**Book Title Extracting from Historical Newspaper Archives: A Named Entity Recognition Approach**

07/07/2024

Niels Bijl

NOTE: With the approval of my thesis supervisor, generative AI was utilized for grammar correction and for assistance in debugging some code issues during the development process.

**Chapter 1: Introduction**

Book reviews in newