence, it is useful when information is distributed across sentence parts or multiple sentences (Pradhan et al., 2012; Quirk and Poon, 2017). Nonetheless, just applying NER (with coreference resolution) will not provide the relationships between entities. Hence, the subtasks of relation extraction (RE), event extraction (EE) and semantic role labeling are important to consider when constructing an NLP system.

As outlined by Elloumi et al. (2012), relation extraction is the act of obtaining binary relations between entities, whereas event extraction describes the identification of n-ary relations among entities. These relations are useful when constructing a knowledge base. For instance, without RE there will be no distinction between victim and perpetrator. For example, when "person A" kills “person B” the relationship of ”murder” can be determined according to relation extraction techniques. However, relation extraction and event extraction often rely on predefined relations (Hendrickx et al., 2010), and therefore lack flexibility and limit the number of relation types.

Semantic role labeling is a useful NLP task to apply in the field of information extraction as well (Christensen et al., 2010). In contrast to RE or EE, it does not need predefined relations. Instead, it uses a predicate-argument structure, where the arguments are extracted given a particular predicate. Often, the predicate is the main verb in the sentence (Palmer et al., 2005; Schuurman et al., 2010). For instance, given the sentence ”John kills Jack”, SRL identifies that ”John” and ”Jack” are related to the predicate of ”kills”. Moreover, if other information is present that is related to the predicate of ”kills”, SRL can extract this (who, where, when, how ”kills” ?).

This thesis mainly focuses on named entity recognition, since it is a fundamental IE task that has shown to be effective in knowledge base creation (Singh, 2018). Secondly, this thesis explores semantic role labeling due to the successes of SRL in IE research (Christensen et al., 2010, 2011). Furthermore, the question that SRL tries to answer (”Where, when how did it happen?”) is highly related to the desired homicide information.

Nevertheless, before I take a more thorough look at NER and SRL, this chapter describes the theoretical background of more generic NLP research. This theory lays the foundation for how typical NLP systems operate. Hopefully, at the end of this chapter, the reader is familiar of the current status of natural language processing (particularly NER and SRL), understands the concepts that existing research proposes, and can relate achievements of prior work to this thesis.

To start, the main developments concerning neural networks in the NLP field (section 2.1) as well as text representations (section 2.2) are discussed. Subsequently, based on these concepts, the BERT language model is explained (section 2.3). This allows the reader can get familiar with the terms and concepts that are crucial for understanding the underlying processes of the applied models. Next, the traditional information extraction techniques for named entity recognition (section 2.4) and semantic role labeling (section 2.5) are described in detail. This section is followed by a section that delineates the usage of BERT for NER and SRL (section 2.6). Finally, existing research in the field of information extraction in the crime domain is elaborated on (section 2.7).

2.1 Neural networks for NLP

Traditionally, natural language tasks have been dominated by linear models (Goldberg and Hirst, 2017) such as support vector machines (SVMs) or logistic regression. Since 2014, however, neural networks show superior performance. Neural networks started to be successful in parsing tasks (Chen and Manning, 2014; Weiss et al., 2015), sequence labeling (Ling et al., 2015), machine translation (Bahdanau et al., 2014) and question-answering (Iyyer et al., 2015) to name a few. Moreover, recent NLP breakthroughs such as BERT (Devlin et al., 2018) and Generative Pre-trained Transformer 2 (GPT-2) (Radford et al., 2019) are based on neural network frameworks. The former of which is relevant to the current thesis, since BERT has successfully been applied for NER (Devlin et al., 2018) and SRL (Shi and Lin, 2019). However, to understand the mechanics of BERT, the underlying principles of the neural network needs to be explored in more depth. Evidently, an understanding of the mechanics is important since it helps to motivate decisions in this thesis concerning model design and model optimization. Therefore, this section elucidates the relevant concepts regarding neural networks. First, subsection 2.1.1 discusses the typical feed-forward neural networks. Subsequently, the recurrent neural networks are explained (subsection 2.1.2). Finally, I focus on the transformer architecture in subsection 2.1.3.

2.1.1 Feed-forward neural networks

The feed-forward neural networks, also known as multi-layer perceptrons (MLPs), were first introduced by Rosenblatt (1962). The network uses multiple perceptrons (Rosenblatt, 1958) in order to find non-linear patterns in the data. A typical feed-forward neural network is visualized in Figure 2.1. The network in the figure consists of an input layer, two hidden layers, and an output layer. Each node in the hidden layers and the output layer has an activation function that transforms the input values into a non-linear output. Popular activation functions are the Sigmoid function, tanh, or the rectifier function (Goldberg and Hirst, 2017). Before the activation function is applied, however, the input values are multiplied by weights and a bias value is added. Hence, an output of a node in the network can be calculated according to the following formula:

$$output = activation(x_1 w_1 + x_2 w_2 + ... + x_i w_i + bias)$$

Where x represents the input values and w the weights of the connections. During the training process, the weights are computed based on the loss of the model, which reflects the discrepancy between the model predictions and true values. These computations are performed by an optimization algorithm such as stochastic gradient descent, RMSprop (Tieleman and Hinton, 2012) or Adam (Kingma and Ba, 2014). Furthermore, each of these algorithms has a learning rate (lr) that determines the extent to which the weights are updated.

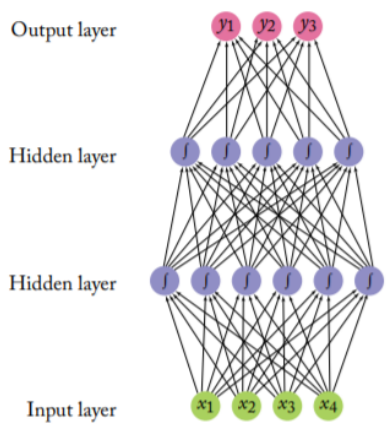


Figure 2.1: Fully connected feed-forward neural network, as visualized by Goldberg and Hirst (2017)

There are several ways in which the structure of the neural network can be optimized for the task at hand. For instance, one can tweak the number of nodes in a layer, the number of hidden layers, the activation functions, or the optimization algorithm. Furthermore, the number of connections can be altered through a technique called dropout. This technique ignores some of the connections between nodes during the training process. Prior research (Hinton et al., 2012) has shown that dropout is useful since a model with dropout generalizes better than a network where all the nodes are connected.

2.1.2 Recurrent neural networks

In the case of natural language processing, the input data is sequential since the words within a sentence depend on each other. However, in regular feed-forward neural networks, it is not possible to account for previous items of the input. Hence, there is a more effective neural network called a recurrent neural network (RNN) for NLP tasks. Recurrent neural networks, proposed by Elman (1990), attempt to capture temporal information through the conservation of prior activation values. Figure 2.2 illustrates the structure of a typical RNN that is used for named entity recognition. The sequential input x is fed into the hidden layer h. All the nodes in h are consecutively connected. Finally, the output layer y labels the input words based on the hidden activation values.

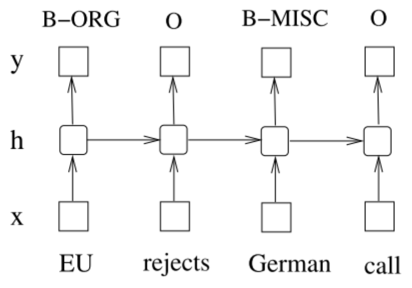


Figure 2.2: RNN modeel as visualized by Huang et al. (2015)

Long-term short memory networks

An adaption of the RNN structure is the Long-Short Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997). In an LSTM architecture, long-range dependencies can be maintained more effectively compared to a vanilla RNN (Hochreiter and Schmidhuber, 1997). The LSTM accomplishes this by using input, forget, and output gates for every hidden node. The gates determine what amount of information from the preceding hidden nodes should be incorporated in the calculation of the new hidden state. This approach differs from a regular RNN, where information from all the previous states is used.

The LSTM model can be transformed into a bidirectional LSTM model (BiLSTM), that uses not only previous input information but future information as well. In this way, the whole context of a word can be taken into account. A simple illustration of a BiLSTM architecture for NER is shown in Figure 2.3. The striped nodes in the hidden layer represent the LSTM cells with input, forget, and output gates.

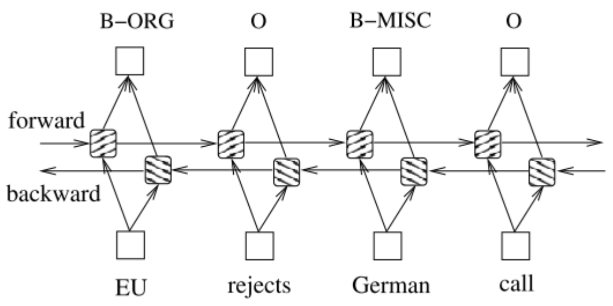


Figure 2.3: BiLSTM model as visualized by Huang et al. (2015)

Attention

Many RNNs are based on an encoder-decoder architecture (Cho et al., 2014b). In such an architecture, one model (e.g. RNN) encodes the input sequence into hidden representations, which subsequently are decoded into the output through a second model (typically another RNN). A simple visualization of this structure is shown in Figure 2.4, with inputs x, a summary of input activations c and outputs y. This structure allows the model to output a sequence of a possibly different length than the input sequence. Hence, this structure has been especially popular in tasks such as neural machine translation (Cho et al., 2014b; Wu et al., 2016) and summarization (Nallapati et al., 2016). Still, it can be implemented for other tasks like automatic speech recognition (Chan et al., 2016) and named entity recognition (Straková et al., 2019) as well.

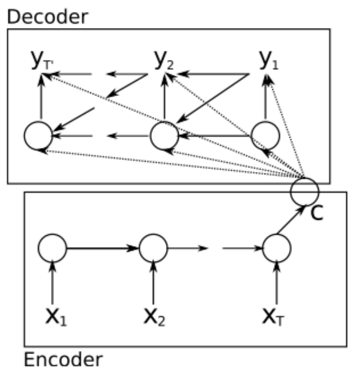
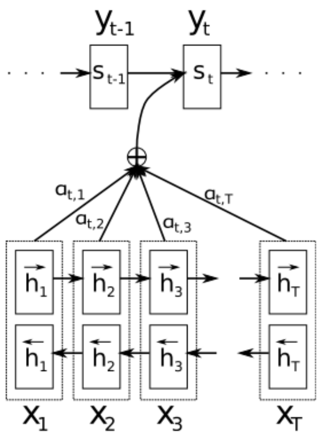


Figure 2.4: Encoder-Decoder RNN architecture, as visualized by Cho et al. (2014b)

Despite the popularity of encoder-decoder architectures, the original architecture has a potential drawback. Namely, as suggested by Bahdanau et al. (2014), the compression of information into one summary (c in Figure 2.4) can be difficult. In line with this notion, Cho et al. (2014a) show that performance of the encoder-decoder architecture as presented by Cho et al. (2014b) degrades for longer sentences. Therefore, to optimize encoder-decoder models, Bahdanau et al. (2014)

introduced an attention mechanism commonly referred to as additive attention. This mechanism helps the decoder to focus on specific parts of the input by obtaining information of each hidden unit separately. More specifically, rather than encoding the input into a single summarization vector, the decoder uses the output from each hidden state. Figure 2.5 provides a graphical representation on how the decoder uses attention outputs of the hidden states (i.e. the context vector) as well as the previous decoder state.



y_{t-1}
y_t
s_{t-1}
s_t
$a_{t,1}$
$a_{t,2}$
$a_{t,3}$
$a_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 2.5: Illustration of the attention mechanism while using a BiLSTM encoder. As visualized by Bahdanau et al. (2014)

According to Bahdanau et al. (2014), the attention mechanism enables the decoder to select relevant and irrelevant information more effectively because individual hidden representations are maintained. In line with this notion, Rocktäschel et al. (2015) qualitatively analyse the recognition of textual entailment of the attention mechanism. The authors apply attention to discover if two sentences (premise and hypothesis) are contradicting, not related or whether the premise entails the hypothesis. The qualitative analysis of the researchers results in convincing visualizations. One of their attention visualizations is shown in Figure 2.6. The figure illustrates that the mechanism is successful at paying attention to the relevant words.

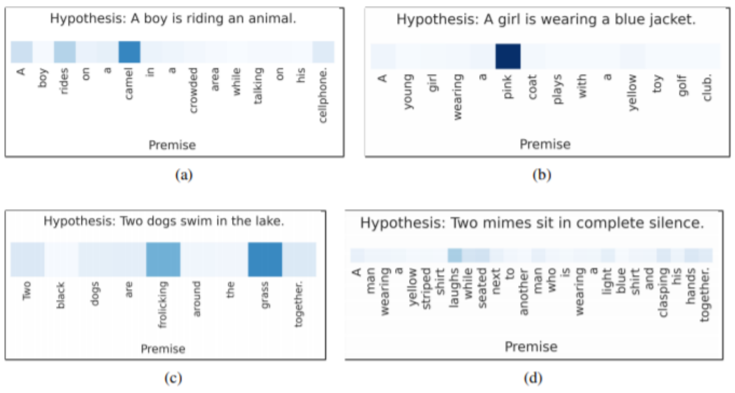


Figure 2.6: Example of how an attention mechanism attends to words in the sentence. As visualized by Rocktäschel et al. (2015)

2.1.3 Transformer networks

Although LSTM-models have had many successes, the transformer model (Vaswani et al., 2017) recently challenged the status quo. The network follows an encoder-decoder architecture, where the decoder uses additive attention as proposed by Bahdanau et al. (2014). Vaswani et al. (2017) refer to this attention mechanism as encoder-decoder attention. Additionally, the authors apply attention within the encoder and decoder as well, which they refer to as multi-head self-attention layers. Furthermore, instead of RNN encoders and decoders, the transformer fully relies on these multi-head self-attention layers in combination with a regular feed-forward neural network. The network structure is illustrated in Figure 2.7.

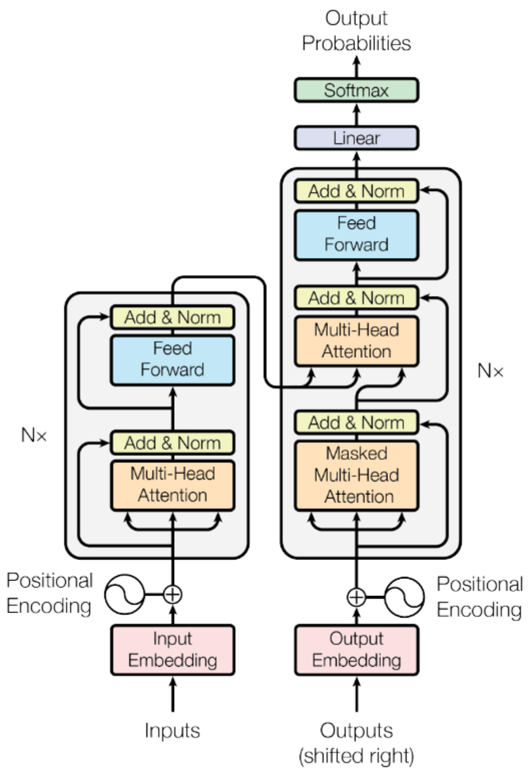


Figure 2.7: Transformer architecture (Vaswani et al., 2017)

In contrast to encoder-decoder attention, self-attention is contained within an encoder or a decoder block. Hence, self-attention can find relevant units (e.g. words) in a sequence for any individual unit of the same input sequence. Before the transformer, self-attention had been successfully implemented for several natural language processing tasks already (Cheng et al., 2016; Lin et al., 2017). Moreover, Cheng et al. (2016) visualize the self-attention mechanism in Figure 2.8. The figure clearly demonstrates how the attention is distributed for each word in the sentence. The fact that the transformer uses multi-head self-attention, allows for multiple self-attentions to run in parallel. This ensures that long-distance dependencies can be captured efficiently (Vaswani et al., 2017). Each attention head can capture a different property in the sentence. For instance, existing research has discovered that the self-attention in transformer models are able to capture named entities (Raganato and Tiedemann, 2018), dependency relations (Vig and Belinkov, 2019) and part-of-speech tags (Raganato and Tiedemann, 2018; Vig and Belinkov, 2019).

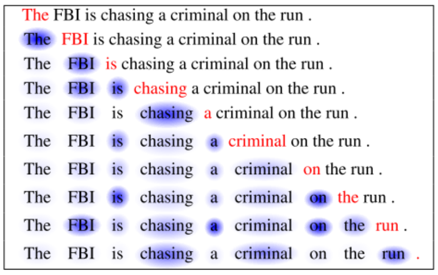


Figure 2.8: Example of self-attention (Cheng et al., 2016)

Some of the most notable applications of the transformer model are GPT-2 proposed by Radford et al. (2019), and BERT from Devlin et al. (2018). The latter of which is applied in this thesis. However, before explaining the mechanisms of BERT in more detail, the next section covers different methods to represent textual data.

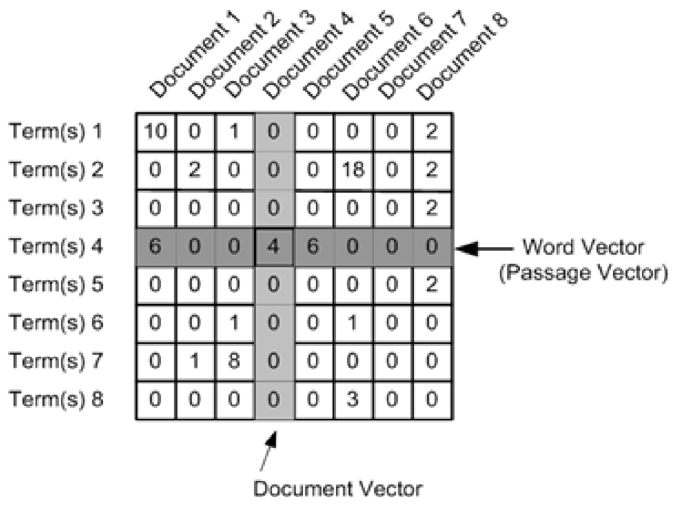
2.2 Text representations

In the past decades, there have been several popular approaches to represent the texts. Frequently, a distinction between the count-based and prediction-based representations is made (Baroni et al., 2014; Levy et al., 2015). For both approaches, the central purpose remains the same; to convert raw textual input into a numeric representation (i.e. vector). Subsequently, the numeric representations can be used by statistical algorithms to perform NLP tasks. Hence, text representation methods are crucial for any NLP system.

In this section, popular count-based representations are described first (subsection 2.2.1). Secondly, the prediction-based methods are elaborated on by discussing static and contextualized representations (subsection 2.2.2).

2.2.1 Count-based representations

Count-based representations initialize the word vectors based on the frequency of word appearances in the text. An intuitive approach to represent texts is the Bag of Words (BoW) method. The BoW approach discards grammar and word order and only considers the word count. The word counts represent the features of a document. Where a document can be anything from a sentence to a multi-paragraph text. Traditionally, this approach was applied to find document similarities based on the document vectors (Baeza-Yates and Ribeiro-Neto, 1999). Nonetheless, BoW can be applied to obtain word representations as well by using the term frequencies. In the latter case, a row vector as in Figure 2.9 initializes the word vector.
To enhance these simple BoW representations, one can use a more advanced method that weighs the terms. A popular method is TF-IDF (Manning et al., 2008), where the term frequency (TF) in a document is multiplied with the inverse document frequency (IDF). The document frequency



	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

Word Vector
(Passage Vector)
Document Vector

Figure 2.9: BoW document and word representations as visualized by Marksberry and Parsley (2011)

is the number of documents that contain the term. Hence, this method assigns higher values to unique words than common words compared to BoW.

Nonetheless, both BoW and TF-IDF as described here, fail to incorporate the context of the words. To some extent, this can be circumvented by entering n-grams into the word-document matrix. For instance, a bigram can be used to capture two consecutive words such as "Eindhoven University". Generally, a BoW with bigrams is more powerful than a single word BoW (Goldberg and Hirst, 2017). Nevertheless, the use of n-grams results in many irrelevant entries as well (Goldberg and Hirst, 2017).

To obtain the context of words more successfully, other count-based representations have been implemented. These context-aware systems take the co-occurrences into consideration by creating a co-occurrence matrix. The values at each position in the matrix describe the number of times one word co-occurs with another one. One simple possibility to define the co-occurrence would be to state that a word w co-occurs with another word u if w is written directly after u. However, some word co-occurrences would obtain an inappropriate similarity score because of frequent word occurrences (e.g. 'the', 'and'). Furthermore, the co-occurrence matrix takes up much of the memory due to its sparsity. Therefore, according to Baroni et al. (2014), the counts should be weighted or transformed based on dimensionality reduction techniques such as Singular Value Decomposition (Golub and Kahan, 1965) or non-negative matrix factorization (Lee and Seung, 2001).

One of the most successful methods that use count-based word representations is GloVe (Global Vectors for Word Representations) (Pennington et al., 2014). GloVe embeddings are based on a co-occurrence matrix with a weighted least-squares optimization calculation. It outperforms the prediction-based method of Word2Vec on tasks such as named entity recognition (Pennington et al., 2014).

2.2.2 Prediction-based representations

Different from count-based representations, prediction-based representations use a supervised approach to obtain word-vectors. Basically, this means that the word representation is tweaked based on a training corpus. This helps to create non-sparse (i.e. dense) vectors directly, without the need for additional dimensionality reduction techniques such as Singular Value Decomposition. These prediction-based representations have been referred to as neural-based representations as well (Levy et al., 2015) since these representations are created through neural network algorithms. There are two different types of prediction-based representations that can be distinguished: static and contextual representations. Besides, the previously discussed count-based embeddings can be categorized as static representations as well.

Static representations

The prediction-based representations started to gain popularity when Mikolov et al. (2013) introduced Word2Vec. Word2Vec is can be applied based on the Continuous Bag of Words (CBOW) model or the continuous skip-gram model. The CBOW model attempts to classify a word based on the words that surround the word of interest. For this classification, a simple neural network architecture is used, with one input layer describing the context words, one hidden layer, and one output layer that predicts the target word. The skip-gram model tries to accomplish the opposite from CBOW, by predicting the context words based on the target word. Both approaches create word embeddings that represent the textual data.

Within a few years, Word2Vec had shown its success (Ling et al., 2015; Nguyen and Grishman, 2015; Rocktäschel et al., 2015). Nonetheless, one of the disadvantages of Word2Vec is that words that do not appear in the corpus cannot be assigned a proper value. A more recent static prediction-based method called FastText (Joulin et al., 2016) solves this problem by vectorizing n-grams of characters instead of words. Similar to Word2Vec, FastText applies the CBOW algorithm to represent the n-grams in vector space.

Contextualized representations

One of the major limitations of static word representations is the fact that the same word always has the same vector value, no matter the context. Even long before the introduction of Word2Vec, Schütze (1998) addresses this drawback of word vectors. Already, methods that use n-grams can capture a bit more context. For instance, the character-based n-grams in FastText continue to capture information when a word ends. However, in contrast to these n-gram approaches, contextualized word embeddings tackle the problem of context differently. More specifically, contextualized embeddings dynamically update the word embedding depending on the context a word appears in.

Two modern methods to contextualized representations are ELMo (Peters et al., 2018) and Flair embeddings (Akbik et al., 2018). Both methods use a bidirectional LSTM architecture to embed the textual information into a context-vector. Nevertheless, Flair embeddings are at character-level whereas ELMo applies word-level models.

However, these LSTM-based representations are limited by their recurrent architecture. As discussed in subsection 2.1.3, this architecture cannot be parallelized and therefore is substantially less efficient than transformer architectures (Vaswani et al., 2017). Another approach called BERT (Devlin et al., 2018) avoids this limitation by utilizing the encoder from the transformer architecture to create contextualized embeddings. Since this thesis experiments with the application of BERT, the next section elaborates on the creation and architecture of BERT.

2.3 BERT

The Bidirectional Encoder Representation Transformer (BERT) is a language model that was introduced by Devlin et al. (2018). As the name suggests, it learns bidirectional text representations based on the encoder from a transformer model. Hence, the BERT model mainly relies on the multi-headed self-attention mechanism as explained in subsection 2.1.3. In this section, the internal BERT processes are described first, through the explanation of the input representations (subsection 2.3.1) and pre-training steps (subsection 2.3.2). Subsequently, two major approaches for the usage of BERT are explored in subsection 2.3.3: feature-based BERT and fine-tuned BERT. Finally, subsection 2.3.4 delineates some alternative BERT-inspired models.

2.3.1 Input representations

Similar to Flair and FastText, BERT does not use full words as input. Instead, the text is represented through WordPiece embeddings (Wu et al., 2016). For instance, a word as ”feud” gets split into the WordPieces ”_fe” and ”ud”, where the ”_” character marks the start of a word. The use of WordPieces ensures that out-of-vocabulary words can still be represented while restraining the complexity that character-based models have. In addition to WordPieces, BERT uses the classification token ([CLS]) at the start of each sentence. Furthermore, the separation token ([SEP]) denotes the end of the sentence. In addition to these token embeddings, BERT representations take segment embeddings into account, which are unique for each input sentence. Moreover, the information on the position of each WordPiece is represented through position embeddings. The token, segment, and position embeddings are summed up in order to create the final input embeddings. Figure 2.10 illustrates this construction of the BERT input representations.

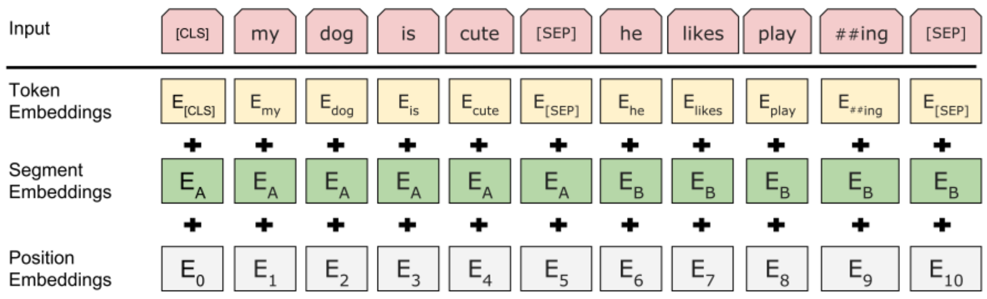

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.10: BERT input representations (Devlin et al., 2018)

2.3.2 Pre-training

The resulting input embeddings are used in the pre-training process of BERT. For English, the pre-training corpus consists of 3,300 million words. Moreover, Devlin et al. (2018) introduce a multilingual BERT as well, that is trained on 104 languages. In contrast to pre-training methods which represent words by considering historic and future words (Peters et al., 2018), BERT pre-trains through two high-level unsupervised tasks called Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

First, MLM is applied to predict a token based on the context tokens. To accomplish this, 15% of the input tokens are masked. Subsequently, the model predicts the tokens that belong on the masked positions. The second task of NSP predicts if a sentence B is the next sentence when an input sentence A is given. According to Devlin et al. (2018), this task helps the model to understand the relationships between sentences.

This pre-training property of BERT is crucial to its success. For instance, Sun et al. (2019) study the effect of pre-training and the number of training samples for the task of sentiment analysis. They discover that more pre-training does not only improve the final model performance but compensates for a lack of task-specific data as well. Accordingly, a model's performance on a small dataset improves when the model is pre-trained. Therefore, the use of pre-trained models opens the gates to research on novel smaller datasets. Consequently, the use of BERT is beneficial to this thesis, since the homicide corpus is relatively small.

2.3.3 Application of BERT

For English, there are two BERT models called BERT-large and BERT-base. BERT-large has 24 encoder layers with 16 attention heads each. BERT-base, on the other hand, consists of 12 encoder layers, each with 12 attention heads. Similarly, for foreign languages Devlin et al. (2018) created multilingual BERT, which uses 12 layers with 12 heads as well.

Irrespective of the model type or size, there are two main approaches to apply BERT for downstream tasks; fine-tuning and feature-based (Devlin et al., 2018).

Feature-based approach

The feature-based method does not update the model weights of the BERT encoder (i.e. it freezes the weights of the layers). Instead, only the pre-trained weights are used to compute the BERT representations (i.e. BERT features). Consequently, the BERT representations are comparable to Flair (Akbik et al., 2018) or ELMo embeddings (Peters et al., 2018). Hence, a feature-based BERT model still needs an additional model (e.g. LSTM) to learn task-specific patterns.

There are several approaches to compute BERT features. More specifically, Devlin et al. (2018) show six different feature extraction methods. Three of them simply extract the outputs from either the embeddings layer, the second-to-last hidden layer, or the last hidden layer. However, more advanced methods use the weighted sum of the last four hidden layers, the concatenation of the last four hidden layers, or the weighted sum of all 12 layers. Depending on the task, different

BERT features can be used. For NER, the authors discover that concatenation of the last four hidden layers results in the best performance on the CoNLL-2003 dataset (F1-score of 0.961 on the development set).

Fine-tuned approach

In contrast to the feature-based approach, fine-tuning does not freeze the weights in the BERT architecture. Instead, fine-tuning updates the pre-trained weights during the training of a down-stream task. Therefore, fine-tuning allows the text representations to incorporate task-specific characteristics. Furthermore, models that apply this fine-tuned approach add a simple task-specific layer on top of the BERT architecture. This layer decodes the fine-tuned representations into the desired output. Devlin et al. (2018) illustrate that this fine-tuned approach marginally outperforms the feature-based BERT for NER (F1-score of 0.964 and 0.961 respectively on the CoNLL-2003 development set). Besides, they visualize the BERT model (Figure 2.11).

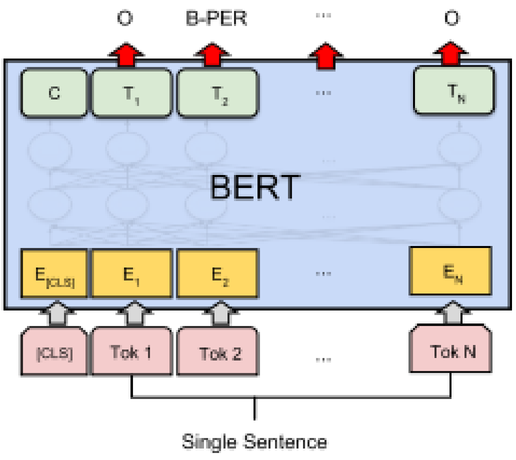


Figure 2.11: Named Entity Recognition with BERT (Devlin et al., 2018)

2.3.4 Models inspired by BERT

The introduction of BERT by Devlin et al. (2018) provided three main models of BERT-base, BERT-large, and multilingual BERT. Since then, there have been many models that are inspired by the original BERT language model. Shortly after the original BERT paper, a pre-trained BERT model for the Chinese language was proposed (Cui et al., 2019). Similarly, De Vries et al. (2019) use the BERT architecture to pre-train on Dutch texts only, resulting in a model called BERTje. Furthermore, the researchers expand the volume of data substantially. Whereas multi-lingual BERT is trained on the full Wikipedia dump (about 1.5GB of Dutch texts), De Vries et al. (2019) describe that BERTje supplements the data with more than 10GB. Aside from Wikipedia, they include Dutch texts that are provided by SoNaR (Stevin Nederlandstalig Referentie) corpus (Oostdijk et al., 2013), web news, books and Twente News Corpus (TwNC) (Ordelman et al., 2007). Due to pre-training on this large volume of language-specific data, the authors achieve better performance for BERTje compared to multilingual BERT on tasks such as NER (F1-score of 0.883 and 0.807 on CoNLL-2002 (Tjong Kim Sang, 2002) respectively) and part-of-speech (POS) tagging.

Besides language-specific models, however, different variants of BERT have entered the research field as well. For instance, Liu et al. (2019) improve the performance on tasks like question-answering, language understanding, and reading comprehension with the Robustly Optimized BERT Pre-Training Approach (RoBERTa). RoBERTa applies only masked language modeling and thus removes the next sentence prediction task in the pre-training process. Furthermore, it is trained in 160 GB of data, compared to only 16 GB of the original BERT models. Although RoBERTa has no multilingual version yet, there are monolingual adaptions such as RoBERT for Dutch (Delobelle et al., 2020).

Another recent BERT-based model, called DistilBERT (Sanh et al., 2019), is a distillation from the normal BERT model. While maintaining 97% of the language understanding of BERT-base, DistilBERT is 60% faster and 40% smaller than the BERT-base model (Sanh et al., 2019).

2.4 Traditional techniques for NER

In the previous sections, I cover generic neural network concepts such as hidden layers, dropout, encoder, decoder, and attention, as well as some neural network architectures like LSTM and the transformer. Furthermore, different approaches to represent text are discussed. One of these text representation methods called BERT is described more extensively. These sections help to comprehend the mechanisms of modern neural network architectures. In contrast, this section and successive sections describe how such networks as well as other approaches have been used to perform NER and SRL. First, in the current section, some of the traditional approaches to NER are delineated.

2.4.1 Conditional random fields

Conditional random fields (CRFs), introduced by Lafferty et al. (2001), is a framework to build probabilistic models for segmentation and labeling of sequential data. Therefore, it is a classification algorithm that can be used for several applications, such as sentence classification (Arulanandam et al., 2014) or named entity recognition (Finkel et al., 2005). Originally, it was presented as an alternative to hidden Markov models (HMMs) and maximum entropy Markov models (MEMMs). Compared to HMMs, conditional models (e.g. CRFs or MEMMs) assure that strict independence between observation is not necessary (Lafferty et al., 2001). Since language observations depend on each other (e.g. context words), conditional methods are more suitable for language tasks than HMMs. On the other hand, MEMMs tend to create a bias towards hidden states (e.g. the unknown named entity labels) that have few outgoing transitions (Lafferty et al., 2001). Where a transition is defined according to the probability of going from one hidden state to another hidden state. Hence, in the case of imbalanced data, MEMMs are not ideal. In contrast, CRFs has the benefits of HMMs and MEMMs without the disadvantages. This is facilitated by the fact that

CRFs calculate the joint probability for a whole sequence rather than individual label probabilities (Lafferty et al., 2001). A simple visualization of a typical CRF network is shown in Figure 2.12. The figure illustrates that the order of the labels (i.e. hidden states) is incorporated in the model (the labels are connected).

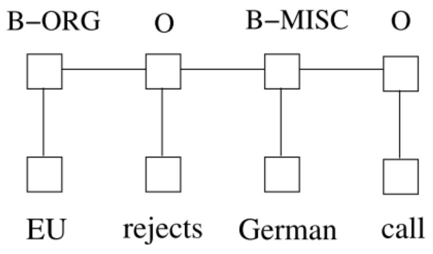


Figure 2.12: CRF model as visualized by Huang et al. (2015)

Today, CRF still proves to be useful for NER. Huang et al. (2015) show that a CRF model outperforms LSTM and BiLSTM networks on CoNLL-2003 when using Senna word-embeddings and gazetteer features (i.e. pre-defined dictionary of entity values such as city names). However, the authors show that combining one of these recurrent neural networks with a CRF classifier results in the best performance (F1-score of 0.901 on CoNLL-2003 for BiLSTM-CRF and 0.884 for LSTM-CRF). Furthermore, other researchers apply similar LSTM-CRF architectures (Akbik et al., 2018; Lample et al., 2016; Yadav et al., 2018). The reason for this advantageous combination can be deduced from Figure 2.13. From the figure, it is evident that this approach acquires information about the observations through the LSTM model while maintaining information on joint probabilities of the sequence labels through the CRF layer.

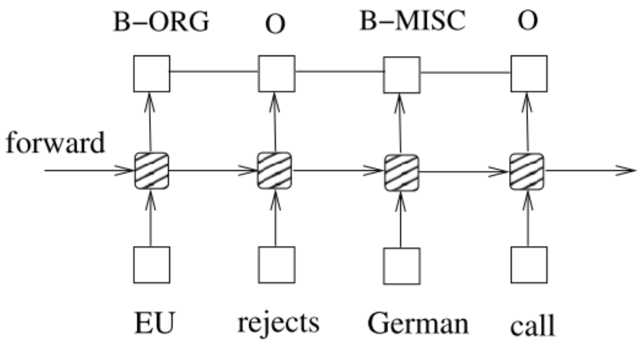


Figure 2.13: LSTM-CRF model as visualized by Huang et al. (2015)

2.4.2 Recurrent neural networks

As explained in the previous subsection, based on the results from Huang et al. (2015), the (bidirectional) LSTM models perform worse than a CRF algorithm in case of NER. However, when both approaches are combined, the best performance is achieved. Yadav et al. (2018) apply this approach for Dutch NER as well. The authors first use randomized character-level embeddings for the initialization of a character-based BiLSTM. Secondly, this character information is combined with Dutch FastText embeddings (Bojanowski et al., 2017) and subsequently provided as input for another BiLSTM. Finally, they add a CRF layer to classify the entities. With this system, the researchers obtain a F1-score of 0.875 on CoNLL-2002.

Instead of a CRF classification layer, a simple Softmax function can be applied as well. Generally speaking, however, CRF outperforms Softmax (Reimers and Gurevych, 2017). In line with this notion, Figure 2.14 illustrates that CRF generally scores higher than Softmax in multiple BiLSTM models that use either CRF or Softmax as a final layer.

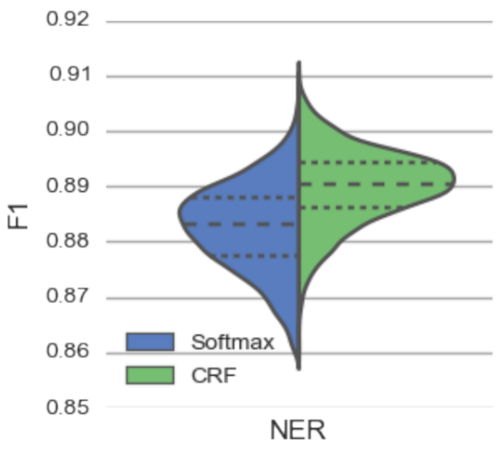


Figure 2.14: F1-scores on CoNLL-2003 using static word-embeddings with CRF or Softmax as last layer of a BiLSTM network, as visualized by Reimers and Gurevych (2017)

Instead of CRF or Softmax, one can apply another LSTM to decode the encoded representations. These models are known as sequence-to-sequence (seq2seq) models (Bahdanau et al., 2014). These seq2seq models, however, are aimed to perform more complicated language tasks such as neural machine translation (Bahdanau et al., 2014; Cho et al., 2014b). Still, Straková et al. (2019) compare seq2seq models with LSTM-CRF models for the more simple language task of NER. The authors find that for flat NER the LSTM-CRF outperforms the seq2seq model (F1-score of 0.934 and 0.931 on CoNLL-2003 respectively). Nonetheless, for the more sophisticated task of nested NER, where entities can be part of an outer entity as well, the seq2seq model achieves a higher F1-score compared to LSTM-CRF (0.844 and 0.812 on ACE-2004 respectively). Hence, their results support the claim that seq2seq is more useful when the task is more complex.

2.4.3 Other supervised methods

Still, other approaches to NER achieve good performance as well. Agerri et al. (2016) introduce an end-to-end information extraction system for events covered in the news in the Newsreader project. The project attempts to discover what happened, where, when, and who was involved by analyzing news articles in English, Dutch, Italian, and Spanish. They implement models for fourteen NLP tasks such as NER and SRL. For NER, they use the ixa-pipe-nerc module (Agerri et al., 2014), which uses the perceptron algorithm as described by Collins (2002). Although the perceptron algorithm is relatively simple compared to the more frequently implemented RNNs, the algorithm still obtains a F1-score of 0.850 on the Dutch CoNLL-2002 corpus (Agerri et al., 2016) and a F1-score of 0.877 on SoNaR (Vossen et al., 2016). Similarly, the relatively simple model achieves a F1-score of 0.914 on the English CoNLL-2003 corpus. Nonetheless, as shown by Vossen et al. (2016), the simple perceptron algorithm achieves a F1-score of only 0.639 on Dutch and 0.701 on English out-of-domain texts from the self-constructed MEANTIME (Multilingual Event ANd TIME) corpus (Minard et al., 2016).

2.4.4 NER on novel ontologies

Most of the research in NER focuses on regular entities such as a person, location and organization. In the current thesis, however, more specific entities such as the victim or suspect need to be distinguished. Such a set of entities is referred to as an ontology. Similar to this thesis, a subset of NLP research examines custom ontologies as well. For instance, Fleischman and Hovy (2002) distinguish between eight entity types (i.e. athlete, businessperson, doctor, entertainer, lawyer, police, politician). The authors create features based on local semantic context as well as the more global topic of the text. Subsequently, the contextual information is inputted into several classifiers in order to predict the correct entity labels. Their most successful classifier is a decision tree classifier C4.5 (Quinlin, 1993), achieving an entity accuracy of 70.1%.

In contrast to Fleischman and Hovy (2002), Cimiano and Völker (2005) apply an unsupervised approach for NER on novel entity ontologies. The authors propose an unsupervised NER method using the syntactic context of each word. According to the researchers, their unsupervised approach potentially enhances NER performance when novel entity classes need to be recognized. Furthermore, they argue that supervised approaches are less efficient since all the classes need to be labeled. The researches create a new ontology that contains over 600 different classes. In the end, their best rule-based algorithm obtains a F1-score of 0.326.

2.5 Traditional techniques for SRL

Whereas a typical entity annotation in NER involves the full name mention, a semantic role can be annotated in various ways. Hence, in comparison to a named entity, the definition of a semantic role is less evident. Nonetheless, the definition of a semantic role is especially relevant for this thesis, since the semantic roles need to be informative for the police. Therefore, this section discusses some SRL annotation techniques first (subsection 2.5.1). Subsequently, the second part of this section describes some traditional SRL algorithms that are used in prior research (subsection 2.5.2).

2.5.1 SRL annotation

Semantic role labeling enables the discovery of semantic roles that are related to a given predicate. As addressed by Màrquez et al. (2008), the PropBank annotations (Palmer et al., 2005) are primarily applied in the task of SRL. The PropBank only labels verb predicates and uses syntactic constituents (i.e. spans) to represent arguments. The arguments represent the semantic roles in the sentence. Palmer et al. (2005) define five core arguments that are presented in PropBank. The first two arguments are independent of the verb. Hence, for all verbs Argument 0 (Arg0) is the "Agent" and Argument 1 (Arg1) is the "Patient" or "Theme" of the verb. The other arguments (Arg2, Arg3, Arg4) depend on the verb predicate. For instance, the verb "kill" has an Arg2 defined to be an "instrument". Aside from the five arguments, PropBank defines eleven adjuncts as well. These are labeled to be argument modifiers (ArgM). The adjuncts describe the location (ArgM-LOC), time (ArgM-TMP), negation (ArgM-NEG), direction (ArgM-DIR), and more.

After the introduction of PropBank other corpora such as CoNLL-2005 (Carreras and Márquez, 2005), CoNLL-2012 (Pradhan et al., 2012) or the Dutch SoNaR (Schuurman et al., 2010) still adhere to the PropBank annotation structure. Nevertheless, other styles of annotations have been developed as well. For instance, even though the traditional SRL task evolves around verb predicates (Màrquez et al., 2008), more recent approaches try to enrich the predicate types. Bonial et al. (2014) propose PropBank style annotations with the addition of adjective and noun predicates. The authors argue that verbs will not always capture the relevant information of a sentence, as illustrated by the following three sentences:

1. John fears Jack
2. John is afraid of Jack
3. John's fear of Jack

The researchers show that the traditional PropBank would correctly identify "John" and "Jack" to be separate arguments of the verb predicate "fears". Nonetheless, in the second sentence the relation of the verb "is" will contain "John" as well as the whole span of "afraid of Jack". Moreover, it would fail to return any argument in the last sentence, since there is no verb present. Therefore, Bonial et al. (2014) argue that adjective predicates ("afraid" in the second sentence) and noun predicates ("fear" in the third sentence) need to be considered to find the most meaningful relations.

As delineated by Màrquez et al. (2008), when the input sentence and its designated verb are provided, a SRL system should identify the arguments and classify the semantic role for each of the arguments. There are several methods to accomplish this task. As suggested by Shi and Lin (2019) and Li et al. (2019), the two core approaches are the dependency-based SRL and span-based SRL. For span-based SRL the whole argument is annotated. In contrast, dependency-based SRL labels one word of the argument based on the word dependencies in the phrase. Frequently, this word corresponds with the head word of the dependency structure (Johansson and Nugues, 2008; Li et al., 2019). An example of the span and dependency annotation techniques is visualized by Li et al. (2019) in Figure 2.15.

The figure illustrates that the dependency-based annotations consist of one word (i.e. head word), whereas the span annotations cover the whole argument span. However, the head words are not always informative (e.g. "from" in Figure 2.15). Therefore, Surdeanu et al. (2003) propose the use of content words instead of head words. According to the authors, these content words still use the dependency structure but convey the most informative information. Hence, instead of the head word "from" the content word "John" would be labeled in the example from Figure 2.15.

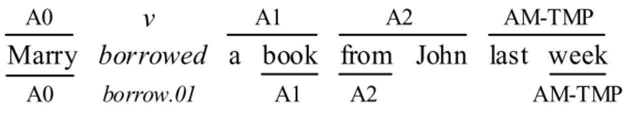


Figure 2.15: Example of span (above) and dependency (below) SRL annotations, as visualized by Li et al. (2019)

2.5.2 SRL algorithms

Many algorithms have been applied to perform SRL. At first, researchers used syntactic features as input for a linear classifier such as SVMs (Pradhan et al., 2004, 2005). Typically, these features are features such as the part of speech tag, the position of the phrase, the syntactic head of the phrase (Gildea and Jurafsky, 2002). During these early stages, the state-of-the-art for the SRL algorithms came close to a 70% F1-score (0.696 F1-score on PropBank for Pradhan et al. (2005)). Furthermore, other research combines feature input has with non-linear classification algorithms. Daelemans et al. (2010) apply a memory-based learning algorithm to find word similarity from syllable-based features. As a classifier the authors use a combination of the decision tree classifier IGTREE (Daelemans et al., 1997) and the nearest neighbor algorithm. Using this method, Vossen et al. (2016) obtain a F1-score of 0.740 for SRL on the Dutch SoNaR dataset. Nevertheless, with the rising popularity of neural networks, the technical methods for SRL have been adapted. Although some neural network models still use some type of syntactic features (Fitzgerald et al., 2015; Roth and Lapata, 2016), other researchers have proposed syntax-agnostic models (Li et al., 2019; Ouchi et al., 2018; Zhou and Xu, 2015).

Zhou and Xu (2015) were the first to pioneer with the syntax-agnostic approach. They apply a BiLSTM-CRF model with self-trained 32-dimensional word embeddings as input. The model outperforms previous syntactic-based approaches, by achieving a F1-score of 0.828 on CoNLL-2005 and 0.813 on CoNLL-2012. More recently, Ouchi et al. (2018) trumped these performances by using an ensemble method (F1-score of 0.885 on CoNLL-2005 and 0.870 on CoNLL-2012). The ensemble method combines five runs of BiLSTM models that are initialized with ELMo embeddings.

2.6 BERT for NER and SRL

Even though traditional algorithms have resulted in high NER (Yadav et al., 2018) and SRL (Ouchi et al., 2018) performances, BERT still improves the task performances to some extent. Hence, in this section delineates different BERT approaches for NER (subsection 2.6.1) and SRL (subsection 2.6.2).

2.6.1 BERT for NER

Recently, Straková et al. (2019) obtain state-of-the-art performances for named entity recognition in English, Dutch, and Spanish, using a feature-based method. The authors experiment with BERT, ELMo, and Flair representations. Inspired by Devlin et al. (2018), the researchers concatenate the final four hidden layers to calculate the BERT features. Subsequently, the representations are entered into a model that uses a BiLSTM to encode the representations and a CRF layer to decode the output from the BiLSTM into NER labels. For Dutch NER on CoNLL-2002, the combination of BERT and Flair representations results in the state-of-the-art performance (F1-score of 0.927). Still, when only BERT representations are used, the model achieves a comparable F1-score of 0.925. Moreover, the research discovers that models with BERT representations outperform models without BERT representations for every tested language.

Despite the fact that state-of-the-art performances for NER are achieved with feature-based BERT (Straková et al., 2019), Devlin et al. (2018) and Peters et al. (2019) show that the performance from fine-tuned BERT surpasses feature-based BERT for NER and other NLP tasks. Nonetheless, performance differences are relatively minor. For instance, as described earlier in section 2.3.3, Devlin et al. (2018) find that a fine-tuning approach obtains a F1-score of 0.964 on the ConLL-2003 development set, compared to 0.961 of the best performing feature-based model. Similarly, Peters et al. (2019) conclude that fine-tuned BERT with a Softmax classification layer slightly beats a feature-based BERT with BiLSTM and CRF layers (F1-score of 0.924 and 0.922 on CoNLL-2003 respectively).

Next to English NER, Souza et al. (2019) show that the results from the fine-tuning approach exceed results from feature-based BERT on Portuguese NER on the HAREM corpus (Santos et al., 2006) as well (F1-score of 0.734 and 0.721 respectively). Furthermore, the authors apply an additional fine-tuned approach that adds a CRF layer to the transformer model instead of a more basic Softmax classifier. This results in the best performance (F1-score of 0.742).

2.6.2 BERT for SRL

In addition, Shi and Lin (2019) show that the fine-tuning approach is successful in the extraction of relations through relation extraction and semantic role labeling. The authors implement BERT-based models for both these tasks. In the case of SRL, they produce state-of-the-art results on CoNLL-2005 (F1-score of 0.888). Their model has BERT as an encoder and decodes the representations with one BiLSTM layer and a final multi-layer perceptron as classification layer. Still, compared to alternative methods (Ouchi et al., 2018), the performance differences of in-domain results are small. Furthermore, for CoNLL-2012 the ensemble model from Ouchi et al. (2018) scores even higher than the BERT-based approach (F1 of 0.870 and 0.865 respectively). Nonetheless, the BERT-based model from Shi and Lin (2019) outperforms the ensemble approach in the out-of-domain performance (F1 of 0.820 and 0.796 on CoNLL-2005 respectively).

2.7 Related work in the crime domain

As the previous section explains, BERT-based approaches have been successfully experimented with for IE-related tasks such as NER (Devlin et al., 2018; Straková et al., 2019), and SRL (Shi and Lin, 2019). Nonetheless, BERT has not been applied for information extraction on crime texts. Still, other research does study crime-related information extraction. Therefore, this section describes some of these prior studies (subsection 2.7.1). Furthermore, I discuss what information is relevant to extract for crime texts in subsection 2.7.2.

2.7.1 Information extraction from crime texts

The automatic extraction of information from crime texts has mainly been considered for the English language. Still, Schraagen et al. (2017) apply a NER system on Dutch fraud reports. The researchers create their own manually annotated fraud dataset for evaluation, consisting out of 250 reports involving fraud with 1,059 annotated named entities. The authors use the NER classifier provided by Frog (Van Den Bosch et al., 2007), which is trained on the SoNaR-1 corpus (Schuurman et al., 2010). Frog applies memory-based algorithms to tackle Dutch NLP-tasks which are based on TiMBL (Daelemans et al., 2010), the Tilburg Memory Based Learner. For NER, TiMBL uses a decision-tree classifier called IGTree (Daelemans et al., 1997). Using this method, Schraagen et al. (2017) obtain a F1-score of just 0.38 for NER on the fraud dataset. This is substantially worse than the IGTree performance that Desmet and Hoste (2014) achieve on the SoNaR-1 corpus (F1-score of 0.77). These results imply that using this transfer-learning approach is not sufficient for NER on Dutch fraud texts.

For the English language, there are more successful attempts to extract crime entities. Arulanandam et al. (2014) extract crime locations using 70 articles (523 sentences) about theft cases. First, they split the text into tokenized sentences using the Punkt-tokenizer (Kiss and Strunk, 2006) from the Natural Language Toolkit (NLTK) (Bird et al., 2009). Secondly, the researchers apply NER to identify all the location entity mentions within each sentence using the LBJ-Tagger (Ratinov and Roth, 2009). This tagger is based on a perceptron classifier and gazetteers. Subsequently, the authors deploy a CRF classifier to distinguish between sentences that include crime locations and sentences that do not. Although they obtain promising results on recognizing the sentences with crime locations (F1-score around 0.90), their model is unable to extract crime locations at a word level. Furthermore, information that is spread out over multiple sentences cannot be extracted.

To assure that entity information across sentences is maintained, Dasgupta et al. (2017) use coreference resolution as described in the Stanford CoreNLP toolkit (Manning et al., 2014). Aside from the crime location, the authors extract the victim name, accused name, date of crime, and nature of the crime from 3,000 crime-reporting news articles. To identify these different entity types, they implement a three-step process. First, entities such as names and locations are extracted according to the default Stanford Named Entity Recognizer (Finkel et al., 2005). The Stanford NER module applies CRF in combination with Gibbs sampling (Geman and Geman, 1984). Secondly, to distinguish between victim and accused name, each word of the text is classified into a related crime category (e.g. victim). The classification is performed using the co-occurrence of words with an SVM classifier. Thirdly, the authors apply a statistical learning method based on mutual information to improve name resolution. Since crime texts usually refer to the same person in multiple ways (full name, last name, name abbreviation, etc.), the researchers consider name resolution a valuable step in the process of information extraction. Nonetheless, the extraction of criminal names performs poorly (F1-score of 0.64) compared to the location and date (F1-score of 0.88 and 0.87 respectively). In addition to the extraction of entities, the researchers use the dependency parser from the CoreNLP toolkit to extract relations within sentences. The extraction of these relations in combination with the entity extraction enables them to construct a crime knowledge base.

Recently, Srinivasa and Thilagam (2019) expand the idea of a crime knowledge base by merging information of multiple crime texts into one knowledge graph. The authors extract several entities such as persons, locations, days, and organizations. They use a rule-based approach for NER and relation extraction. However, according to Chiticariu et al. (2013), most academics proclaim that rule-based systems are obsolete compared to ones that use machine learning. That said, the main goal of the researchers is to merge information into a knowledge graph rather than information extraction itself. For this purpose, they apply a semantic merging method. This method merges information based on similarities of vector representations of the texts. This allows the researchers to omit duplicate information.

2.7.2 Twelve components of interest

Before one can extract relevant information, the scope that defines the relevance needs to be defined. In crime-related IE, the relevant information goes beyond proper name and location extraction, since victim names and suspect names (Dasgupta et al., 2017), as well as crime locations and other locations (Arulanandam et al., 2014) need to be differentiated. Generally, according to De Kock (2014), there are twelve components that describe all relevant aspects of a crime case.

De Kock (2014) defines twelve elementary scenario components (ESC12) that can be used to describe any narrative. Nevertheless, the author only considers the application of ESC12 to anticipate criminal behavior. Therefore, the described components are useful in the context of the current thesis. The twelve components consist of objective components (i.e. arena, time(frame), context, protagonist, antagonist, means, modus operandi, resistance), subjective components (i.e. motivation, primary objective), and interpretable components (i.e. symbolism, red herring). The objective components are especially useful for information extraction since they are not dependent on any interpretations.

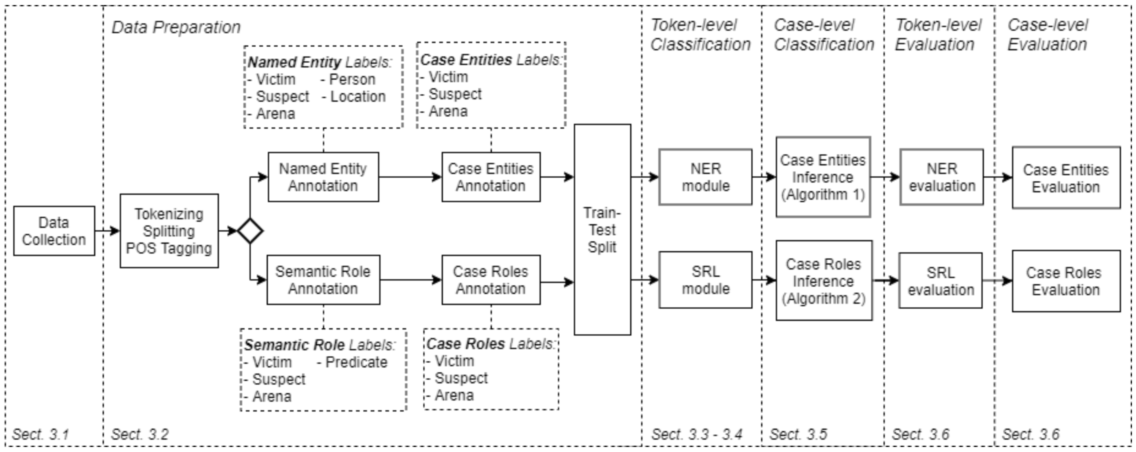
Besides the twelve components, each of the scenario components is composed of variables. For instance, the arena is described according to the variables: country, city, kill zone, static location, general description, etc. Similarly, the protagonist component is subdivided into variables as well. These protagonist variables can describe whether the protagonist is confirmed to have committed the crime, the name of the protagonist, or the number of captured protagonists. Consequently, the scenario components and the corresponding component variables constitute a crime-specific ontology. Evidently, such an ontology is useful for crime-related information extraction.

Chapter 3

Methodology

In this thesis, I build a NER-based system to extract crime entities and a SRL system to enable the extraction of textual crime-related descriptions. However, before NER or SRL can be applied, the Dutch homicide texts are collected and prepared correctly. Furthermore, the raw output from NER or SRL is insufficient, since both these tasks only provide token-level labels. Therefore, these token-level classifications are translated into labels at the case level through two rule-based algorithms. Finally, the token-level and case-level predictions are evaluated.

Altogether, the experimental pipeline consists of six main parts: (1) Data collection (section 3.1), which scrapes and filters the data; (2) Data preparation (section 3.2), which prepares the data and defines a homicide ontology for NER and SRL to enable further processing; (3) Crime token classification (section 3.3 and section 3.4), which delineates BERT-based models for NER and SRL to classify tokens; (4) Crime case classification (section 3.5), which defines rule-based algorithms to translate the token-level output into case-level labels; (5) Token-level evaluation (section 3.6), which evaluates the token-level predictions; (6) Case-level evaluation (section 3.6), which evaluates performance of the case-level predictions. The complete pipeline is visualized in Figure 3.1. Furthermore, I reference to the code and elaborate on the implementation in Appendix D.



Data Preparation
Token-level Classification
Case-level Classification
Token-level Evaluation
Case-level Evaluation
Named Entity Labels: - Victim - Person - Suspect - Location - Arena
Case Entities Labels: - Victim - Suspect - Arena
Data Collection
Tokenizing Splitting POS Tagging
Named Entity Annotation
Case Entities Annotation
Semantic Role Annotation
Case Roles Annotation
Train-Test Split
NER module
SRL module
Case Entities Inference (Algorithm 1)
Case Roles Inference (Algorithm 2)
NER evaluation
SRL evaluation
Case Entities Evaluation
Case Roles Evaluation
Semantic Role Labels: - Victim - Predicate - Suspect - Arena
Case Roles Labels: - Victim - Suspect - Arena
Sect. 3.1
Sect. 3.2
Sect. 3.3 - 3.4
Sect. 3.5
Sect. 3.6
Sect. 3.6

Figure 3.1: Experimental pipeline

Information to extract

As explained in subsection 2.7.2, there are a total of twelve relevant components. Nonetheless, this research limits its scope by focusing on the extraction of three main components: protagonist, antagonist, and arena. Using these three components helps to find the suspects (i.e. protagonists) victims (i.e. antagonists), and homicide locations (i.e. arenas). Importantly, to avoid confusion with the ESC12 components as described by De Kock (2014) and the three crime-related components in this thesis, I refer to the current crime-related components as crime classes. To be clear, these crime classes are the victim, suspect, and arena class. The choice for these three classes is in line with selected entities from related work in the crime domain (subsection 2.7.1). For instance, Arulanandam et al. (2014) use the location of crime as only entity of interest, and Dasgupta et al. (2017) extract the crime location, victim and suspect entities.

Furthermore, in the interest of time, not all the relevant component variables (De Kock, 2014) are considered. More specifically, only the "city name", "name of the protagonist" and "name of the antagonist" are taken into account in the NER module. On the other hand, the SRL module is less restricted since it extracts more than just names. Consequently, variables such as "Kill zone", or "Protagonist Age" can be discovered by the SRL module. Still, the SRL module is designed to distinguish between the three crime classes and not between specific component variables.

3.1 Data collection

The IE systems that are proposed in this thesis use the self-constructed homicide corpus for training and evaluation. This homicide corpus is composed out of textual Dutch descriptions of homicides that took place in the Netherlands sometime between 2005 and 2011. The data is scraped from Moordzaken.com[1]. This website describes cases involving 712 victims, where each victim has its separate web page (hereafter victim page). This victim page is divided into four parts: an information table, a case description, a suspect description, and the court decision. See Figure A.1 in Appendix A for an example. From these four parts, the information table and case description are present for all the victim pages, whereas the suspect description and court decision are optional.

The information table contains structured information about the case, such as the victim name, crime location, and current status of the case (solved or in progress). In contrast, the case description, suspect description, and court decision are unstructured texts. Since this thesis focuses on information extraction from unstructured raw texts, the information table only helps to accelerate the labeling process. Consequently, the table is excluded from the training corpus. Additionally, the court decisions are excluded because these paragraphs complicate the text without providing new information regarding the victim, suspect, or arena.

After the case descriptions and suspect descriptions are scraped, these texts combined into one text for each victim page. Subsequently, there are two main filtering steps to conduct. First, the crime classes are not necessarily included in the raw text. Hence, to ensure that all texts contain

[1]`https://moordzaken.com/Moordlijsten`

sufficient relevant information, some of the victim pages are filtered out. More specifically, only the victim pages with a text that contains the victim name and the crime location (according to the information table) are included.

Second, when a homicide case has multiple victims, the website creates multiple victim pages with duplicated texts. These duplicated texts can affect the training due to the over-representation of a homicide case. Therefore, when there are at least two victims, all the victims are assigned to one victim page. Subsequently, the remaining duplicate victim pages are removed from the homicide corpus. After these steps, 510 cases remain (712 at the start, 542 after the first filtering step, 510 after the second filtering step). These 510 cases constitute the final homicide corpus. In total, this corpus contains 4,454 sentences.

3.2 Data Preparation

After the homicide cases are collected and filtered, the data is prepared to allow for the automatic extraction of the crime classes. In this section, I explain how the case texts are split into sentences and tokens at first (subsection 3.2.1). Subsequently, the procedures of annotating four different annotation types (i.e. entities, semantic roles, case-level entities, case-level roles) are discussed. First, the named entities are annotated (subsection 3.2.2). These entity-annotations are at token-level and constitute as input for the NER module. Similarly, the token-level semantic roles are annotated (subsection 3.2.3) to prepare the data for semantic role labeling by the SRL module. However, in contrast to prior NER and SRL research, in this thesis the token-level annotations need to be transformed into case-level annotations such that the victim(s), suspect(s), and arena(s) can be distinguished for each case separately. Therefore, subsection 3.2.4 describes the labeling of entities at a case-level, and subsection 3.2.5 elucidates the annotation procedure concerning the case-level semantic roles.

3.2.1 Data tokenization, sentence splitting and POS tagging

The approach for tokenization and sentence splitting and POS tagging is the same as Agerri et al. (2016) propose for the end-to-end Dutch Newsreader system. Accordingly, I use the publicly available modules from the IXA-pipeline[2] (Agerri et al., 2014). More specifically, the texts are tokenized with the Alpino parser[3] according to the IXA-tokenizer. Since the Alpino parser is designed for the Dutch language, it is the parser of choice for the current homicide corpus. Furthermore, the IXA-sentence-splitter is applied in order to split the sentences. Finally, the IXA-pos-tagger is implemented to label the tokens with their corresponding POS tags. Based on the results from Agerri et al. (2014), this tagger achieves around 97% accuracy. Still, after these steps, a minor tokenization issue remains. Namely, for a name abbreviation such as "John C.", the "." is separated from the "C" when it appears as the final word in a sentence. Hence, the produced tokens are "John", "C" and ".." instead of "John", "C." and ".". Nevertheless, this problem is solved through a small python script.

[2]https://ixa2.si.ehu.es/ixa-pipes/

[3]http://www.let.rug.nl/vannoord/alp/Alpino/

3.2.2 Named entity annotation

In this thesis, I apply two different token-level annotation methods. The current section elaborates on the creation of the named entity annotations. The second method is called semantic role annotation, where a broader range of phrases is labeled. This second approach is explained in more detail in the next subsection.

BILOU annotation

For the entity labels, the BILOU annotation method is applied. Even though the BIO scheme is a frequently applied annotation method (CoNLL-2002, MEANTIME, SoNaR), BILOU annotations have shown to outperform BIO annotations (Ratinov and Roth, 2009) by a substantial margin (up to 5.1% F1-score). For the BIO scheme, the first token of an entity mention is labeled with a begin label ('B-*ENTITY*'). Subsequently, if consecutive tokens continue with the entity mention, the tokens are labeled with an inside label ('I-*ENTITY*'). Finally, all the remaining tokens are labeled with the outside label ('O'). The BILOU adapts this scheme such that the latest I-token is replaced with a last label ('L-*ENTITY*'). Furthermore, if an entity mention consists of only one token, this token is labeled with a unit label ('U-*ENTITY*').

NER annotation guidelines

The named entities used in this thesis follow a flat labeling structure for personal names and location names. Hence, no unnamed descriptions or references (e.g. the man, he) are considered. This annotation style is in line with annotation guidelines provided CoNLL (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) and MUC-6 (Grishman and Sundheim, 1996). Nevertheless, the current entity crime classes (i.e. victim, suspect, arena) are different from the classes that CoNLL and MUC consider. Therefore, each of these entity classes is properly defined such that the entity mentions can be annotated in a consistent manner. A detailed overview of definitions for the entity classes is presented in Table 3.1.

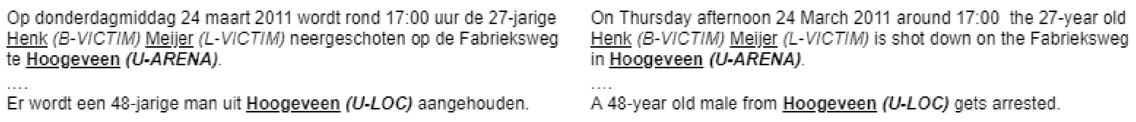
The definition of the victim class is relatively straightforward. Regarding the suspect class, however, it should be noted that people who have been acquitted are still labeled to be a suspect. Consider a text which describes that a person got arrested and, at a later stage, the text explains this person got acquitted. Since the NER module only considers sentence-level input, such a cross-sentence pattern cannot be discovered. Therefore, to maintain consistency in the annotations, this particular person would still be labeled to be a suspect since the person was arrested at one point.

Furthermore, the persons and locations that are no victim, suspect or arena, are considered as well. It is assumed that this enrichment of information helps the NER module to distinguish person and location names more effectively. Moreover, by adding regular person and regular location to the entity classes, the corpus allows for a wider range of future experiments. For instance, it enables the use of traditional NER models that are not trained with crime-specific entity classes.

Entity class	**Definition**	**Annotation**	**Count**	**Avg. per case**
VICTIM	Any mention of a person who got murdered.	Name mention Nickname or name abbreviation	1,841	3.6
SUSPECT	Any mention of a person who has committed the homicide or has been arrested at some point.	Name mention Nickname or name abbreviation	892	1.7
PERSON	Any mention of a person that is not VICTIM or SUSPECT	Name mention Nickname or name abbreviation	83	0.2
ARENA	Any mention of the location of the homicide where the mention is clearly linked to the homicide. Or, if no location of the homicide is present, any mention of the location where the body was found. Again, the mention is clearly linked to the finding place.	City name	510	1.0
LOCATION	Any mention of a location that is not ARENA.	City name Region name (district level or larger) Country name	269	0.5

Table 3.1: Named entity annotations for the homicide corpus

With respect to the locations, the difference between the ARENA and LOCATION entity-classes can be confusing. Namely, the city of the homicide can be labeled as LOCATION at instance A, even though the same city name is labeled as ARENA at instance B in the text. However, this only happens if instance A does not clearly link to the homicide. Figure 3.2 illustrates how the same city can be labeled with different classes because the second mention does not directly relate to the location of the homicide.



Op donderdagmiddag 24 maart 2011 wordt rond 17:00 uur de 27-jarige Henk *(B-VICTIM)* Meijer *(L-VICTIM)* neergeschoten op de Fabrieksweg te **Hoogeveen** ***(U-ARENA)***.
....
Er wordt een 48-jarige man uit **Hoogeveen** ***(U-LOC)*** aangehouden.

On Thursday afternoon 24 March 2011 around 17:00 the 27-year old Henk *(B-VICTIM)* Meijer *(L-VICTIM)* is shot down on the Fabrieksweg in **Hoogeveen** ***(U-ARENA)***.
....
A 48-year old male from **Hoogeveen** ***(U-LOC)*** gets arrested.

Figure 3.2: Example with labeled entity classes. It illustrates that the same city can be labeled differently (shown in bold).

NER annotation procedure

The named entities are annotated according to two consecutive processes: the automatic annotation and the manual annotation. The automatic annotation uses the victim name and the crime location (i.e. arena) from the information table to automatically annotate these entities for the tokenized text. Still, during the manual annotation phase, I read all the texts in the homicide corpus to review the automatic annotations for missing or wrong victim and arena annotations. Furthermore, I manually annotate all the suspect, person, and location mentions. These manually adapted and annotated labels constitute the ground truth NER values for the homicide corpus.

3.2.3 Semantic role annotation

For the semantic role annotations, not only the names but descriptions and references of the victim, suspect, and arena are labeled as well. As a consequence, more component variables, as defined by De Kock (2014), can be captured. For instance, the description might contain the number of suspects or the kill zone (i.e. urban or rural).

In this subsection, I first define the semantic role annotation guidelines. These guidelines specify the homicide ontology for SRL. Secondly, the annotation procedure is described.

SRL annotation guidelines

The semantic role annotations used in this thesis are specifically tuned for the homicide context. Nonetheless, the PropBank guidelines (Palmer et al., 2005) are considered to be the foundation for the crime role labels. Hence, in line with these guidelines, I label one verb predicate for each sentence. Consequently, when there are multiple predicates in a sentence, the sentence is duplicated such that each duplicate has one labeled predicate remaining. Besides the predicates, the PropBank annotations contain argument spans (Arg0, Arg1, etc.), as discussed in subsection 2.5.1. In a straightforward example, the PropBank Arg0, Arg1, and ArgM-LOC annotations represent the victim, suspect, and arena classes. For example:

$$[\text{John}]_{Arg0}\ [\text{kills}]_{V}\ [\text{Jack}]_{Arg1}\ [\text{on the street}]_{ArgM-LOC}\ .$$

However, an Arg0 span does not inherently contain the suspect and an Arg1 span does not have to refer to the victim. For instance, according to the PropBank guidelines the suspect should be labeled as Arg1 for the verb-predicate "arrested". Therefore, the arguments as prescribed by the PropBank guidelines cannot be applied for the crime role annotation directly. Consequently, I label the PropBank argument spans with the corresponding crime class, disregarding the type of PropBank argument (i.e. Arg0, Arg1, Arg2, etc.). This general annotation process can be formulated as follows:

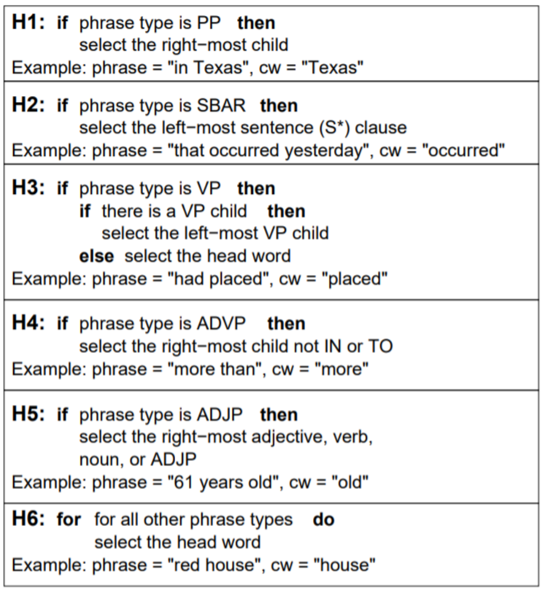
Given a sentence:

1. Identify the predicate V
2. Identify the PropBank argument spans of V
3. Based on the argument spans infer the crime class for the spans

Nevertheless, the texts can have a complicated structure where the crime class span is different from the regular PropBank argument spans. In this thesis, it is desirable to keep close to the PropBank representations, since those annotations have proven to be successful over the years

(Màrquez et al., 2008). Moreover, it is expected that the consistency of using the PropBank argument spans leads to more robust results since only a small set of data is available to learn patterns from. On the other hand, it is advantageous to have the labels resemble the actual crime classes of victim, suspect, and arena. Otherwise, post-processing algorithms would still need to extract the classes from a potentially ambiguous argument span. Furthermore, if one only uses unambiguous argument spans to represent the crime class spans, some relevant information about the crime classes gets lost. Therefore, a couple of annotation rules are defined. These rules help to label the homicide cases in a structured manner while maintaining most information that is specific to the crime classes.

First, there is a possibility that a crime class mention (i.e. a mention in the text that refers to the victim, suspect, or arena class) is embedded within a PropBank argument span. For instance the victim "Jack" in the sentence "The bike of Jack was found.", is embedded in the PropBank argument span "The bike of Jack". Obviously, "Jack" is a victim but "The bike of Jack" is no victim. For such an ambiguous case, the content word, as proposed by Surdeanu et al. (2003), is identified. According to the authors, the content word is the most informative word in a given phrase. Furthermore, the authors define six heuristics to extract the content word of a phrase (Figure 3.3).



```
H1: if  phrase type is PP   then
        select the right-most child
Example: phrase = "in Texas", cw = "Texas"

H2: if  phrase type is SBAR   then
        select the left-most sentence (S*) clause
Example: phrase = "that occurred yesterday", cw = "occurred"

H3: if  phrase type is VP   then
        if  there is a VP child   then
            select the left-most VP child
        else  select the head word
Example: phrase = "had placed", cw = "placed"

H4: if  phrase type is ADVP   then
        select the right-most child not IN or TO
Example: phrase = "more than", cw = "more"

H5: if  phrase type is ADJP   then
        select the right-most adjective, verb,
        noun, or ADJP
Example: phrase = "61 years old", cw = "old"

H6: for  for all other phrase types   do
         select the head word
Example: phrase = "red house", cw = "house"
```

Figure 3.3: Heuristics to detect the content word, as defined by Surdeanu et al. (2003)

Following from these heuristics, PropBank arguments can be filtered based on informative words. More specifically, PropBank argument spans that do not have part of a crime class mention as a content word or are not related to a crime class mention through a conjunction, are excluded. This avoids labeling the argument "The bike of Jack", which has "bike" as the content word. Nevertheless, for the argument span "A friend and Jack", the argument span is kept for further processing, since the content word "friend" is related to victim "Jack" through the conjunction "and". The next further processing step demands that the sub-argument spans ("Jack", "friend") are labeled individually if it represents a crime class. For example, the following sentence shows that victim and suspect are labeled separately due to the conjunction "and" (assuming Jack is the victim and John the suspect):

Crime role annotation: [John]$_{SUS}$ and [Jack]$_{VIC}$ [got]$_{V}$ into a fight.
PropBank annotation: [John and Jack]$_{Arg0}$ [got]$_{V}$ [into a fight]$_{Arg2}$.

Another rule that I consider relates to prepositions. If the PropBank argument span is a prepositional phrase (PP), the preposition is removed from the span. For example, the following annotation illustrates the prepositional exclusion:

Crime role annotation: [Jack]$_{VIC}$ was [shot]$_{V}$ by [John]$_{SUS}$.
PropBank annotation: [Jack]$_{Arg1}$ was [shot]$_{V}$ [by John]$_{Arg0}$.

The above-described rules to assess and determine the crime class span can be summarized in three steps:

1. If the PropBank argument span denotes a PP, the preposition should be removed from the argument span.
2. The resulting argument span should be annotated in full if the content word is part of a crime class mention **and** there is no conjunction within the argument span.
3. The resulting argument span should be annotated by sub-argument spans (for each crime class mention) if a conjunction in the argument span exists **and** the content word is part of a crime class mention or is related to a crime class mention through the conjunction.

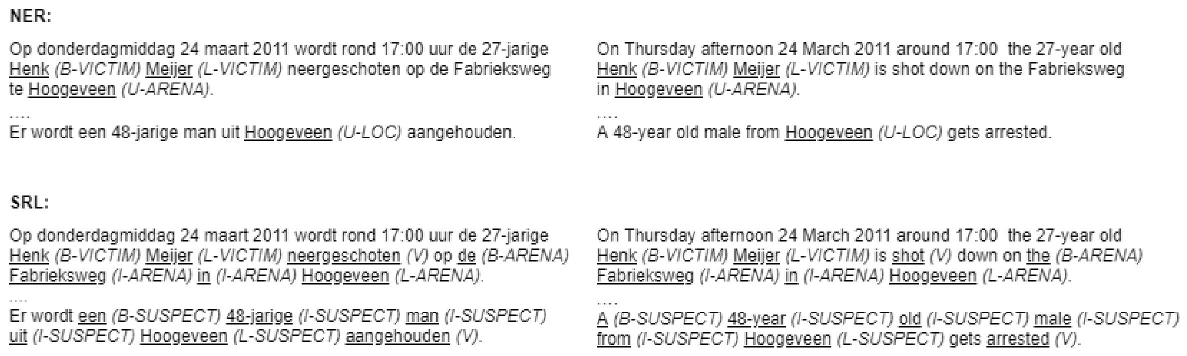
Additionally, one more adaption from the PropBank annotation guidelines is implemented. The PropBank guidelines prescribe the use of coreference chains for empty categories. These types of annotations involve references to arguments as long as these arguments are not part of any other argument span. However, the current research has few classes and more empty categories than PropBank. Consequently, the coreference annotations would exhibit a more complicated annotation structure than in PropBank. Furthermore, Schuurman et al. (2010) show that a semantic role corpus can successfully be built without this coreference property. Hence, to keep the annotation process efficient, the current research mostly disregards the coreference chains. Nonetheless, the coreferences are annotated for words that link a clause or phrase to a noun or pronoun in the same sentence. More specifically, the annotations include coreferences of relative pronouns and pronominal adverbs that link persons or locations. These relatively straightforward references can result is more informative crime class spans. In the sentence "The 40-year old male, who murdered his friend, was arrested", there are two predicates (i.e. murdered, arrested). The predicate "murdered" belongs to the clause "who murdered his friend". In this clause, there exists a SUSPECT-span "who". However, by including the relative pronoun coreferences, the coreference "The 40-year old male" is annotated for the predicate "murdered" as well.

Role class	Definition	Annotation	Count	Avg. per case
VICTIM	Any mention of the person(s) who got murdered.	Name Description Reference	2,881	5.6
SUSPECT	Any mention of the person(s) who has or have committed the homicide or who has or have been arrested at some point.	Name Description Reference	3,252	6.4
ARENA	Any mention of the location of the homicide where the mention is clearly linked to the homicide. Or, if no location of the homicide is present, any mention of the location where the body was found. Again, the mention is clearly linked to the finding place.	Location name (e.g. city name, region name) Description Reference	1,011	2.0

Table 3.2: Semantic role annotations for the homicide corpus

Following the aforementioned guidelines, the texts can be labeled as long as proper definitions for the crime classes are provided. Accordingly, Table 3.2 provides a detailed overview of crime role class definitions, annotation types, and corpus statistics. Notably, the definition of VICTIM and SUSPECT allow crime class mentions that describe multiple persons (e.g. "a group of people"). Furthermore, in contrast to the NER annotations, the regular person (non-victim and non-suspect) and regular location (non-arena) are not included. This exclusion reduces the time spent on the annotation process, while the most important classes (i.e. victim, suspect, arena) are maintained.

Based on the guidelines and role class definitions, the SRL annotation process can be initiated. Figure 3.4 shows an example of the SRL annotations as well as the NER annotations. Note that the SRL annotations detect the suspect mention, whereas the NER annotations fail to do so.


NER:

Op donderdagmiddag 24 maart 2011 wordt rond 17:00 uur de 27-jarige Henk *(B-VICTIM)* Meijer *(L-VICTIM)* neergeschoten op de Fabrieksweg te Hoogeveen *(U-ARENA)*.
....
Er wordt een 48-jarige man uit Hoogeveen *(U-LOC)* aangehouden.

On Thursday afternoon 24 March 2011 around 17:00 the 27-year old Henk *(B-VICTIM)* Meijer *(L-VICTIM)* is shot down on the Fabrieksweg in Hoogeveen *(U-ARENA)*.
....
A 48-year old male from Hoogeveen *(U-LOC)* gets arrested.

SRL:

Op donderdagmiddag 24 maart 2011 wordt rond 17:00 uur de 27-jarige Henk *(B-VICTIM)* Meijer *(L-VICTIM)* neergeschoten *(V)* op de *(B-ARENA)* Fabrieksweg *(I-ARENA)* in *(I-ARENA)* Hoogeveen *(L-ARENA)*.
....
Er wordt een *(B-SUSPECT)* 48-jarige *(I-SUSPECT)* man *(I-SUSPECT)* uit *(I-SUSPECT)* Hoogeveen *(L-SUSPECT)* aangehouden *(V)*.

On Thursday afternoon 24 March 2011 around 17:00 the 27-year old Henk *(B-VICTIM)* Meijer *(L-VICTIM)* is shot *(V)* down on the *(B-ARENA)* Fabrieksweg *(I-ARENA)* in *(I-ARENA)* Hoogeveen *(L-ARENA)*.
....
A *(B-SUSPECT)* 48-year *(I-SUSPECT)* old *(I-SUSPECT)* male *(I-SUSPECT)* from *(I-SUSPECT)* Hoogeveen *(L-SUSPECT)* gets arrested *(V)*.

Figure 3.4: Example with named entity labels and semantic role labels

SRL annotation procedure

Since the semantic roles cover more than just names, it is inefficient to annotate everything manually. Therefore, similar to the named entity annotation procedure, the semantic role annotation procedure follows two main steps: the automatic annotation and the manual annotation. First, the initial labels are assigned automatically according to five steps:

1. The POS-tagged verbs are identified, using the IXA-pos-tagger (Agerri et al., 2014). Each identified verb denotes the potential predicate of the sentence.
2. The sentences are duplicated in order to have one potential predicate per sentence.
3. The SoNaR training corpus (Schuurman et al., 2010) is prepared to make predictions. Furthermore, only Arg0, Arg1, and ArgM-LOC labels are considered, since the other arguments are less likely to contain a crime component.
4. A BERT-based model[4] (Zhangguoxioa, 2019) is trained on the SoNaR training corpus to predict the initial argument labels for the homicide corpus.
5. The Arg0 predictions are renamed to SUSPECT, the Arg1 predictions to VICTIM, and the ArgM-LOC predictions to ARENA.

After this automatic labeling procedure, the argument predictions are manually assessed and adjusted such that they fit the earlier defined SRL annotations guidelines. I perform these adaptions myself, using the online tool Doccano[5].

Finally, the resulting annotations are transformed according to the BILOU annotation format. This scheme deviates from the status-quo (e.g. PropBank, SoNaR), which uses the BIO scheme. However, it is expected that, as with NER (Ratinov and Roth, 2009), the more specific BILOU annotations improve the classification performance.

3.2.4 Case entities annotation

From the token-level named entity labels the case-level entity labels are deduced. The collection of these case-level entity labels describe victim(s), suspect(s) and arena(s) for each homicide case.

All of the cases have a clear full name mention (i.e. first name and last name) the victim. Therefore, every homicide case has at least one case entity label for the victim class. Furthermore, if there are several victims, a list of victim names is annotated. In regard to the suspect, the full name is rarely mentioned. Consequently, I apply a simple heuristic to determine the suspect case label. This heuristic takes the longest suspect name mention that appears in the case. For example, this ensures that the mention "Jack Smith" is labeled instead of "Jack S." or "Jack". When there is no suspect mention in the text at all, the case suspect label is assigned a null value. Furthermore, when a victim or suspect has a nickname. The nickname is annotated in addition to the normal name mention.

[4] `https://github.com/zhangguoxiao/bert-for-srl`
[5] `https://github.com/doccano/doccano`

Case-level entity class	**Definition**	**Annotation**	**Count**	**Avg. per case**	**# cases present**
VICTIM	A person who got murdered.	Full name Nickname	567	1.11	510
SUSPECT	A person who has committed the homicide or who has been arrested at some point.	Longest name that is present in the complete text Nickname	289	0.57	251
ARENA	Location of the homicide where the location is clearly linked to the homicide. Or, if no location of homicide is present, the location where the body was found. Again, the location is clearly linked to the finding place.	City name	490	0.96	488

Table 3.3: Case-level entity annotations for the homicide corpus

For the arena class, the city of homicide is labeled. However, if the city of homicide is not included in the text, the city where the body is found is annotated. In 22 cases the text does not mention any of these two options. For those cases, the arena value is given a null value.

Table 3.3 summarizes the annotations for the case-level entities. Notably, as presented in the table, in a subset of 57 cases (567 - 510) there are multiple victims. Furthermore, only 251 of the cases have a suspect annotation.

3.2.5 Case roles annotation

The case roles labels are selected from the token-level semantic role labels. The SRL labels are diverse, containing person names and city mentions as well as more generic descriptions and references. Each of these semantic roles potentially enriches the information. For instance, one of the roles might indicate that the suspect is a family member of the victim, whereas another can provide information about the age-group of the suspect. Therefore, it is decided to construct case-level roles based on a list of almost all the semantic role annotations of a case. Nonetheless, two types of semantic role mentions are excluded from the case-roles annotations: mentions that are fully contained in another mention and pronouns. These mentions are ignored because they are deemed to convey insufficient relevant information. Furthermore, when a crime class has no semantic roles for a particular case, the corresponding class is annotated as a null value for this case.

Table 3.4 shows an overview of the definitions, annotation types, and statistics concerning the case-level role annotations.

Case-level role class	**Definition**	**Annotation**	**Count**	**Avg. per case**	**# cases present**
VICTIM	The person(s) who got murdered.	Victim role that is not contained in another victim role and is no pronoun	1,001	1.96	510
SUSPECT	The person(s) who has or have committed the homicide or who has or have been arrested at some point.	Suspect role that is not contained in another suspect role and is no pronoun	1,125	2.21	466
ARENA	Location of the homicide where the location is clearly linked to the homicide. Or, if no location of homicide is present, the location where the body was found. Again, the location is clearly linked to the finding place.	Arena role that is not contained in another arena role and is no pronoun	825	1.62	490

Table 3.4: Case-level role annotations for the homicide corpus

3.2.6 Train-test split

The final preprocessing step splits the data into a training, development, and test set. Both the NER and SRL modules use the training set to update the weights of the model. Subsequently, the development set is utilized to assess models during the optimization processes. Finally, the test set is used to evaluate the NER and SRL approaches. In addition, the case-level entity and case-level role predictions are evaluated on the test set as well.

The homicide corpus consists of 510 cases with a total number of 4,454 sentences. Importantly, the cases need to remain intact such that case-level entities and roles can be extracted in the final phase. Furthermore, when cases are not intact, the sentences that belong to a case can be scattered across the different data sets (i.e. train, development, and test set). Evidently, this could cause data leakage from the train to the development or test set. Nevertheless, the NER and SRL models are trained on sentences, so the actual number of sentences in the train, development, and test set should be considered as well. Therefore, this thesis strives to have approximately has 75% of the sentences in the training set, 15% in the development set, and 10% in the test set. However, the exact split-percentages slightly differ such that the homicide cases are kept intact. A summary of the number of sentences, the number of cases and the number of annotations in the train, development, and test set are shown in Table 3.5.

	# Sentences	# Cases	# Named entities	# Semantic roles	# Case entities	# Case roles
Train	3,338 (74.9%)	383 (75.1%)	2,691	5,297	1,001	2,202
Development	662 (14.9%)	82 (16.1%)	571	1,155	220	470
Test	454 (10.2%)	45 (8.8%)	333	692	125	279
Total	4,454 (100%)	510 (100%)	3,595	7,144	1,346	2,951

Table 3.5: Train-dev-test split

3.3 Named Entity Recognition module

After the data is collected, filtered and annotated, it is utilized by the NER or SRL module. In this section the approaches of NER module are delineated. In regard to this NER module, I implement five experimental BERT architectures and a baseline model. In subsection 3.3.1 the the fine-tuned BERT architectures are delineated. Subsequently, subsection 3.3.2 discusses the architectures that use feature-based BERT. Furthermore, I describe the baseline architecture in subsection 3.3.3. Finally, the model parameter configurations are outlined in subsection 3.3.4. To summarize, the following architectures are implemented:

- BERT fine-tuned + Softmax
- BERT fine-tuned + CRF
- BERT feature-based + BiLSTM + Softmax
- BERT feature-based + BiLSTM + CRF
- BERT feature-based + CRF
- FastText + BiLSTM + CRF (baseline from Yadav et al. (2018))

The experiments as described in this section help to discover the best NER model architecture for the homicide corpus. In order to run the BERT experiments, I modify publicly available code[6] from Lu (2020), who apply the HuggingFace implementation of BERT (Wolf et al., 2019). In contrast, for the FastText baseline model, the official code[7] from Yadav et al. (2018) is used. Furthermore, to train all the models I use Google Colab with a Tesla P100-PCIE-16GB GPU.

3.3.1 Fine-tuned BERT

The first experiments are related to the fine-tuned BERT model. Figure 3.5 visualizes the fine-tuned BERT architecture that is used in this thesis. It shows that the words are tokenized into WordPiece tokens first. Subsequently, the BERT encoder learns the BERT representations. Then, the representations are inputted to a dense layer followed by the classification layer. Note that, in contrast to Devlin et al. (2018), CRF as well as Softmax classification layers are applied. Finally, the BILOU labels for the crime classes are derived from the classification layer.

[6] `https://github.com/Louis-udm/NER-BERT-CRF`
[7] `https://github.com/vikas95/Pref_Suff_Span_NN`

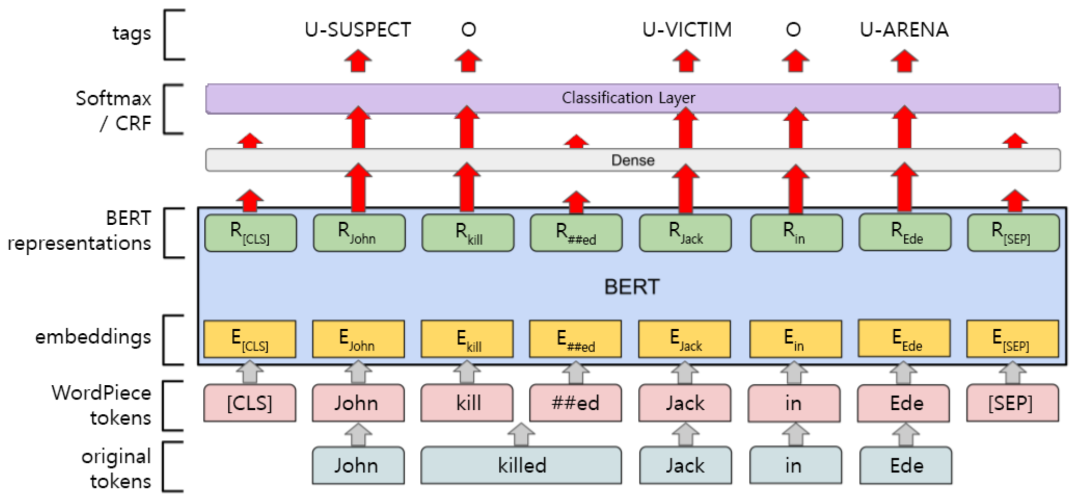


Figure 3.5: Model architecture for fine-tuned BERT. The red arrows indicate where weight updates take place, whereas the grey arrows visualize the input representations. Note that the actual model has Dutch texts. The English text in the figure is just used for illustrative purposes. The visualization is adapted from Arkhipov et al. (2019).

Classification layer

Softmax In line with the best performing BERT implementation from Devlin et al. (2018), the current research applies fine-tuned BERT with Softmax classification. This Softmax function uses inputs from the dense layer to calculate the most likely class of the input word. The following formula shows the Softmax calculation, where x_i represents the dense layer value for a NER class label i:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j))}$$

CRF Based on prior literature (Reimers and Gurevych, 2017), deep learning models with a CRF classification layer mostly outperform models with a Softmax classification layer. Hence, CRF is often the preferred method of implementation (e.g. Souza et al. (2019)). Still, the original implementation of BERT (Devlin et al., 2018) uses Softmax, with close to state-of-the-art results. Hence, both CRF and Softmax classification layers are applied. In contrast to Softmax, the CRF layer takes the sequence of the NER labels into account. The CRF-formula below shows the calculation of the probability of labels $\bar{y}$ based on dense representations $\bar{x}$. Furthermore, f_j denotes a feature function that represents a learned pattern between sequential y-labels. Subsequently, this value is multiplied by a weight w_j.

$$p(\bar{y}|\bar{x}) = \frac{exp(\sum_i \sum_j w_j f_j(y_{i-1}, y_i, \bar{x}, i))}{\sum_{y' \in Y} exp(\sum_i \sum_j w_j f_j(y'_{i-1}, y'_i, \bar{x}, i))}$$

3.3.2 Feature-based BERT

Existing literature shows that feature-based BERT models only slightly underperform compared to fine-tuned BERT (Devlin et al., 2018; Peters et al., 2019). Furthermore, even though fine-tuned BERT implementations have shown to improve on feature-based results (Devlin et al., 2018; Peters et al., 2019), state-of-the-art performances are achieved with BERT as input features (Straková et al., 2019). These contradicting results motivate the decision to experiment with fine-tuning as well as feature-based approaches. The various feature-based models of this thesis differ with respect to the classification layer (i.e. Softmax or CRF), as well as the number of BiLSTM layers (i.e. one, two, or no BiLSTM). A general overview of the model is shown in Figure 3.6.

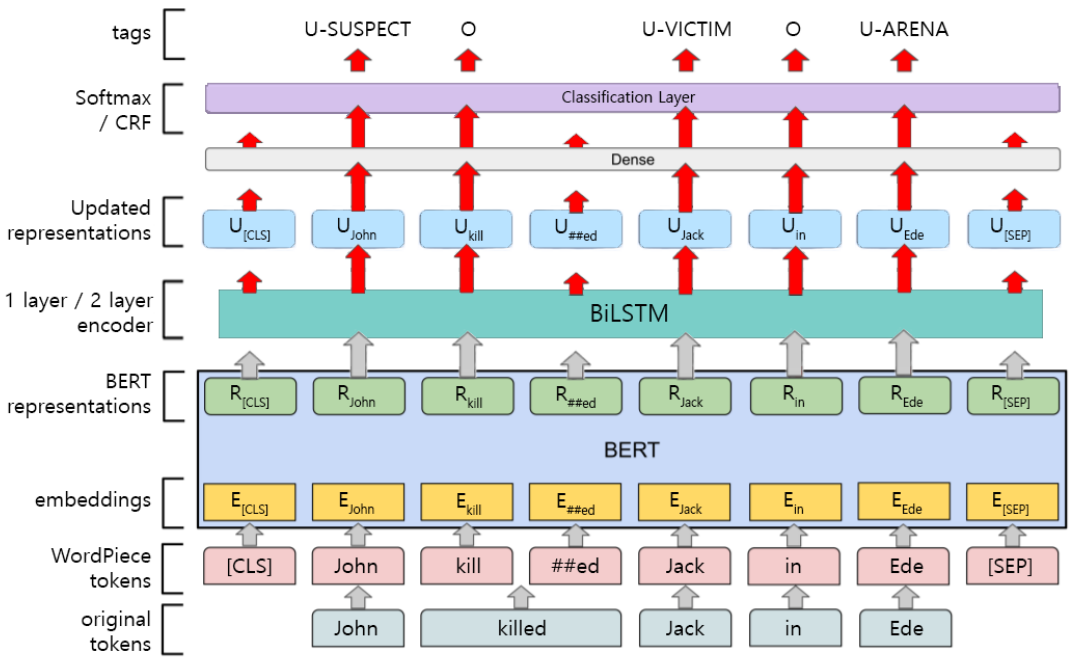

tags
Softmax / CRF
Updated representations
1 layer / 2 layer encoder
BERT representations
embeddings
WordPiece tokens
original tokens
U-SUSPECT
O
U-VICTIM
O
U-ARENA
Classification Layer
Dense
$U_{[CLS]}$ U_{John} U_{kill} $U_{\#\#ed}$ U_{Jack} U_{in} U_{Ede} $U_{[SEP]}$
BiLSTM
$R_{[CLS]}$ R_{John} R_{kill} $R_{\#\#ed}$ R_{Jack} R_{in} R_{Ede} $R_{[SEP]}$
BERT
$E_{[CLS]}$ E_{John} E_{kill} $E_{\#\#ed}$ E_{Jack} E_{in} E_{Ede} $E_{[SEP]}$
[CLS] John kill ##ed Jack in Ede [SEP]
John killed Jack in Ede

Figure 3.6: Model architecture for feature-based BERT. The red arrows indicate where weight updates take place, whereas the grey arrows visualize the input representations. Note that the actual model has Dutch texts. The English text in the figure is just used for illustrative purposes. The visualization is adapted from Arkhipov et al. (2019).

BiLSTM

Straková et al. (2019) and Souza et al. (2019) both use the BiLSTM-CRF implementation as proposed by Lample et al. (2016). Their model uses one layer of BiLSTM and an additional fully connected hidden layer that is positioned just before the CRF layer. Even though Straková et al. (2019) and Souza et al. (2019) both use BERT features, Straková et al. (2019) combine the BERT features with character-level and sub-word-level embeddings as well as with other contextualized embeddings (i.e. ELMo and Flair embeddings). Nevertheless, to enable a fair comparison between the feature-based and fine-tuned models, I follow the approach from Souza et al. (2019), who only use BERT features to initialize the model with.

In contrast to Straková et al. (2019) and Souza et al. (2019), Devlin et al. (2018) applies a two-layer BiLSTM network for the feature-based approach. Therefore, this thesis experiments with both the two-layered BiLSTM as well as the one-layered BiLSTM.

In addition, models with no BiLSTM layers are implemented as well. Most prior research (Devlin et al., 2018; Souza et al., 2019; Straková et al., 2019) do not consider this configuration. Still, I experiment with this architecture such that performance differences between fine-tuned and feature-based models can be examined more extensively.

Classification layer

Softmax As with the fine-tuned BERT models, the Softmax function is used to classify the entities for some of the feature-based models. Note, however, that the Softmax layer is only reasonable with a BiLSTM encoder in place. If there is no BiLSTM, the Softmax layer would classify straight from the BERT features. As a consequence, such a model is not able to learn task-specific patterns. Therefore, the Softmax layer is only implemented when a BiLSTM encoder is applied.

CRF Recent literature achieves state-of-the-art results for Dutch NER using feature-based BERT with a BiLSTM encoder and a CRF classification layer (Straková et al., 2019). Similarly, Souza et al. (2019) apply this structure to Portuguese texts as well. In line with these prior works, this thesis implements feature-based BERT models with a CRF classification layer.

3.3.3 FastText + BiLSTM + CRF

Finally, I compare a baseline model with the BERT-based models. As a baseline model, I apply the NER model from Yadav et al. (2018). Their model is a proper baseline since the model obtained state-of-the-art performance before the advent of BERT. The researchers initialize their model with FastText embeddings and use a one-layer BiLSTM encoder with a CRF layer as a decoder. To resemble the implementation from Yadav et al. (2018), I use the same hyperparameter settings. Only the maximum number of epochs is adjusted from 150 to 100 to preserve time. Consequently, this baseline model is not subjected to any hyperparameter optimization. As mentioned earlier, I use the published code from researchers[8] Yadav et al. (2018) for the current implementation.

3.3.4 Model optimization

Many of the existing BERT studies use the Adam optimizer (Devlin et al., 2018; Peters et al., 2019; Straková et al., 2019) to update model weights. This thesis does not deviate from the status quo, and thus uses the Adam optimizer as well. Nonetheless, some of the hyperparameters are experimented with. Typically, the learning rate, number of epochs, batch size, dropout ratio, or weight decay are adapted during model experimentation (Devlin et al., 2018; Peters et al., 2019). However, due to time constraints, I only consider the learning rate and dropout ratio between encoder and decoder.

[8] `https://github.com/vikas95/Pref_Suff_Span_NN`

Learning rate In line with Devlin et al. (2018), this thesis uses the learning rates 5e-5 and 2e-5 for BERT. Additionally, for other layers (i.e. BiLSTM and CRF) the default Adam learning rate of 0.001 is applied. This learning rate corresponds with the learning rate that Souza et al. (2019) and Straková et al. (2019) use as well.

Dropout Concerning the dropout rate between encoder and decoder, both 0.1 and 0.5 are experimented with. Devlin et al. (2018) apply a dropout rate of 0.1 for the whole BERT model and after the BERT encoder. In contrast, Straková et al. (2019) and Souza et al. (2019) use a dropout rate of 0.5 after the BERT representation layer. Therefore, both 0.1 and 0.5 dropout ratios are applied. Importantly, similar to Devlin et al. (2018), the dropout rate in between the 12 BERT layers stays at 0.1.

Batch size The batch size is 32 for all the models. This choice is based on Devlin et al. (2018), who use a batch size of 16 or 32. Furthermore, Masters and Luschi (2018) show that batch sizes of 32 or smaller generally achieve the best performance in deep learning models. The current thesis does not experiment with batch sizes smaller than 32, in order to limit the computation time.

Maximum number of epochs The maximum number of epochs is predetermined as well. The authors that introduce BERT (Devlin et al., 2018), use a maximum of four epochs for their fine-tuned models. In contrast, Souza et al. (2019) obtain the best results with 15 epochs for the fine-tuned models and 50 for the feature-based models. This thesis replicates the values from Souza et al. (2019). Importantly, however, these epoch values only function as a maximum. Consequently, the final model for each BERT configuration is trained on at most 15 epochs for the fine-tuned models and at most 50 epochs for the feature-based models.

Maximum sequence length The BERT model allows for a maximum of 512 input tokens. Nonetheless, for the homicide corpus, there is no need to use all 512 tokens since most sentences have far fewer BERT tokens, Therefore, all the experimental BERT models in this thesis have maximum sequence length of 128 BERT tokens. This is the same that Sun et al. (2019) use for their BERT model. Still, one sentence in the training data, with 150 BERT tokens, surpasses this threshold. For this instance, the sentence is truncated at the end, such that the first 126 WordPieces, the classification token ([CLS]), and the separation token ([SEP]) remain.

Remaining parameters The remaining hyperparameter values are kept at a constant value to reduce the number of experiments. Most of them are replicated from Devlin et al. (2018). Hence, the same weight decay value of 0.01 is applied to BERT and non-BERT layers. Furthermore, the same β_1 (0.9) and β_2 (0.999) are used. However, the number of warmup steps is reduced compared to Devlin et al. (2018) since the homicide corpus is smaller than the corpora that Devlin et al. (2018) work with. Hence, similar to Sun et al. (2019), a warmup proportion of 0.1 is applied. Since the fine-tuned models have at most 2,100 training steps, there are 210 warmup steps. Furthermore, the feature-based models have a maximum of 7,350 training steps, and hence 735 warmup steps.

To summarize, there are four different hyperparameter settings for the two fine-tuned BERT architectures (combinations of dropout and the learning rate). Similarly, there are four settings for the two feature-based architectures with a BiLSTM encoder (combinations of dropout and the number of layers). Finally, there are two dropout settings for the feature-based BERT with CRF architecture (and without a BiLSTM). Consequently, there are eighteen different BERT model configurations. Additionally, the FastText-based model is implemented as well. A complete overview of the models and parameter configurations is presented in Table 3.6. Furthermore, since neural networks are inherently stochastic, each run with the same parameters could lead to different results. Therefore, to make the predictions more robust, I run each of the nineteen models four times.

Architecture	**Parameter configuration**		
	Dropout*	Learning rate**	# BiLSTM-layers
BERT fine-tuned + Softmax	0.5	5e-5	NA
BERT fine-tuned + Softmax	0.5	2e-5	NA
BERT fine-tuned + Softmax	0.1	5e-5	NA
BERT fine-tuned + Softmax	0.1	2e-5	NA
BERT fine-tuned + CRF	0.5	5e-5	NA
BERT fine-tuned + CRF	0.5	2e-5	NA
BERT fine-tuned + CRF	0.1	5e-5	NA
BERT fine-tuned + CRF	0.1	2e-5	NA
BERT feature-based + BiLSTM + Softmax	0.5	NA	1
BERT feature-based + BiLSTM + Softmax	0.5	NA	2
BERT feature-based + BiLSTM + Softmax	0.1	NA	1
BERT feature-based + BiLSTM + Softmax	0.1	NA	2
BERT feature-based + BiLSTM + CRF	0.5	NA	1
BERT feature-based + BiLSTM + CRF	0.5	NA	2
BERT feature-based + BiLSTM + CRF	0.1	NA	1
BERT feature-based + BiLSTM + CRF	0.1	NA	2
BERT feature-based + CRF	0.5	NA	NA
BERT feature-based + CRF	0.1	NA	NA
FastText + BiLSTM + CRF (Yadav et al., 2018)	0.55	NA	1

* dropout rate between BERT and its successive layer

** learning rate of the BERT encoder

Table 3.6: All NER model configurations

3.4 Semantic Role Labeling module

Since NER is the core focus of this thesis, the experiments of the SRL module are limited. Hence, I only use two BERT models and do not implement a baseline model. The choice of the two models is based on the intermediate evaluation of the NER models. Since the feature-based BERT with two BiLSTM layers and a Softmax layer performs best on NER, this model configuration is applied for SRL as well. Subsequently, I implement the fine-tuned BERT with CRF to examine the performance of the opposite architecture (fine-tuned is the opposite of feature-based and CRF is the opposite of Softmax). This provides insights on whether the SRL module exhibits similar patterns as the NER module.

The code of both experiments is inspired by publicly available code[9] from Zhangguoxioa (2019). This is the same code that was used for the initialization of SRL labels, as described in subsection 3.2.3. Below, the model configurations are shown in more detail. Furthermore, Table 3.7 summarizes the two model configurations.

BERT feature-based + BiLSTM + Softmax The best performing NER model is the feature-based BERT model with a Softmax classification layer, where the dropout rate is equal to 0.5 and two BiLSTM-layers are inserted before the Softmax-layer. Therefore, this same model structure is applied for the SRL task. In contrast to the NER model, however, the number of epochs is reduced from 50 to 25 for time efficiency purposes. Furthermore, the model is evaluated after every 5 epochs instead of every epoch.

BERT fine-tuned + CRF The second SRL experiment utilizes the BERT fine-tuned model with a CRF layer. Furthermore, since the best performing NER model with this architecture has a dropout rate of 0.5 and a learning rate of 5e-5, the same hyperparameter configuration is applied for the SRL model. Nevertheless, whereas the NER model is evaluated at every epoch, the SRL model is evaluated after every 3 epochs to preserve time.

Architecture	**Parameter configuration**		
	Dropout*	Learning rate**	# BiLSTM-layers
BERT fine-tuned + CRF	0.5	5e-5	NA
BERT feature-based + BiLSTM + Softmax	0.5	NA	2

* dropout rate between BERT and its successive layer

** learning rate of the BERT encoder

Table 3.7: All SRL model configurations

[9]https://github.com/zhangguoxiao/bert-for-srl

3.5 Case-level classification

Subsequent to the NER module and the SRL module the token-level predictions are translated into case-level predictions. These case-level predictions enable the police to quickly access relevant information about a case (i.e. victim, suspect, arena). Nevertheless, note that this thesis prioritizes the experimentation with the NER module. Therefore, I construct just simple rule-based algorithms to infer the case-level labels. In this section, I first explain Algorithm 1 (subsection 3.5.1), which provides the case-level entity predictions based on the NER predictions. Subsequently, Algorithm 2 is defined (subsection 3.5.2), which infers the case-level roles using the SRL predictions.

3.5.1 Case entities inference (Algorithm 1)

The input for the case-level entity inference algorithm consists of word tokens and their corresponding token-level BILOU labels. Importantly, since only the victim, suspect, or arena classes are of interest at a case level, the NER predictions of regular persons and regular locations are discarded. Consequently, the output of the algorithm contains a list of predictions for each of the three crime classes. However, these predictions need to adhere to three properties.

1. Similar to the labeling process (subsection 3.2.4), there should be only one prediction for each unique victim, suspect, or arena. Therefore, when multiple entity mentions refer to the same person or location, only one of these mentions should be selected.

2. Different names need to be distinguished when several victims, suspects, or arenas are present.

3. The three classes are disjoint (e.g. a suspect cannot be a victim). Consequently, the entity inference algorithm needs to present disjoint predictions as well.

To account for the first property, the proposed algorithm considers entity mentions to refer to the same person or location when the mention contains another mention of the same crime class (e.g. ”John” is contained in ”John C.”). Sometimes, however, a mention is not fully contained in the entity mention, even though they refer to the same person or location (e.g. ”J.C.” is not contained in ”John C.”). Therefore, the entity mentions with the same capitalized letters are assumed to refer to the same person or location as well. Naturally, the algorithm only checks the capitalized letters if the entity mention has at least two words or is an abbreviation. All the remaining mentions of a crime class are assumed to refer to a different person or location. Hence, this rule-based algorithm satisfies the first two properties.

Finally, names that are classified to be both suspect and victim need special care. For these instances, I consider two basic approaches: (1) identify them as a victim only, or (2) identify them as a suspect only. Based on intermediate results on the development set, the second option is selected.

The following pseudo-code summarizes the rule-based entity inference algorithm that satisfies the three properties. This algorithm is referred to as Algorithm 1.

```
Data: NER predictions
Result: Case-level entity predictions
for each case do
    for class in VICTIM, SUSPECT, ARENA do
        append raw entity predictions with list of NER predictions for class;
    end
    for raw entity predictions do
        if entity mention contains another entity mention of the same class then
            select longest of these entity mentions;
        else if entity mention consists of multiple words or is an abbreviation and the
          capitalized letters of the entity mention are equal to capitalized letters of another
          entity mention of the same class then
            select longest of these entity mentions;
        else
            select the entity mention
        end
    end
    if entity mention belongs both to the victim and suspect class then
        remove entity mention from victim class;
    end
end
```

Algorithm 1: Inference of case-level entities

3.5.2 Case roles inference (Algorithm 2)

The case-roles inference algorithm utilizes the token-level predictions from the SRL module. In contrast to the entities, there can be multiple useful descriptions that are captured by the semantic roles. Hence, the case-roles inference algorithm should be able to predict multiple roles for a victim, suspect, or arena class. Furthermore, in contrast to the named entities, the semantic roles of the three crime classes are not necessarily disjoint. For instance, the role "a man" can be assigned to both a victim and a suspect. Consequently, the properties of the case entities inference algorithm do not necessarily hold for the case roles inference algorithm. Therefore, similar to the case roles annotations as described in subsection 3.2.5, the semantic role predictions are filtered according to two steps.

First, predicted semantic roles that are contained in other semantic role mentions from the same crime class are discarded. Second, the semantic role is discarded when it equals a pronoun. This rule-based algorithm can be simply described according to the pseudo-code below. This algorithm is referred to as Algorithm 2.

```
Data: SRL predictions
Result: Case-level role predictions
for each case do
    for class in VICTIM, SUSPECT, ARENA do
        append raw role predictions with list of SRL predictions for class;
    end
    for raw role predictions do
        if role mention contains another role mention of the same class then
            remove shortest of these mentions from the list of role mentions;
        end
        if role mention is not a pronoun then
            select the role mention;
        end
    end
end
```

Algorithm 2: Inference of case-level roles

3.6 System Evaluation

There are four evaluation steps in this thesis. The first two steps evaluate the NER and SRL module. Subsequently, the last two steps evaluate the case-level entity predictions after Algorithm 1 as well as the case-level role predictions from Algorithm 2. In this section, the evaluation metrics are delineated for each of these four systems.

3.6.1 NER evaluation

First, the performances of the NER models are evaluated. In line with other research in the field of named entity recognition (Devlin et al., 2018; Straková et al., 2019; Yadav et al., 2018), the micro F1-score is used to evaluate the performance on. Importantly, to compute the F1-score of a model, the precision and recall need to be calculated first. These three metrics are defined according to the following formulas:

$$precision = \frac{True\ positives}{True\ positives + False\ positives}$$

$$recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

To execute these computations, I apply the Python seqeval package[10]. This package calculates

[10]`https://github.com/chakki-works/seqeval`

the metric scores for each of the entity classes (i.e. victim, suspect, arena, person, location). Nevertheless, since the native seqeval package does not offer BILOU support, I manually adapt the package to enable the evaluation of BILOU labels.

There are three aspects are evaluated for the NER models. First of all, I focus on the general performances of the five BERT architectures and the baseline (section 3.3) and compare them with each other. Second, the effect of the hyperparameter settings for each of the architectures are compared. The evaluation of both these two aspects helps to deduce the best performing model configuration. As a consequence, SRQ2 can be answered accordingly.

Finally, I evaluate the metric scores for the specific entity classes as well. This allows me to assess the extent to which the NER models can successfully extract the crime classes. Based on these assessments, SRQ1 and SRQ2 can be answered.

3.6.2 SRL evaluation

Second, similar to the NER evaluation, SRL is evaluated using the precision, recall, and micro F1 metrics from the seqeval package. Again, the seqeval package is adapted to support the evaluation of BILOU annotations. This time, however, only the three crime classes (i.e. victim, suspect, arena) are evaluated, since the regular person and regular location are not annotated for SRL. Moreover, I do not experiment with hyperparameters for SRL, so the different parameter settings are not evaluated. Nevertheless, I still evaluate the results of two BERT architectures (section 3.4) and the performances for the crime classes. Consequently, I can provide an answer to SRQ3.

3.6.3 Case entities evaluation

Third, the case-level entity predictions are evaluated with respect to the crime classes as well. However, different from the token-level predictions, the F1-score is not used to evaluate teh case-level predictions. Instead, the accuracy per case is calculated for each of the three classes. The accuracy is defined according to the following formula:

$$accuracy = \frac{True\ positives + True\ negatives}{Total}$$

Subsequently, the final accuracy score of a crime class is computed by averaging the case-level scores. This final mean accuracy score reflects the proportion of the correctly identified victim, suspect, and arena entities across all the cases.

Besides using just the mean accuracy, the results should allow for more extensive evaluation that allows for error analysis as well. Therefore, the numbers of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are evaluated as well. Note, however, that the algorithm predicts string values (person and location names). Therefore the definitions of ”positive” and ”negative” as used in typical classification evaluations do not apply. Therefore, positive instances are defined to be instances that have some entity value (e.g. victim or suspect name) and negative instances are defined to be instances where there is no value at all (e.g. if no suspect is present in the text). More specifically, the measures can be defined as defined as follows:

$$\begin{aligned}
TP &= \textit{predicted value and true value are the same entity value} \\
TN &= \textit{predicted value and true value are a null value} \\
FP &= \textit{predicted value is an entity value while the true value is a null value or a different} \\
&\quad \textit{entity value} \\
FN &= \textit{predicted value is a null value while the true value is an entity value}
\end{aligned}$$

Finally, the algorithm is evaluated on the exact accuracy as well. The exact accuracy represents the accuracy where an entire crime class is correctly classified for a specific case. For example, consider a case with victims Jack and Dennis. If the algorithm classifies only Jack to be the victim, the normal accuracy is 50% whereas the exact accuracy is 0%. Using this exact accuracy on top of the regular accuracy helps to provide a better understanding of the usefulness of the algorithm. Hence, for each homicide case the following calculation is performed per crime class:

$$exact\ accuracy = \begin{cases} 1 & all\ true\ values = all\ predicted\ values \\ 0 & otherwise \end{cases}$$

Overall, these metrics help to evaluate the case-level entity predictions. Accordingly, they provide part of the answer to the MRQ.

3.6.4 Case roles evaluation

Finally, the SRL system (i.e. SRL-module combined with Algorithm 2) is evaluated based on the mean accuracy, exact accuracy as well. Moreover, I assess the errors of the system based on true positives, true negatives, false positives, and false negatives. However, this time the predicted values and true values are semantic roles instead of named entities. Therefore, the accuracy for a particular case denotes the proportion of the correctly labeled semantic role labels compared to the wrongly or not assigned semantic role labels. Hence, the evaluations regarding the case-level role predictions help to answer the MRQ.

Chapter 4

Results

This chapter presents the results in four distinct sections. First, the performances of the NER architectures are assessed in section 4.1. Subsequently, the evaluation results of the SRL module are provided in section 4.2. The third evaluation involves the case-level entity predictions from Algorithm 1 (section 4.3). Finally, the results of the case roles from Algorithm 2 are evaluated in section 4.4.

4.1 NER module results

For each NER architecture, the different parameter configurations are evaluated. In this section, an overview of the results is revealed first (subsection 4.1.1). Subsequently, the results of the separate NER architectures are presented with more detailed information about the performances of different hyperparameter settings and the metric scores for the entity classes. Furthermore, all the shown results represent the mean metric values of four different runs. The results for all the architectures, as discussed in section 3.3, are presented:

- BERT fine-tuned + Softmax (subsection 4.1.2)
- BERT fine-tuned + CRF (subsection 4.1.3)
- BERT feature-based + BiLSTM + Softmax (subsection 4.1.4)
- BERT feature-based + BiLSTM + CRF (subsection 4.1.5)
- BERT feature-based + CRF (subsection 4.1.6)
- FastText + BiLSTM + CRF (subsection 4.1.7)

4.1.1 NER results overview

The results for the NER architectures are summarized by Table 4.1. From the table, one can deduce that the feature-based BERT in combination with BiLSTM and Softmax layers achieves the highest metric scores (precision of 0.846, recall of 0.848, and F1-score of 0.847). Furthermore, the results show that the architectures with a Softmax classification layer outperform the architectures with a CRF layer. Also, the feature-based BERT models with a BiLSTM encoder achieve higher results than their fine-tuned counterparts. Nevertheless, feature-based BERT without a BiLSTM has the worst performance (F1-score of 0.677). Finally, the FastText-based baseline obtains the second-lowest F1-score (0.796).

All the displayed metric scores are from the best configuration of the corresponding NER architecture. For instance, feature-based BERT with BiLSTM and Softmax layers uses a dropout rate of 0.5 and two BiLSTM layers. Subsequent subsections delineate the results of the different parameter configurations for each architecture. Furthermore, a complete overview of test results for all configurations can be found in Appendix B (Table B.1). Additionally, Table B.2 in Appendix B provides all results on the development set as well.

Architecture	**Precision**	**Recall**	**F1**
BERT fine-tuned + Softmax	0.832	0.845	0.838
BERT fine-tuned + CRF	0.822	0.837	0.829
BERT feature-based + BiLSTM + Softmax	**0.846**	**0.848**	**0.847**
BERT feature-based + BiLSTM + CRF	0.821	0.847	0.834
BERT feature-based + CRF	*0.644*	*0.715*	*0.677*
FastText + BiLSTM + CRF (Yadav et al., 2018)	0.802	0.791	0.796

italics = worst score
bold = best score

Table 4.1: Results for the NER architectures on the test set

4.1.2 NER results of BERT fine-tuned + Softmax

For the fine-tuned BERT encoder with a Softmax classification layer the different dropout and learning rates are compared in Table 4.2.

Parameter configuration		**Precision**	**Recall**	**F1**
Dropout	Learning Rate			
0.5	5e-5	*0.799*	*0.820*	*0.809*
0.5	2e-5	0.820	0.842	0.831
0.1	5e-5	0.815	0.844	0.829
0.1	2e-5	**0.832**	**0.845**	**0.838**

italics = worst score
bold = best score

Table 4.2: NER results of parameter configurations for BERT fine-tuned + Softmax on the test set

As shown in Table 4.2, this architecture achieves the best results with a dropout value of 0.1 and a learning rate of 2e-5 (F1-score of 0.838). Hence, Table 4.3 presents the results of this parameter configuration for each entity class. The table shows that the victim and arena classes are extracted most successfully with very similar F1-scores (0.889 and 0.885 respectively).

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.867	0.911	0.889	183
SUSPECT	0.785	0.735	0.759	83
ARENA	0.819	0.963	0.885	41
PERSON*	0.050	0.083	0.063	3
LOCATION**	0.813	0.598	0.688	23
Total	0.832	0.845	0.838	333

* person who is not suspect or victim
** location that is not arena

Table 4.3: NER results of BERT fine-tuned + Softmax (dropout = 0.1, lr = 2e-5) on the test set

4.1.3 NER results of BERT fine-tuned + CRF

Similar to the previously discussed architecture, different dropout and learning rates are experimented with for the fine-tuned BERT model with a CRF layer (Table 4.4).

Parameter configuration		**Precision**	**Recall**	**F1**
Dropout	Learning Rate			
0.5	5e-5	**0.822**	**0.837**	**0.829**
0.5	2e-5	0.804	0.824	0.814
0.1	5e-5	0.813	0.830	0.821
0.1	2e-5	*0.795*	*0.812*	*0.803*

italics = worst score
bold = best score

Table 4.4: NER results of parameter configurations for BERT fine-tuned + CRF on the test set

From Table 4.4 the best parameter combination can be deduced. Accordingly, the NER architecture achieves the highest metric scores with a dropout of 0.5 and a learning rate of 5e-5 (F1-score of 0.829). Table 4.5 elaborates on the results of this model. This time, the arena class is extracted most effectively (F1-score 0.906).

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.861	0.889	0.874	183
SUSPECT	0.763	0.741	0.746	83
ARENA	0.847	0,976	0.906	41
PERSON*	0.000	0.000	0.000	3
LOCATION**	0.819	0.630	0.712	23
Total	0.822	0.837	0.829	333

* person who is not suspect or victim
** location that is not arena

Table 4.5: NER results of BERT fine-tuned + CRF (dropout = 0.5, lr = 5e-5) on the test set

4.1.4 NER results of BERT feature-based + BiLSTM + Softmax

In contrast to the fine-tuned BERT architectures, the learning rate is not adjusted for the feature-based BERT architectures. This follows from the fact that the BERT features are unaffected by learning rate changes. Nevertheless, the number of BiLSTM layers is experimented with. Table 4.6 presents the results for the different parameter settings when a Softmax classification layer is added on top of the feature-based BERT and BiLSTM.

Parameter configuration		**Precision**	**Recall**	**F1**
Dropout	# BiLSTM-layers			
0.5	1	0.833	0.839	0.836
0.5	2	**0.846**	**0.848**	**0.847**
0.1	1	*0.832*	*0.835*	*0.833*
0.1	2	0.844	0.842	0.843

italics = worst score
bold = best score

Table 4.6: NER results of parameter configurations for BERT feature-based + BiLSTM + Softmax on the test set

According to Table 4.6, the best model uses a dropout-rate of 0.5 and 2 BiLSTM layers in between the BERT-model and the Softmax layer (F1-score of 0.847). Table 4.7 explores the class-level scores of this particular model. In line with the previously evaluated model, the arena class obtains the highest performance (F1-score of 0.927).

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.875	0.889	0.882	183
SUSPECT	0.776	0.774	0.773	83
ARENA	0.864	1.000	0.927	41
PERSON*	0,750	0.250	0.375	3
LOCATION**	0.832	0.598	0.695	23
Total	0.846	0.848	0.847	333

* person who is not suspect or victim
** location that is not arena

Table 4.7: NER results of BERT feature-based + BiLSTM + Softmax (dropout = 0.5, BiLSTM layers = 2) on the test set

4.1.5 NER results of BERT feature-based + BiLSTM + CRF

For the feature-based BERT model with a BiLSTM encoder and a CRF classification layer, the results of different parameter settings are compared as well (Table 4.8).

Parameter configuration		**Precision**	**Recall**	**F1**
Dropout	# BiLSTM-layers			
0.5	1	0.788	0.825	0.806
0.5	2	**0.821**	**0.847**	**0.834**
0.1	1	*0.762*	*0.803*	*0.782*
0.1	2	0.803	0.841	0.821

italics = worst score
bold = best score

Table 4.8: NER results of parameter configurations for BERT feature-based + BiLSTM + CRF on the test set

Similar to the other feature-based architecture, the feature-based BERT with a CRF layer obtains best results (F1-score of 0.834) when the dropout proportion equals 0.5 and two BiLSTM layers are used (see Table 4.8). Therefore, the performance of the model with these parameter configurations is displayed in more detail in Table 4.9. Again, the arena class acquires the best F1-score (0.946), followed by the victim class (F1-score of 0.875).

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.865	0.885	0.875	183
SUSPECT	0.771	0.762	0.765	83
ARENA	0.897	1.000	0.946	41
PERSON*	1.000	0.333	0.500	3
LOCATION**	0.886	0.641	0.743	23
Total	0.821	0.847	0.834	333

* person who is not suspect or victim
** location that is not arena

Table 4.9: NER results of BERT feature-based + BiLSTM + CRF (dropout = 0.5, BiLSTM layers = 2) on the test set

4.1.6 NER results of BERT feature-based + CRF

The results of the final BERT-based model correspond to the feature-based BERT model with a CRF layer without a BiLSTM encoder. Since no BiLSTM layers are present, only the dropout proportion is experimented with. The results of the different dropout proportions are visualized in Table 4.10.

Parameter configuration		**Precision**	**Recall**	**F1**
Dropout	# BiLSTM-layers			
0.5	NA	**0.644**	**0.715**	**0.677**
0.1	NA	0.575	0.706	0.634

bold = best score

Table 4.10: NER results of parameter configurations for BERT feature-based + CRF on the test set

As can be deduced from Table 4.10, a dropout proportion of 0.5 results in a better performance (F1-score of 0.677). Therefore, Table 4.11 presents the entity-level scores for the feature-based BERT with a CRF classification layer with a 0.5 dropout proportion. The arena class scores high again (F1-score of 0.908). However, the victim and suspect classes drop in performance compared to the other models (approximately 10% for the victim class and 15% for the suspect class).

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.750	0.817	0.782	183
SUSPECT	0.666	0.515	0.581	83
ARENA	0.941	0.878	0.908	41
PERSON*	0.000	0.000	0.000	3
LOCATION**	0.886	0.424	0.574	23
Total	0.644	0.715	0.677	333

* person who is not suspect or victim
** location that is not arena

Table 4.11: NER results of BERT feature-based + CRF (dropout = 0.5) on the test set

4.1.7 NER results of FastText + BiLSTM + CRF

Finally, the FastText-based model of Yadav et al. (2018) with a BiLSTM and CRF layer is evaluated. Note that for this model, the parameters as described by Yadav et al. (2018) are replicated and no parameters are tuned. Therefore, this architecture only has one model. The class-level results of this model are shown in Table 4.12. Similar to the BERT models, the table shows that the arena class is most successfully extracted (F1-score of 0.89) for the FastText-based model as well.

Entity class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.836	0.857	0.846	183
SUSPECT	0.721	0.645	0.677	83
ARENA	0.847	0.939	0.890	41
PERSON[*]	0.000	0.000	0.000	3
LOCATION[**]	0.816	0.641	0.715	23
Total	0.802	0.791	0.796	333

[*] person who is not suspect or victim
[**] location that is not arena

Table 4.12: NER results of FastText + BiLSTM + CRF on the test set

4.2 SRL module results

Two semantic role labeling models are experimented with. As explained in section 3.4, these models represent the best NER model architecture (BERT feature-based + BiLSTM + Softmax) and the opposite architecture (BERT fine-tuned + CRF). At first, the summary of the results of these two architectures is provided in subsection 4.2.1. Subsequently, more detailed results of the two model configurations are delineated in the other sections. Again, the presented results of the SRL models are the aggregate of four separate runs.

4.2.1 SRL results overview

The performances of the two models are compared in Table 4.13. Both models use the same parameter configurations as their NER counterparts. Interestingly, the table shows that the fine-tuned BERT model with a CRF classification layer outperforms the alternative model (BERT feature-based + BiLSTM + Softmax) with a F1-score of 0.714 compared to 0.655 respectively.

Architecture	**Precision**	**Recall**	**F1**
BERT feature-based + BiLSTM + Softmax	0.634	0.677	0.655
BERT fine-tuned + CRF	**0.703**	**0.725**	**0.714**

bold = best score

Table 4.13: Overview of results for the SRL architectures on the test set

Similarly, the models can be evaluated on the development set as well. The results of this development set are summarized in Table 4.14. Again, the fine-tuned BERT with a CRF obtains the highest F1-score (0.73).

Architecture	**Precision**	**Recall**	**F1**
BERT feature-based + BiLSTM + Softmax	0.658	0.715	0.685
BERT fine-tuned + CRF	**0.707**	**0.754**	**0.730**

bold = best score

Table 4.14: Overview of results for the SRL architectures on the development set

4.2.2 SRL results of BERT feature-based + BiLSTM + Softmax

As shown in Table 4.1, the best performing NER model is a feature-based BERT model with 2 layers of BiLSTM layers and a Softmax classification layer. Furthermore, the model uses a 0.5 dropout rate. This same configuration is evaluated for the task of semantic role labeling. Table 4.15 presents the results of this SRL model. In contrast to the NER results, the arena class of the SRL model has the lowest score (F1-score of 0.58) compared to the other classes. Furthermore, the SRL model extracts the victim class most successfully (F1-score of 0.70).

Role class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.707	0.695	0.700	268
SUSPECT	0.655	0.685	0.670	325
ARENA	0.556	0.606	0.580	99
Total	0.634	0.677	0.655	692

Table 4.15: SRL results of BERT feature-based + BiLSTM + Softmax (dropout = 0.5, BiLSTM layers = 2) on the test set

4.2.3 SRL results of BERT fine-tuned + CRF

The other SRL model is the opposite of the first model with respect to the classification layer (CRF instead of Softmax) and the BERT approach (fine-tuned instead of feature-based). Furthermore, the parameters are configured such that they resemble the best performing NER model which uses fine-tuned BERT with a CRF. Consequently, a learning rate of 5e-5 and a dropout proportion of 0.5 is applied. These results are displayed in Table 4.16. Similar to the other SRL model, the results show that the arena class has a worse F1-score (0.63) than the victim (0.76) and suspect (0.71) classes.

Role class	**Precision**	**Recall**	**F1**	**Support**
VICTIM	0.745	0.779	0.761	268
SUSPECT	0.731	0.697	0.713	325
ARENA	0.592	0.672	0.629	99
Total	0.703	0.725	0.714	692

Table 4.16: SRL results of BERT fine-tuned + CRF (dropout = 0.5, lr = 5e-5) on the test set

4.3 Case entities results

Before applying Algorithm 1, I review the NER results on the development set. Based on these results (see Table B.2 in Appendix B), feature-based BERT with 2 BiLSTM layers and a Softmax classification layer (dropout = 0.5) is used to predict the token-level entities. From these initial token-level predictions Algorithm 1 infers the entities at a case-level. Table 4.17 and Table 4.18 show the case-level results. The presented results belong to the 45 homicide cases in the test set and are aggregated over four separate runs.

Table 4.17 displays the overall mean accuracy and exact accuracy. The exact accuracy represents the accuracy where the entire crime class is correctly classified for a specific homicide case. The table shows that the NER system identifies the crime classes with a mean accuracy of 0.87 and an exact accuracy of 0.79. Furthermore, the arena class is most accurately identified (0.93 mean accuracy and 0.90 exact accuracy).

Additionally, Table 4.18 shows the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), as defined in subsection 3.6.3.

Crime class	**Exact accuracy**	**Mean accuracy**
VICTIM	0.744	0.873
SUSPECT	0.717	0.819
ARENA	0.900	0.930
Total	0.787	0.874

Table 4.17: Algorithm 1 (case entities) accuracy results on the test set

Crime class	**TP**	**TN**	**FP**	**FN**	**# positive labels***	**# negative labels****
VICTIM	47.25	0	11.5	1	49	0
SUSPECT	30.5	16.25	11.25	1.5	35	19
ARENA	40	3.75	6.25	0	41	5
Total	117.75	20	29	2.5	125	24

* true non-null values
** true null values

Table 4.18: Algorithm 1 (case entities) error analysis on the test set

Three examples of case-level entity predictions can be found in Appendix C.

4.4 Case roles results

Algorithm 2 uses the generated token-level predictions from the SRL module to infer the case-level role labels. The fine-tuned BERT with a CRF classification layer (dropout = 0.5, learning rate = 5e-5) achieves the highest F1-score on the development set (see Table 4.14). Therefore, the predictions from this model configuration constitute the input for Algorithm 2. As explained in subsection 3.6.4, the mean accuracy, exact accuracy, and the number of true positives, true negatives, false positives, and false negatives are all evaluated. All the results for the 45 homicide cases in the test set are presented in Table 4.19 and Table 4.20. Similar to all prior evaluations, the results are the average of four runs.

Based on the results in Table 4.19, one can deduce that the mean accuracy equals 0.69 and the exact accuracy 0.47. Furthermore, the arena class has the lowest mean accuracy score (0.62) and the suspect class has the lowest exact accuracy score (0.42).

Crime class	**Exact accuracy**	**Mean accuracy**
VICTIM	0.511	0.750
SUSPECT	0.417	0.698
ARENA	0.467	0.622
Total	0.465	0.690

Table 4.19: Algorithm 2 (case roles) accuracy results on the test set

Crime class	**TP**	**TN**	**FP**	**FN**	**# positive labels**[*]	**# negative labels**[**]
VICTIM	57.5	0	29.5	6.25	87	0
SUSPECT	70.5	2.75	35	11.5	119	4
ARENA	42	0.75	32	6.5	73	2
Total	170	3.5	96.5	24.25	279	6

[*] true non-null values
[**] true null values

Table 4.20: Algorithm 2 (case roles) error analysis on the test set

Three examples of case-level role predictions can be found in Appendix C.

Chapter 5

Discussion

In this chapter, the results are discussed in more detail. First, in section 5.1, section 5.2, and section 5.3 the models' performances are interpreted by relating them to the sub research questions and prior research. Subsequently, I interpret the results concerning the main research question in section 5.4. Finally, section 5.5 delineates the limitations of the thesis and discusses potential future research.

5.1 Interpretation of NER results

There are two research questions relating the the NER module. First, I focus on SRQ1:

SRQ1: *To what extent can BERT be leveraged for **named entity recognition** on homicide-related Dutch texts?*

Based on the results, BERT for NER on the homicide corpus is promising for two main reasons. First, the BERT architectures generally outperform the FastText-BiLSTM-CRF (Yadav et al., 2018) baseline. Hence, the BERT models can be leveraged to a greater extent than the baseline model. Second, the F1-score (0.847) of the best model configuration (BERT feature-based + BiLSTM + Softmax) is higher than the scores of crime-related NER from prior works (Dasgupta et al., 2017; Schraagen et al., 2017).

As described in subsection 2.7.1, Dasgupta et al. (2017) use NER to extract crime-related named entities. The researchers obtain a high F1-score of 0.88 for the extraction of crime locations. Similarly, as shown in Table 4.7, the arena entity class has the highest F1-score (0.927) on the homicide corpus. Interestingly, the other model architectures (except fine-tuned BERT with Softmax) obtain the highest F1-score for the arena class as well. Therefore, in line with the results from Dasgupta et al. (2017), the results of this thesis imply that the arena (i.e. location of the crime) is relatively easy to extract. Even when the architectures are simple (feature-based BERT + CRF) or do not include BERT (FastText + BiLSTM + CRF), the arena extraction has good performances. Note that for two BERT architectures (feature-based BERT + BiLSTM + Softmax and feature-based BERT + BiLSTM + CRF) the recall of the arena class achieves the perfect score of 1.0, indicating that the NER system extracts the arena for all instances. Importantly, however, since the size of the test set is small, a perfect score like this is more likely to occur.

On the other hand, the extraction of the suspect class is the worst of the crime classes for all the model architectures. Especially the simple BERT architecture (feature-based BERT + CRF) and the baseline (FastText + BiLSTM + CRF) show low F1-scores (0.58 and 0.68 respectively). This is in line with the results from Dasgupta et al. (2017), who obtain their worst score on the extraction of the criminal name (F1-score of 0.64). Interestingly, however, the performances on the suspect class of the remaining BERT models are substantially better. The best model (feature-based BERT + BiLSTM + Softmax) even achieves a F1-score of 0.77. These results imply that the use of BERT allows for improvement regarding the extraction of the suspect class.

Still, one should note that the corpora from this thesis and Dasgupta et al. (2017) are different. Therefore, it remains difficult to compare results from the current homicide corpus and their crime corpus directly. Nonetheless, the precision and recall of the suspect class are interesting to compare, since Dasgupta et al. (2017) obtain a large discrepancy between the two measures. More specifically, their model has a precision of 0.93 and a recall of 0.49 for the criminal name extraction. For two of the implemented models from this thesis, such a pattern is present to a lesser extent as well. First, the feature-based BERT with only a CRF layer obtains a precision of 0.67 and a recall of 0.52 for the suspect class. Second, the FastText-based model has a precision of 0.72 and a 0.65 of recall for the suspect class. Nevertheless, most of the remaining BERT models obtain a similar precision and recall score. For instance, as shown in Table 4.7, the precision and recall scores of the suspect class for the best model (BERT feature-based + BiLSTM + Softmax) is almost equal (0.776 and 0.774 respectively). These results imply that the BERT-based models do not necessarily favor precision or recall for the suspect class. In contrast, the person and location classes show different results. Notably, however, there are relatively few mentions of these entity classes in the test set (23 location mentions and 3 person mentions). Therefore, it is difficult to make implications for these two entity classes. Moreover, implications regarding these classes are less relevant, since the corresponding entity mentions are unrelated to the homicide.

Consequently, most of the proposed BERT models in this thesis score well for the victim, suspect, and arena classes with respect to precision, recall, and F1-score. Especially the feature-based BERT with BiLSTM and Softmax layers performs well. Still, compared to state-of-the-art F1-scores of regular NER (0.934 on CoNLL-2003 and 0.927 on CoNLL-2002 (Straková et al., 2019)), the F1-score of 0.847 lacks behind. Nevertheless, as with the crime corpus from Dasgupta et al. (2017), direct comparisons with other corpora are not that insightful. For example, Souza et al. (2019) achieve a state-of-the-art F1-score of 0.742 on the Portuguese HAREM corpus (Santos et al., 2006) using similar model architectures that Straková et al. (2019) use. Furthermore, the current homicide corpus is relatively small (4,454 sentences in the training set) compared to a corpus such as the Dutch CoNLL-2002 (Tjong Kim Sang, 2002) or English CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) (15,806 and 14,987 sentences in the training set respectively). Based on prior research (Sun et al., 2019) it is expected that the proposed NER models will perform better when they are trained on more data.

5.2 Best BERT configuration for NER

In this section, I discuss the results concerning the second sub research question:

SRQ2: *What BERT configuration is favorable with respect to named entity recognition on homicide-related Dutch texts?*

5.2.1 The model architectures

The NER results, as presented in section 4.1, help to answer this question. Firstly, as can be deduced from Table 4.1, four of the five BERT models (F1-score between 0.829 - 0.847) outperform the baseline (F1-score of 0.796). These observations are in line with existing literature, where BERT-based models achieve state-of-the-art results (Souza et al., 2019; Straková et al., 2019).

Interestingly, however, the BERT architecture consisting of feature-based BERT with a CRF layer and without BiLSTM layers shows a substantially worse performance (F1-score of 0.677). Even though this score is lower than the scores of the other architectures, this pattern is to be expected. First of all, the literature that successfully implements feature-based BERT (e.g. Devlin et al. (2018), Straková et al. (2019)), only apply feature-based BERT in combination with a BiLSTM encoder. Moreover, prior to the advent of BERT, the combination of BiLSTM and CRF layers proved to be superior to a stand-alone CRF for NER on CoNLL-2003 (Huang et al., 2015). Still, the difference in performance between CRF and BiLSTM-CRF that Huang et al. (2015) find is considerably smaller (0.03 F1-score difference) than the difference in this thesis (0.16 F1-score difference). One of the reasons for this disagreement could be related to the homicide ontology. Namely, the results in Table 4.11 show that especially the victim and suspect classes score worse compared to the other architectures. This implies that more complex architectures such as a BiLSTM and fine-tuned BERT are necessary to distinguish between the victim and suspect classes. Another reason for the observation might be that the CRF layer is not as useful for NER on the homicide corpus as it has been on other corpora. As discussed in subsection 2.4.1, the CRF learns patterns based on the order of consecutive labels. However, in the homicide corpus there are relatively few labels in the training set (2,691 entities) compared to other corpora such as CoNLL-2003 (23,499 entities) or the HAREM corpus that Souza et al. (2019) use (5,017 entities). Hence, this property might make it difficult for a CRF layer to learn useful patterns. Furthermore, based on the results from prior research (Reimers and Gurevych, 2017; Souza et al., 2019) one would expect that a CRF layer outperforms Softmax layer for NER. In contrast, the results from Table 4.1 indicate that BERT configurations with a CRF layer score slightly lower than the same models with a Softmax classification layer. Again, this result can probably be attributed to the fact that there are too few NER labels in the homicide corpus that the CRF layer can learn patterns from.

Besides the classification layer, the BERT-approach (i.e. feature-based or fine-tuned) is experimented with as well. Existing research (Devlin et al., 2018; Peters et al., 2019; Souza et al., 2019) also apply both approaches. As described in subsection 2.6.1, these studies find that the fine-tuned approach is slightly superior to the feature-based BERT-models. In contrast, in this thesis, the feature-based approach (with BiLSTM layers) outperforms the fine-tuned approach on the homicide corpus. Even though this contradicts most prior research, it is not as surprising as one might think. First, existing research only finds minor differences between the two approaches (Devlin et al., 2018; Peters et al., 2019; Souza et al., 2019). Furthermore, the current state-of-the-art on CoNLL-2003 and CoNLL-2002 utilizes feature-based BERT as well (Straková et al., 2019). Hence, when to use fine-tuned or feature-based models is not evident. However, based on the results of this thesis, the feature-based models are more suitable for NER on the homicide corpus.

5.2.2 The model parameters

Other relevant observations of the BERT models concern the hyperparameter settings. For four of the five BERT architectures, the dropout rate of 0.5 achieves the best performances. Only the fine-tuned BERT model with a Softmax layer has the highest F1-score when the dropout rate equals 0.1. Interestingly, the benefits of a dropout rate of 0.5 are most apparent when a CRF layer is applied. Three observations that indicate this. First, the feature-based BERT with a CRF has the largest discrepancy (0.043 F1-score) between 10% dropout and 50% dropout (Table 4.10). Second, the fine-tuned models with Softmax score worse with 50% dropout (Table 4.2). Third, the feature-based BERT models with BiLSTM and Softmax layers and a dropout rate of 0.5 only marginally outperform the same models with a dropout rate of 0.1 (Table 4.6). Furthermore, these observations are in line with prior literature. Namely, similar to the second observation, Devlin et al. (2018) apply a 10% dropout for their fine-tuned Softmax model. Moreover, Straková et al. (2019) use a dropout rate of 0.5 for their CRF models. Nonetheless, both these studies as well as other prior research (Peters et al., 2019; Souza et al., 2019) have not published results for BERT models that concern dropout tuning. Hence, it remains unclear if the observations are specific to the homicide corpus or they are more universal.

In addition, the learning rate is tuned for the fine-tuned models. Interestingly, the fine-tuned model with a CRF layer scores better with a learning rate of 5e-5, whereas the fine-tuned model with a Softmax layer prefers a learning rate of 2e-5. Therefore, similar to the dropout rate, the optimal learning rate of a model depends on the classification layer.

Furthermore, the number of BiLSTM layers are adjusted for the feature-based models. For these models, a two-layer BiLSTM achieves the best results irrespective of the classification layer. Still, the difference in F1-score is most apparent when the CRF layer is applied (Table 4.8). Nevertheless, research with state-of-the-art performance for NER (Souza et al., 2019; Straková et al., 2019) combine a one-layer BiLSTM with a CRF layer. Notably, however, these studies do not experiment with the number of BiLSTM layers. Hence, the results of this thesis do not directly contradict results form Souza et al. (2019) or Straková et al. (2019).

5.3 Interpretation of SRL results

The main contribution of this thesis concerns the NER module. Nevertheless, results of the SRL module (section 4.2) help to answer the following research question:

SRQ3: *To what extent can BERT be leveraged for* ***semantic role labeling*** *on homicide-related Dutch texts?*

Compared to the NER results (best F1-score of 0.847), the SRL results show a lower F1-score (best of 0.714). This difference is in line with some of the prior literature. For example, Vossen et al. (2016) obtain a F1-score of 0.877 for NER on the SoNaR corpus while their SRL algorithm only achieves a F1-score of 0.740 on the same corpus. Nevertheless, more recent research (Ouchi et al., 2018; Shi and Lin, 2019) apply SRL more successfully with F1-scores close to 90% on CoNLL-2005 and thus nearing the state-of-the-art scores of NER. However, due to the use of different corpora, it remains difficult to compare the performances directly.

On the homicide corpus, I apply two BERT-based models for SRL: fine-tuned BERT + CRF and feature-based BERT + BiLSTM + Softmax. As shown in Table 4.13, the former scores substantially better than the latter model (F1-score of 0.714 and 0.655 respectively). These results are opposite to the NER results, where the feature-based BERT + BiLSTM + Softmax has the best performance. However, since this thesis focuses on NER and therefore does not extensively experiment with the BERT architectures for SRL, it is not possible to determine whether the fine-tuning, the CRF, or both are responsible for the higher F1-score. Nonetheless, it can be deduced that the results for NER do not follow the same pattern as the SRL results. One of the reasons for this difference might be related to the number of labels. As illustrated in Table 3.5, the number of SRL labels constitutes of almost twice the number of NER labels. As a consequence, the CRF layer is probably more useful for SRL since patterns between consecutive labels can be learned more easily.

Furthermore, both the SRL models score considerably lower on the arena role class compared to the victim and suspect classes. Especially the precision of the arena class suffers (as shown in Table 4.15 and Table 4.16). One possible reason for this observation is the fact that there are approximately three times fewer arena role mentions than there are victim or suspect role mentions (Table 3.2). Moreover, the role annotations are more varied (i.e. location names, descriptions, references) than the entity annotations (i.e. city names). Hence, the limited number of labels probably has a stronger effect on SRL than NER.

To summarize, the fine-tuned BERT with CRF shows that semantic roles involving victims and suspects can be extracted in the homicide corpus relatively successfully. However, the arena class is more difficult to extract. Additionally, the general performance of the SRL task falls short compared to NER.

5.4 Interpretation of case-level results

The previous sections discussed the NER and SRL results. However, the extraction of homicide-related information involves the extraction of case-level named entities and case-level roles as well. Therefore, based on these case-level results, as presented in section 4.3 and section 4.4, the main research question can be answered:

MRQ: *To what extent can BERT be leveraged for information extraction on homicide-related Dutch texts?*

Case-level entity extraction

As discussed in section 5.1, the BERT-based NER models obtain fruitful results. Furthermore, as depicted by Table 4.17, these token-level entities can successfully be translated into case-level entity predictions (mean accuracy of 0.874). Moreover, the exact accuracy, as defined in subsection 3.6.3, is still decently high as well (0.787). Especially the arena class has a high exact accuracy (0.90) compared to the victim class (0.744) and suspect class (0.717). Interestingly, the results show that the victim and suspect classes score considerably lower on the exact accuracy compared to the mean accuracy. This pattern probably follows from the fact that the texts contain far more entity mentions of persons (i.e. victim, suspect, and person classes) than mentions of locations (i.e. arena and location classes), as shown in Table 3.1. Most likely, the models confuse the entity mentions of persons more easily among each other than with the entity mentions of locations. Consequently, the probability to have at least one wrongly classified entity mention is higher for the case-level suspect and victim classes than for the arena class.

Furthermore, when the errors are analyzed in Table 4.18, it needs to be noted that the proposed NER system has considerably more false positives compared to false negatives. The major cause of this behavior is the structure of the homicide corpus. Namely, the corpus has far more positive labels (when a victim, suspect, or arena is present) than negative labels (when there is no victim, suspect, or arena). Consequently, the NER module is more prone to wrongly labeling an entity mention than wrongly dismissing an entity mention.

To summarize, the case-level extraction of the arena class can successfully be extracted based on the NER predictions. However, the suspect and victim classes obtain lower accuracies than the arena class. Still, since the exact accuracies of all components are over 70%, the case-level entity extraction is considered to perform decently.

Case-level role extraction

As presented in Table 4.19, the mean accuracy for the case-roles predictions (0.690) is substantially lower than the case-entities predictions (0.874). The most evident reason for this discrepancy is related to the performance gap between the NER and SRL models.

Furthermore, some patterns of the token-level results of SRL are similar to the case-level results from Algorithm 2. For instance, as with the SRL results, the arena class has the worst performance (mean accuracy of 0.62) while the victim class has the best (mean accuracy of 0.75). Nevertheless, the exact accuracies of the case-level classes do not follow this pattern since the suspect class has the lowest exact accuracy (0.417). Moreover, the exact accuracy for all three classes is quite low, with a maximum of only 0.511 for the victim class. Probably, this difference between the mean and the exact accuracy is related to the large number of case-roles per class. In fact, as shown in Table 3.5, there are more than twice as many case-roles (2,951) than case-entities (1,346). As a consequence, it is more probable to mislabel at least one of the case roles than one of the case-entities. Similarly, as depicted in Table 4.20, the number of positive case-level suspect class roles is considerably larger compared to the victim or arena counterparts. Probably, this explains why the exact accuracy is the worst for the suspect class.

Based on these interpretations, one can argue that the information extraction of case-level roles is insufficient at the moment. Especially regarding the exact accuracy, the SRL system is not good enough. Nevertheless, the mean accuracies for the victim and suspect classes are 70% and higher. Therefore, when the exact accuracy is irrelevant for the task at hand, the SRL system can still provide decent insights for the victim and suspect classes.

5.5 Limitations and future research

In the previous section, I elucidate the results. However, there are some important limitations to consider when interpreting the results. Moreover, the limitations of the current thesis shape the foundations for future research. Therefore, this section elaborates on the limitations and relates them to suggestions for future research. To examine the limitations and ideas for future research in a structured manner, I review this thesis according to the four (construct, internal, external, conclusion) validity threats as proposed by Wohlin et al. (2000). First, the construct validity threats reflect on the decisions that I based on theory. Subsequently, the internal validity threats relate to the factors that could have affected the final evaluation scores. Then, the external validity threats discuss the generalizability of the proposed systems. Finally, the conclusion validity threats depict the extent to which the implications of this thesis are valid.

5.5.1 Construct validity threats

First of all, the annotations from the homicide corpus capture only a limited amount of information. For the NER annotations, only names are considered. Consequently, the NER algorithms fail to detect alternative relevant phrases which the SRL algorithms can detect (e.g. a generic description of the arena). Still, the SRL annotations fail to include all the relevant phrases as well. Namely, in line with the original PropBank guidelines (Palmer et al., 2005), only verb predicates are considered. Therefore, as pointed out by Bonial et al. (2014), some relevant information can get lost. For example, the sentence "The death of Jack is remarkable" has no verb predicate but an adjective predicate ("remarkable"). Nevertheless, the sentence conveys relevant information concerning the victim that should be taken into account. Therefore, future research can experiment with the use of non-verb predicates for homicide-related texts. Possibly, by including several predicate types the SRL system can be more effective at homicide-related information extraction.

Additionally, the NER and SRL models identify the victim, suspect, and arena for each sentence separately. Nevertheless, these classes are defined based on the whole text instead of sentences. After all, a victim remains the victim throughout the whole text. Most likely, the lack of information extraction across multiple sentences limits the performance of the NER and SRL models. Consider the sentence: "Jack got shot". In one case "Jack" might be the victim since he died, while in another case he might only be wounded and thus does not constitute a victim (according to this thesis' definition of the victim class). Only models that find inter-sentence dependencies are able to find such a pattern. Hence, future studies should examine techniques that enable the discovery of inter-sentence dependencies. For instance, coreference resolution has shown to be successful for this purpose (Quirk and Poon, 2017).

Another limitation concerns the SRL annotations. In this thesis, I manually annotate the predicates in the sentence to prepare the SRL task. This implies that for every new incoming text the predicates need to be manually annotated before the SRL system can be utilized. Eventually, however, the SRL system should be able to work autonomously. Existing research already explores methods to automatically identify predicates (Björkelund et al., 2010; He et al., 2018). Therefore, future research can examine such predicate detection techniques for crime texts as well.

Furthermore, the current thesis only applies multilingual BERT. However, other promising BERT-inspired models are not experimented with. For instance, RoBERT (Delobelle et al., 2020) might be an interesting alternative. RoBERT uses the RoBERTa architecture, which has recently surpassed the NER performance of regular BERT models (Wang et al., 2020). Similarly, De Vries et al. (2019) show that BERTje outperforms multilingual BERT for typical SRL and NER tasks. Therefore, future research can investigate how BERT-inspired models improve information extraction tasks for homicide-related texts. Possibly, some BERT-inspired models outperform the baseline model to a larger extent than the currently applied multilingual BERT.

Finally, Algorithm 1 and Algorithm 2 are rule-based algorithms that infer the case-level labels from the NER and SRL token-level predictions. However, in this thesis, I do not consider any alternative approaches. For instance, one can use a method that resembles the semantic merging method as proposed by Srinivasa and Thilagam (2019). Hence, future studies should investigate the approach on how to successfully infer case-level predictions.

5.5.2 Internal validity threats

Due to time constraints, the texts are manually annotated once. Accordingly, there are no subsequent annotation assessments. Hence, there can be accidentally wrongly annotated samples in the ground truth annotations. This could tamper with the training of the models as well as evaluation calculations and thus negatively affect the internal validity of the experiments.

Another limitation concerns the metrics of the case-level evaluations. In contrast to the token-level predictions, the case-level predictions consist of a list of strings rather labels. In this thesis, a prediction is considered to be correct only when the entire predicted string is equal to the string from the ground truth. For example, "John K." is not equal to "John K" (no dot at the end) and thus wrong according to the current evaluation metric. Consequently, the current evaluations for the case-level predictions are conservative. In future studies, other evaluations can be applied that provide a more accurate representation of the actual performance.

5.5.3 External validity threats

In this thesis, I use the self-constructed homicide corpus for information extraction. However, there are a few limitations concerning this corpus. First of all, the corpus is small compared to other commonly used corpora such as CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003), CoNLL-2012 (Pradhan et al., 2012) or SoNaR (Vossen et al., 2016). Consequently, the homicide corpus contains relatively little data to learn patterns from. Second, the corpus originates from one source (moordzaken.com). The texts from this source are relatively short with a high proportion of relevant information. Therefore, it remains unclear to what extent the models' performance is generalizable to texts from other sources that do not have these characteristics. For instance, as explained in subsection 2.4.3, Vossen et al. (2016) show a severe drop in NER performance when their models are applied to corpora that were not used for training. Therefore, future research should evaluate the models from this thesis on homicide texts from alternative sources. Additionally, future research can enrich the homicide corpus by supplementing the corpus with homicide texts from other sources with more complex or ambiguous texts. Hypothetically, this would make the models more effective and make them generalize better.

Furthermore, the annotations for NER and SRL of this thesis are confined to a relatively small set of crime classes (i.e. victim, suspect, arena). Based on the ESC12 (De Kock, 2014), there is more relevant information that can be extracted. Especially the timeframe and means are suitable for the NER and SRL tasks since clauses that contain dates or weapons can easily be annotated with the BIO or BILOU schemes. Moreover, according to De Kock (2014), the timeframe and means of a homicide case are objective components (i.e. they do not depend on interpretations, feelings, or imaginations). Therefore, future research can apply information extraction to a wider range of crime classes to enhance the applicability of the systems.

5.5.4 Conclusion validity threats

This thesis mainly focuses on the NER models. Therefore, I only implement two BERT model configurations and no baseline. As a consequence, this limits the implications that can be inferred from the SRL results. Therefore, future studies can compare a wider range of BERT architectures for SRL (e.g. the NER architectures from this thesis). Such research will help to discover and understand more effective SRL architectures. Furthermore, the SRL models in this thesis are not subjected to any hyperparameter tuning. Therefore, subsequent research should explore the impact of different parameters for SRL to obtain more robust conclusions.

In addition, the NER models are restricted to a few hyperparameter configurations. Hence, future research can explore the effect of other parameter settings on the homicide corpus as well. As a start, in line with the suggestions from Peters et al. (2019), the weight decay or certain learning rate schedules can be experimented with. Again, such experimentation can make the corresponding inferences more robust than the current thesis shows.

Finally, another potential limitation concerns the number of runs that are performed for each model. Namely, due to time constraints, I run each model for only four times. However, since the results for the NER architectures are close, running the models for more iterations could lead to different results. Nevertheless, since the patterns of the development and test set are similar (see Table B.2 and Table B.1 in Appendix B), it is expected that the four runs are sufficiently representative of the actual model performances. Still, subsequent studies can test this claim by running for more iterations.

Chapter 6

Conclusion

This thesis aimed to extract relevant information from publicly available Dutch homicide-related texts. Therefore, I constructed a homicide corpus comprising textual descriptions of 510 homicide cases. For the text of each case, the crime classes (i.e. victim, suspect, arena) are annotated to prepare NER and SRL. With the creation of this homicide-specific ontology on a newly constructed homicide corpus, I contribute to the literature on crime-related information extraction.

Subsequently, the three crime classes are extracted through the use of a NER system and a SRL system. Both systems comprise a module to identify the three classes at a token level (e.g. word, punctuation mark), and a rule-based algorithm to infer the case-level classes.

For the NER system, the NER module predicts the crime entity class for each token. In this NER module, I experimented with several BERT-based architectures. The architectures differ in two main aspects. Firstly, BERT is implemented using a feature-based or fine-tuned approach. Secondly, the models use either a Softmax or CRF classification layer. Based on the results of the experiments four out of the five BERT-based architectures outperform the baseline model. Furthermore, the feature-based models and Softmax models achieve higher results than the fine-tuned and CRF models. Consequently, the use of a feature-based BERT with two BiLSTM layers and a Softmax classification layer (dropout rate of 0.5) achieves the best performance for NER on the homicide corpus (F1-score of 0.847). The observations concerning the BERT configurations contribute to the NER literature since prior works (Dasgupta et al., 2017; Souza et al., 2019; Straková et al., 2019) do not compare the BERT configurations to the extent that this research does. Furthermore, the best model of this thesis has better performance than prior crime-related research on NER (Dasgupta et al., 2017), but worse compared to regular NER research (Straková et al., 2019). Interestingly, the arena class is extracted most successfully for all the applied models. Subsequently, Algorithm 1 utilizes the token-level predictions from the best NER model to obtain predictions for entire homicide cases. These case-level entity predictions are retrieved with an accuracy of 87%. Similar to the NER results, the arena is extracted most accurately at a case level. Therefore, it can be concluded that the NER system is able to extract crime-related information quite accurately. Still, some advancements need to be made before the NER system can be used by law enforcement agencies. Especially the extraction of victim names and suspect names should be improved since these classes are perfectly extracted (i.e. exact accuracy) in 74% and 72% of the cases respectively.

Concerning the SRL system, the crime roles are predicted at a token level through the SRL module. The crime roles are defined to be any name, description, or reference of a crime class. In contrast to the NER module, only two BERT-based models are applied for SRL. Interestingly, the results of the SRL models differ from the NER models. Accordingly, the best performing SRL model uses fine-tuned BERT with a CRF layer (F1-score of 0.714). Notably, the SRL module scores poorer compared to the NER module, but it presents more useful predictions as well. With Algorithm 2 these predictions are used to infer the role predictions at a case level. Again, the case-level results are worse compared to NER (accuracy of 69%). Therefore, the performance of the SRL system is insufficient to be deployed by the police for now. Still, the proposed SRL system contributes to the SRL literature since it constitutes the initial step towards a practical application of SRL (i.e. crime knowledge base creation).

Even though the proposed systems should not be used in practice yet, this thesis presents the first step towards real-world applicable information extraction systems. Hence, future research should improve the current systems. For instance, implementing inter-sentence coreference resolution, using more training data, or applying alternative BERT-inspired models, probably increases NER and SRL performances. Additionally, the performance of the SRL module can probably be enhanced by utilizing non-verb predicates or by experimenting with more model configurations.

In conclusion, this thesis shows that there is potential for BERT-based models that conduct crime-specific NER and SRL tasks. Especially the NER system achieves promising results, outperforming alternative models that apply crime-related information extraction (Dasgupta et al., 2017). Still, insights from future research are necessary to improve the performances of the NER and SRL systems. Only after these advancements, the systems can get sufficiently accurate to be deployed by the police.

References

Agerri, R., Aldabe, I., Laparra, E., Rigau, G., Fokkens, A., Huijgen, P., Izquierdo, R., Erp, M. V., Vossen, P., Minard, A.-L., and Magnini, B. (2016). Multilingual Event Detection using the NewsReader Pipelines. In *Proceedings of the Cross-Platform Text Mining and Natural Language Processing Interoperability workshop.* 21, 22, 30

Agerri, R., Bermudez, J., and Rigau, G. (2014). IXA pipeline: Efficient and ready to use multilingual NLP tools. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3823–3828. European Language Resources Association (ELRA). 22, 30, 37

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics. 2, 15, 17, 20

Arkhipov, M., Trofimova, M., Kuratov, Y., and Sorokin, A. (2019). Tuning Multilingual Transformers for Named Entity Recognition on Slavic Languages. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93. Association for Computational Linguistics. viii, ix, 41, 42

Arulanandam, R., Savarimuthu, B. T. R., and Purvis, M. A. (2014). Extracting Crime Information from Online Newspaper Articles. In *Proceedings of the Second Australasian Web Conference - Volume 155*, pages 31–38. Australian Computer Society, Inc. 2, 3, 4, 19, 26, 27, 29

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern information retrieval.* Addison-Wesley Longman Publishing Co., Inc. 13

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473.* viii, 7, 10, 11, 12, 21

Bamman, D. and Smith, N. A. (2015). Open Extraction of Fine-Grained Political Statements. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 76–85. Association for Computational Linguistics. 2

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247. Association for Computational Linguistics. 13, 14

Bethard, S. and Yadav, V. (2018). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158. Association for Computational Linguistics. 6

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python.* O'Reilly Media, Inc., 1st edition. 26

Björkelund, A., Bohnet, B., Hafdell, L., and Nugues, P. (2010). A High-Performance Syntactic and Semantic Dependency Parser. In *COLING 2010: The 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics. 71

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146. 21

Bonial, C., Bonn, J., Conger, K., Hwang, J. D., and Palmer, M. (2014). PropBank: Semantics of New Predicate Types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3013–3019. European Language Resources Association (ELRA). 23, 70

Carreras, X. and Márquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164. Association for Computational Linguistics. 23

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. 10

Chen, D. and Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750. Association for Computational Linguistics. 7

Cheng, J., Dong, L., and Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561. Association for Computational Linguistics. viii, 12, 13

Chiticariu, L., Li, Y., and Reiss, F. R. (2013). Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832. Association for Computational Linguistics. 27

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics. 10

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics. viii, 10, 21

Christensen, J., Mausam, Soderland, S., and Etzioni, O. (2010). Semantic Role Labeling for Open Information Extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60. Association for Computational Linguistics. 7

Christensen, J., Mausam, Soderland, S., and Etzioni, O. (2011). An Analysis of Open Information Extraction Based on Semantic Role Labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture (K-CAP 2011)*, pages 113–120. Association for Computing Machinery. 7

Cimiano, P. and Völker, J. (2005). Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 166–172. INCOMA Ltd. 5, 22

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics. 22

Costantino, M., Morgan, R. G., Collingham, R. J., and Garigliano, R. (1997). Natural language processing and information extraction: qualitative analysis of financial news articles. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pages 116–122. 2

Cui, Y., Che, W., Liu, T., Qin, B., Yang, Z., Wang, S., and Hu, G. (2019). Pre-Training with Whole Word Masking for Chinese BERT. *arXiv preprint arXiv:1906.08101*. 18

Daelemans, W., Van den Bosch, A., and Weijters, T. (1997). IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms. *Artificial Intelligence Review*, 11(1-5):407–423. 24, 26

Daelemans, W., Zavrel, J., Van der Sloot, K., and Van den Bosch, A. (2010). Timbl: Tilburg memory based learner, version 6.3, reference guide. *ILK Research Group Technical Report Series 10-01*. 24, 26

Dasgupta, T., Naskar, A., Saha, R., and Dey, L. (2017). CrimeProfiler: Crime information extraction and visualization from news media. In *Proceedings of the International Conference on Web Intelligence (WI 2017)*, pages 541–549. Association for Computing Machinery. 1, 2, 3, 4, 6, 26, 27, 29, 64, 65, 73, 74

De Kock, P. A. M. G. (2014). Anticipating criminal behaviour: Using the narrative in crime-related data. In *Tilburg: Tilburg center for Cognition and Communication (TiCC)*. 27, 29, 33, 72

De Vries, W., Van Cranenburgh, A., Bisazza, A., Caselli, T., Van Noord, G., and Nissim, M. (2019). BERTje: A Dutch BERT Model. *arXiv preprint arXiv:1912.09582*. 18, 71

Delobelle, P., Winters, T., and Berendt, B. (2020). RobBERT: a Dutch RoBERTa-based Language Model. *arXiv preprint arXiv:2001.06286*. 19, 71

Desmet, B. and Hoste, V. (2014). Fine-Grained Dutch Named Entity Recognition. *Language Resources and Evaluation*, 48(2):307–343. 26

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*. viii, viii, 2, 3, 4, 7, 8, 13, 16, 17, 18, 25, 26, 40, 41, 42, 43, 44, 49, 66, 67

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and Weischedel, R. (2004). The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA). 6

Elloumi, S., Jaoua, A., Ferjani, F., Semmar, N., Besançon, R., Al-Jaam, J., and Hammami, H. (2012). General learning approach for event extraction: Case of management change event. *Journal of Information Science*, 39(2):211–224. 6

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211. 9

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370. Association for Computational Linguistics. 19, 26

Fitzgerald, N., Täckström, O., Ganchev, K., and Das, D. (2015). Semantic Role Labeling with Neural Network Factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970. Association for Computational Linguistics. 24

Fleischman, M. and Hovy, E. (2002). Fine Grained Classification of Named Entities. In *COLING 2002: The 19th International Conference on Computational Linguistics*. 5, 22

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741. 26

Gildea, D. and Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288. 24

Goldberg, Y. and Hirst, G. (2017). *Neural Network Methods for Natural Language Processing.* Morgan Claypool Publishers. viii, 7, 8, 14

Golub, G. and Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2(2):205–224. 14

Grishman, R. and Sundheim, B. (1996). Message Understanding Conference - 6 : A Brief History. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, pages 466–471. Association for Computational Linguistics. 6, 31

He, L., Lee, K., Levy, O., and Zettlemoyer, L. (2018). Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369. Association for Computational Linguistics. 71

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Séaghdha, D. O., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2010). SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)*, pages 33–38. 7

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.* 9

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-term Memory. *Neural Computation*, 9(8):1735–1780. 9

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv preprint arXiv:1508.01991.* viii, viii, viii, viii, 9, 10, 20, 66

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691. Association for Computational Linguistics. 7

Johansson, R. and Nugues, P. (2008). Dependency-based Semantic Role Labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics. 23

Joshi, M., Levy, O., Weld, D. S., and Zettlemoyer, L. (2019). BERT for Coreference Resolution: Baselines and Analysis. *arXiv preprint arXiv:1908.09091.* 2

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759.* 15

Kingma, D. P. and Ba, J. L. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.* 8

Kiss, T. and Strunk, J. (2006). Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 32(4):485–525. 26

Ku, C. H., Iriberri, A., and Leroy, G. (2008). Natural language processing and e-government: Crime information extraction from heterogeneous data sources. In *Proceedings of the 2008 International Conference on Digital Government Research*, pages 162–170. 1

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, page 282–289. Morgan Kaufmann Publishers Inc. 19, 20

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics. 20, 42

Lee, D. D. and Seung, H. S. (2001). Algorithms for Non-negative Matrix Factorization. *Advances in Neural Information Processing Systems*, pages 556–562. 14

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225. 13, 15

Li, Z., He, S., Zhao, H., Zhang, Y., Zhang, Z., Zhou, X., and Zhou, X. (2019). Dependency or Span, End-to-End Uniform Semantic Role Labeling. *arXiv preprint arXiv:1901.05280.* viii, 23, 24

Lin, Z., Feng, M., Dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A Structured Self-attentive Sentence Embedding. *arXiv preprint arXiv:1703.03130.* 12

Ling, W., Dyer, C., Black, A., and Trancoso, I. (2015). Two / Too Simple Adaptations of Word2Vec for Syntax Problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304. 7, 15

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692.* 19

Lu, Z. (2020). NER implementation with BERT and CRF model. *Github repository.* Retrieved from `https://github.com/Louis-udm/NER-BERT-CRF`. 40

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics. 26

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge university press. 13

Marksberry, P. and Parsley, D. (2011). Managing the IE (Industrial Engineering) Mindset : A quantitative investigation of Toyota's practical thinking shared among employees. *Journal of Industrial Engineering and Management*, 4(4):771–799. viii, 14

Màrquez, L., Carreras, X., Litkowski, K. C., and Stevenson, S. (2008). Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159. 2, 4, 22, 23, 34

Masters, D. and Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. *arXiv preprint arXiv:1804.07612*. 44

Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*. 15

Minard, A. L., Speranza, M., Urizar, R., Altuna, B., Van Erp, M., Schoen, A., and Van Son, C. (2016). MEANTIME, the NewsReader mMultilingual Event and Time Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4417–4422. European Language Resources Association (ELRA). 22

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, , and Xiang, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics. 10

Nguyen, T. H. and Grishman, R. (2015). Relation Extraction: Perspective from Convolutional Neural Networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48. Association for Computational Linguistics. 15

Oostdijk, N., Reynaert, M., Hoste, V., and Schuurman, I. (2013). *The Construction of a 500-Million-Word Reference Corpus of Contemporary Written Dutch.*, pages 219–247. Springer Berlin Heidelberg. 18

Ordelman, R. J. F., de Jong, F. M. G., van Hessen, A. J., and Hondorp, G. H. W. (2007). TwNC: a Multifaceted Dutch News Corpus. *ELRA Newsleter*, 12(3-4). 18

Ouchi, H., Shindo, H., and Matsumoto, Y. (2018). A Span Selection Model for Semantic Role Labeling. *arXiv preprint arXiv:1810.02245*. 24, 25, 68

Palmer, M., Kingsbury, P., and Gildea, D. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106. 7, 22, 23, 33, 70

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe : Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics. 14

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics. 2, 15, 17

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. *arXiv preprint arXiv:1903.05987*. 3, 25, 42, 43, 66, 67, 72

Pradhan, S., Moschitti, A., Uryupina, O., Xue, N., and Zhang, Y. (2012). CoNLL-2012 Shared Task : Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40. Association for Computational Linguistics. 6, 23, 72

Pradhan, S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow Semantic Parsing using Support Vector Machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 233–240. Association for Computational Linguistics. 24

Pradhan, S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2005). Semantic Role Labeling Using Different Syntactic Views. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 581–588. Association for Computational Linguistics. 24

Quinlin, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc. 22

Quirk, C. and Poon, H. (2017). Distant Supervision for Relation Extraction beyond the Sentence Boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182. Association for Computational Linguistics. 6, 71

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8). 7, 13

Raganato, A. and Tiedemann, J. (2018). An Analysis of Encoder Representations in Transformer-Based Machine Translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297. Association for Computational Linguistics. 13

Ratinov, L. and Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155. Association for Computational Linguistics. 26, 31, 37

Reimers, N. and Gurevych, I. (2017). Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*. viii, 21, 41, 66

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiský, T., and Blunsom, P. (2015). Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*. viii, 11, 15

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386–408. 8

Rosenblatt, F. (1962). *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms.* Spartan Books. 8

Roth, M. and Lapata, M. (2016). Neural Semantic Role Labeling with Dependency Path Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202. Association for Computational Linguistics. 24

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, 3rd edition. 1

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 19

Santos, D., Seco, N., Cardoso, N., and Vilela, R. (2006). HAREM: An advanced NER evaluation contest for Portuguese. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA). 25, 65

Schraagen, M. P., Brinkhuis, M. J. S., and Bex, F. J. (2017). Evaluation of Named Entity Recognition in Dutch online criminal complaints. *Computational Linguistics in the Netherlands Journal*, 7:3–16. 2, 3, 4, 26, 64

Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123. 15

Schuurman, I., Hoste, V., and Monachesi, P. (2010). Interacting Semantic Layers of Annotation in SoNaR, a Reference Corpus of Contemporary Written Dutch. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA). 7, 23, 26, 35, 37

Shi, P. and Lin, J. (2019). Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv preprint arXiv:1904.05255*. 2, 3, 4, 8, 23, 25, 26, 68

Singh, S. (2018). Natural Language Processing for Information Extraction. *arXiv preprint arXiv:1807.02383*. 2, 6, 7

Souza, F., Nogueira, R., and Lotufo, R. (2019). Portuguese Named Entity Recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*. 3, 4, 25, 41, 42, 43, 44, 65, 66, 67, 73

Srinivasa, K. and Thilagam, P. S. (2019). Crime base: Towards building a knowledge base for crime entities and their relationships from online news papers. *Information Processing and Management*, 56(6). 1, 2, 3, 4, 6, 27, 71

Straková, J., Straka, M., and Hajic, J. (2019). Neural Architectures for Nested NER through Linearization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 5326–5331. Association for Computational Linguistics. 2, 3, 4, 10, 21, 25, 26, 42, 43, 44, 49, 65, 66, 67, 73

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2019). How to Fine-Tune BERT for Text Classification? *arXiv preprint arXiv:1905.05583*. 2, 17, 44, 65

Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15. Association for Computational Linguistics. viii, 23, 34

Tieleman, T. and Hinton, G. (2012). Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. *Technical report*. 8

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. Association for Computational Linguistics. 6, 19, 31, 65

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147. Association for Computational Linguistics. 2, 6, 31, 65, 72

Van Den Bosch, A., Busser, B., Canisius, S., and Daelemans, W. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. In *Computational linguistics in the Netherlands: Selected papers from the Seventeenth CLIN Meeting*, pages 99–114. 26

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems*, page 6000–6010. Curran Associates, Inc. viii, 12, 16

Vig, J. and Belinkov, Y. (2019). Analyzing the Structure of Attention in a Transformer Language Model. *arXiv preprint arXiv:1906.04284*. 13

Vossen, P. T. J. M., Agerri, R., Aldabe, I., Cybulska, A. K., van Erp, M. G. J., Fokkens, A. S., Laparra, E., Minar, A., Palmero Aprosio, A., Rigau, G., and Segers, R. H. (2016). NewsReader: Using knowledge resources in a cross-lingual reading machine to generate more knowledge from massive streams of news. *Knowledge-Based Systems*, pages 60–85. 22, 24, 68, 72

Wang, Y., Sun, Y., Ma, Z., Gao, L., Xu, Y., and Sun, T. (2020). Application of Pre-training Models in Named Entity Recognition. *arXiv preprint arXiv:2002.08902*. 71

Weiss, D., Alberti, C., Collins, M., and Petrov, S. (2015). Structured Training for Neural Network Transition-Based Parsing. 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)*, pages 323–333. Association for Computational Linguistics. 7

Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., and Wesslen, A. (2000). *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers. 70

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771.* 40

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, , Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144.* 10, 16

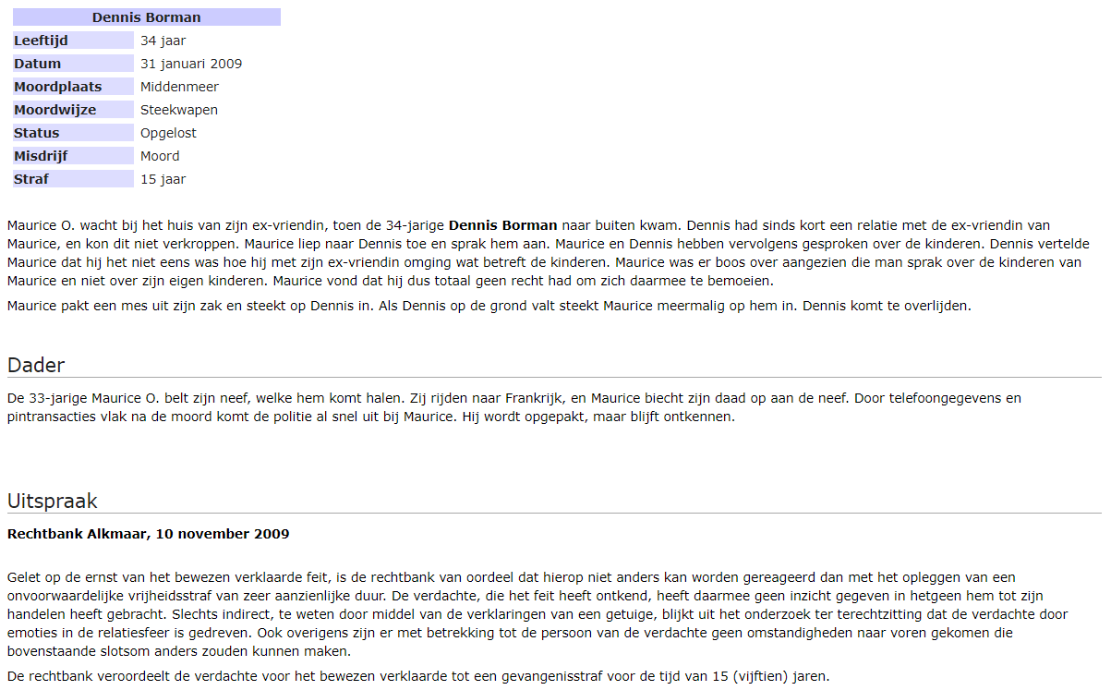
Yadav, V., Sharp, R., and Bethard, S. (2018). Deep Affix Features Improve Neural Named Entity Recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172. Association for Computational Linguistics. 20, 24, 40, 43, 45, 49, 53, 59, 64, 88, 90

Zhangguoxioa (2019). bert-for-srl. *Github repository.* Retrieved from: `https://github.com/zhangguoxiao/bert-for-srl`. 37, 46

Zhou, J. and Xu, W. (2015). End-To-end learning of semantic role labeling using recurrent neural networks. 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)*, pages 1127–1137. Association for Computational Linguistics. 24

Appendix A

Example moordzaken.com



Dennis Borman	
Leeftijd	34 jaar
Datum	31 januari 2009
Moordplaats	Middenmeer
Moordwijze	Steekwapen
Status	Opgelost
Misdrijf	Moord
Straf	15 jaar

Maurice O. wacht bij het huis van zijn ex-vriendin, toen de 34-jarige **Dennis Borman** naar buiten kwam. Dennis had sinds kort een relatie met de ex-vriendin van Maurice, en kon dit niet verkroppen. Maurice liep naar Dennis toe en sprak hem aan. Maurice en Dennis hebben vervolgens gesproken over de kinderen. Dennis vertelde Maurice dat hij het niet eens was hoe hij met zijn ex-vriendin omging wat betreft de kinderen. Maurice was er boos over aangezien die man sprak over de kinderen van Maurice en niet over zijn eigen kinderen. Maurice vond dat hij dus totaal geen recht had om zich daarmee te bemoeien.

Maurice pakt een mes uit zijn zak en steekt op Dennis in. Als Dennis op de grond valt steekt Maurice meermalig op hem in. Dennis komt te overlijden.

Dader

De 33-jarige Maurice O. belt zijn neef, welke hem komt halen. Zij rijden naar Frankrijk, en Maurice biecht zijn daad op aan de neef. Door telefoongegevens en pintransacties vlak na de moord komt de politie al snel uit bij Maurice. Hij wordt opgepakt, maar blijft ontkennen.

Uitspraak

Rechtbank Alkmaar, 10 november 2009

Gelet op de ernst van het bewezen verklaarde feit, is de rechtbank van oordeel dat hierop niet anders kan worden gereageerd dan met het opleggen van een onvoorwaardelijke vrijheidsstraf van zeer aanzienlijke duur. De verdachte, die het feit heeft ontkend, heeft daarmee geen inzicht gegeven in hetgeen hem tot zijn handelen heeft gebracht. Slechts indirect, te weten door middel van de verklaringen van een getuige, blijkt uit het onderzoek ter terechtzitting dat de verdachte door emoties in de relatiesfeer is gedreven. Ook overigens zijn er met betrekking tot de persoon van de verdachte geen omstandigheden naar voren gekomen die bovenstaande slotsom anders zouden kunnen maken.

De rechtbank veroordeelt de verdachte voor het bewezen verklaarde tot een gevangenisstraf voor de tijd van 15 (vijftien) jaren.

Figure A.1: Example case from moordzaken.com

Appendix B

Results on all configurations

Architecture	Parameter configuration			Precision	Recall	F1
	Dropout*	Learning rate**	# BiLSTM-layers			
BERT fine-tuned + Softmax	0.5	5e-5	NA	0.799	0.820	0.809
BERT fine-tuned + Softmax	0.5	2e-5	NA	0.820	0.842	0.831
BERT fine-tuned + Softmax	0.1	5e-5	NA	0.815	0.844	0.829
BERT fine-tuned + Softmax	0.1	2e-5	NA	0.832	0.845	0.838
BERT fine-tuned + CRF	0.5	5e-5	NA	0.822	0.837	0.829
BERT fine-tuned + CRF	0.5	2e-5	NA	0.804	0.824	0.814
BERT fine-tuned + CRF	0.1	5e-5	NA	0.813	0.830	0.821
BERT fine-tuned + CRF	0.1	2e-5	NA	0.795	0.812	0.803

BERT feature-based + BiLSTM + Softmax	0.5	NA	1	0.833	0.839	0.836
BERT feature-based + BiLSTM + Softmax	0.5	NA	2	**0.846**	**0.848**	**0.847**
BERT feature-based + BiLSTM + Softmax	0.1	NA	1	0.832	0.835	0.833
BERT feature-based + BiLSTM + Softmax	0.1	NA	2	0.844	0.842	0.843
BERT feature-based + BiLSTM + CRF	0.5	NA	1	0.788	0.825	0.806
BERT feature-based + BiLSTM + CRF	0.5	NA	2	0.821	0.847	0.834
BERT feature-based + BiLSTM + CRF	0.1	NA	1	0.762	0.803	0.782
BERT feature-based + BiLSTM + CRF	0.1	NA	2	0.803	0.841	0.821
BERT feature-based + CRF	0.5	NA	NA	0.644	0.715	0.677
BERT feature-based + CRF	0.1	NA	NA	0.575	0.706	0.634
FastText + BiLSTM + CRF (Yadav et al., 2018)	0.55	NA	1	0.802	0.791	0.796

bold = best score

* dropout rate between BERT and its successive layer

** learning rate of the BERT encoder

Table B.1: All NER results on the **test** set

Architecture	Parameter configuration			Precision	Recall	F1
	Dropout*	Learning rate**	# BiLSTM-layers			
BERT fine-tuned + Softmax	0.5	5e-5	NA	0.775	0.792	0.783
BERT fine-tuned + Softmax	0.5	2e-5	NA	0.795	0.814	0.804
BERT fine-tuned + Softmax	0.1	5e-5	NA	0.785	0.805	0.795
BERT fine-tuned + Softmax	0.1	2e-5	NA	0.791	0.810	0.800
BERT fine-tuned + CRF	0.5	5e-5	NA	0.754	0.781	0.767
BERT fine-tuned + CRF	0.5	2e-5	NA	0.730	0.759	0.744
BERT fine-tuned + CRF	0.1	5e-5	NA	0.739	0.762	0.750
BERT fine-tuned + CRF	0.1	2e-5	NA	0.727	0.756	0.741
BERT feature-based + BiLSTM + Softmax	0.5	NA	1	0.795	0.811	0.803
BERT feature-based + BiLSTM + Softmax	0.5	NA	2	**0.803**	**0.817**	**0.810**
BERT feature-based + BiLSTM + Softmax	0.1	NA	1	0.788	0.799	0.793
BERT feature-based + BiLSTM + Softmax	0.1	NA	2	0.790	0.806	0.798

BERT feature-based + BiLSTM + CRF	0.5	NA	1	0.761	0.792	0.776
BERT feature-based + BiLSTM + CRF	0.5	NA	2	0.779	0.809	0.794
BERT feature-based + BiLSTM + CRF	0.1	NA	1	0.750	0.785	0.768
BERT feature-based + BiLSTM + CRF	0.1	NA	2	0.777	0.804	0.791
BERT feature-based + CRF	0.5	NA	NA	0.637	0.708	0.671
BERT feature-based + CRF	0.1	NA	NA	0.602	0.703	0.649
FastText + BiLSTM + CRF (Yadav et al., 2018)	0.55	NA	1	0.745	0.725	0.735

bold = best score

* dropout rate between BERT and its successive layer

** learning rate of the BERT encoder

Table B.2: All NER results on the **development** set

Appendix C

Case-level example predictions

C.1 Example 1:

Text:
Op 10 april 2005 vinden voorbijgangers het lichaam van een man aan de Amstelkade te Wilnis .
Onderzoek wijst uit dat het gaat om de 58-jarige Henny Zevenhoven uit Mijdrecht .
Hij is door verstikking om het leven gekomen .
Er worden twee " vrienden " van Henny aangehouden .
Zij bekennen Henny te hebben gewurgd met een schoenveter .
Een uitspraak is niet (meer) online te vinden .
Wij hebben geen verdere informatie kunnen vinden over deze zaak .

VICTIM entity predictions: 'Henny Zevenhoven'
VICTIM role predictions: 'het lichaam van een man', 'de 58-jarige Henny Zevenhoven uit Mijdrecht'

SUSPECT entity predictions: NaN
SUSPECT role predictions: 'twee " vrienden " van Henny'

ARENA entity predictions: 'Wilnis'
ARENA role predictions: 'de Amstelkade te Wilnis'

Note:
To obtain the entity predictions, the text is used as input for the NER system.
To obtain the role predictions, the text and the verb predicates are used as input for the SRL system.

C.2 Example 2:

Text:
Op 25 juni 2006 wordt de 42-jarige Ahmet Naci Havucgil uitgenodigd door twee mannen waarmee hij zaken doet , om een gesprek te voeren .
De mannen denken dat Ahmet hen financieel benadeeld heeft , en willen geld zien van hem .
Ahmet rijdt naar de afgesproken plek in Huissen , alwaar de heren , die ook wat vrienden bij zich hebben , hem vragen even mee te rijden in de auto .
Het stel rijdt naar een afgelegen boerderij in Angerlo .
Daar wordt Ahmet vastgebonden en mishandeld .
Later die avond wordt hij naar de kelder gebracht .
De volgende dag wordt Ahmet door de dan 35-jarige Ertugrul A. en de evenoude Cengiz K. meegenomen naar een bosgebied in Beekbergen .
Daar wurgen ze Ahmet , en verstoppen zijn lichaam onder wat takken en bladeren .
Pas op 10 augustus 2006 wordt het lichaam van Ahmet gevonden .
Enkele van de mannen proberen te pinnen met de bankpas van Ahmet .
Aan hand van deze gegevens worden de eerste verdachten aangehouden .

VICTIM entity predictions: 'Ahmet Naci Havucgil'
VICTIM role predictions: 'zijn lichaam', 'het lichaam van Ahmet', 'de 42-jarige Ahmet Naci Havucgil'

SUSPECT entity predictions: 'Ahmet', 'Cengiz K.', 'Ertugrul A.'
SUSPECT role predictions: 'de heren', 'twee mannen waarmee', 'Enkele van de mannen', 'de eerste verdachten', 'de evenoude Cengiz K.', 'de dan 35-jarige Ertugrul A.'

ARENA entity predictions: 'Huissen', 'Angerlo', 'Beekbergen'
ARENA role predictions: 'de kelder', 'een bosgebied in Beekbergen'

Note:
To obtain the entity predictions, the text is used as input for the NER system.
To obtain the role predictions, the text and the verb predicates are used as input for the SRL system.

C.3 Example 3:

Text:
Op zondag 23 april 2006 scheldt de 49-jarige Gerrit Stijf de voicemail in van zijn ex-vrouw .
Zij laat dit weten aan haar tweelingbroer , Frans van ' t H.
Hij belt twee vrienden .
De maat is vol .
Na de roerige scheiding tussen zijn zus en Gerrit moet het gedonder maar eens afgelopen zijn .
De drie spreken af om Gerrit bang te maken met messen en een stok .
Met een list maakt Frans van ' t H. een afspraak met Gerrit .
Hij haalt Gerrit op bij zijn huis , en rijdt met hem naar een industrieterrein in Kampen , waar hij zijn twee vrienden had afgezet .
Als Gerrit de andere mannen ziet , wil hij via de achterbank van de auto vluchten .
De mannen slaan Gerrit hierop met de stok .
De twee anderen steken op hem in .
Zwaargewond laten ze Gerrit achter op het industrieterrein , waar hij een paar uur later door toevallige passanten wordt gevonden .
Gerrit wordt naar het ziekenhuis gebracht , maar overlijdt een dag later , op 24 april , aan zijn verwondingen .
Naast de 21-jarige broer van Gerrit 's ex , worden ook zijn vrienden Krekar R. (21) en Herman van M. (20) gearresteerd .
Zij leggen allen een uitgebreide verklaring af .
De ex-vrouw van Gerrit wist niets van de plannen van haar broer en diens vrienden .

VICTIM entity predictions: 'Gerrit Stijf'
VICTIM role predictions: 'de 49-jarige Gerrit Stijf'

SUSPECT entity predictions: 'Krekar R.', 'Herman van M.', "Frans van ' t H."
SUSPECT role predictions: 'De mannen', 'De twee anderen', 'Herman van M. (20)', 'De ex-vrouw van Gerrit', 'zijn vrienden Krekar R. (21)', "haar tweelingbroer , Frans van ' t H. drie"

ARENA entity predictions: 'Kampen'
ARENA role predictions: 'zijn huis', 'het industrieterrein', 'een industrieterrein in Kampen'

Note:
To obtain the entity predictions, the text is used as input for the NER system.
To obtain the role predictions, the text and the verb predicates are used as input for the SRL system.

Appendix D

Code of the thesis

The code of this thesis can be found at the following Google Drive folder:
`https://drive.google.com/drive/folders/1ba0L3SXSeYfRZVm2XTjatV4UJBuyui5Z`

The code is divided into three categories: code for data preparation, code for the NER system, and code for the SRL system. The folders in Google Drive are structured according to this division. Each of the folders includes Google Colab files with the code. Importantly, however, some of the folder structure is created after running the code. Moreover, some of the files were copied from a local directory to the Google Drive folder. Consequently, some of the file paths that are stated in the files are no longer accurate.

D.1 Code for data preparation

The folder *prepare-homicide-texts* contains all the code that I use to collect and prepare the data. There are four steps in this process.

First, I scrape the data from the website moordzaken.com. The code where I scrape and filter the majority of the data can be found within the folder "scraping" in the file *scrape_moordzaken.ipynb*.

Second, I use the code in the folder *NER-annotations* to add the NER annotations to the homicide texts. The file *create_initial_dataframe.ipynb* automatically initializes the texts with NER labels. Subsequently, I manually annotate the texts with NER labels. Then, I apply *manual_adjustments_of_initial_dataframe.ipynb* to filter out a few homicide cases and *add_manualEntities_to_dataframe.ipynb* to update the files with the manual labels. Furthermore, the file *NER-traintestsplit* splits the NER data into the training, development, and test set.

Third, in the folder *EntityCase-annotations* the file *add_manualCaseEntities_to_dataframe.ipynb* first initializes the texts with case-level entity labels based on the NER labels. Subsequently, after these labels are manually assessed and corrected, another function in this file adds these manual annotations to the NER data frame.

Fourth, the files in the folder *SRL-annotations* prepare the SRL task for the homicide texts. At first, the file *create-TSVfiles-from-XML.ipynb* transforms the SoNaR XML data into tsv files. Subsequently, *create-initial-SRLlabels.ipynb* creates tsv files with the initial SRL labels for the homicide texts. Then, *predict-labels-with-BERT.ipynb* predicts the PropBank argument spans for these homicide texts based on the SoNaR data. In the next step, the predictions are transformed to allow for manual annotation by running *predictions-to-annotate-format.ipynb.* Finally, after the manual annotations are finished, the homicide texts with the manual annotations are converted to a tsv file by *manualAnnotations-to-finalTSV.ipynb.*

D.2 Code for the NER system

The second main folder named *NER* consists of all the code that is relevant for the NER system. Firstly, the *Data* folder contains all the data used by the BERT models (*bilou-data* folder), the FastText-based model (*bilou-data-fasttext* folder) and Algorithm 1 (*case-data* folder).

Most importantly, the folder called *NER-module* contains the code for all the conducted NER experiments. The folders *Run1*, *Run2*, *Run3* and *Run4* include all the experiments with the BERT configurations. Furthermore, *FastText-model* contains the code for the four runs of the baseline FastText-based model. Additionally, the *NER-module* folder includes a prediction file for each run (*predictions-feature-do50-lr5e5.ipynb*) that uses the most successful model to create predictions for the test and development set. In addition, *readEvaluation.ipynb* collects the evaluation scores of all the BERT configurations to present an overview of the results.

Furthermore, I use the NER predictions to infer the case-level labels by implementing Algorithm 1 in *predictCaseLabels-Algorithm1.ipynb.* The results for each run are manually copied in the Excel file *CaseEntityResults.xlsx* to obtain the averaged results.

D.3 Code for the SRL system

The final main folder is called *SRL* and includes all the code that is related to the SRL experiments. The *Data* folder contains all the annotated tsv files that the BERT models use to train, develop, and test the models on.

Furthermore, the folder *SRL-module* contains the files with code that trains and evaluates the two BERT models for four runs. The evaluation results for each run are manually copied into the Excel file *SRLresults.xlsx*, where the mean scores of the SRL models are calculated.

Finally, the file called *SRLpredictCaseLabels-Algorithm2.ipynb* applies Algorithm 2 to infer the case-level role from the SRL predictions. Subsequently, the corresponding case-level results are evaluated using the Excel file *CaseRoleResults.xlsx.*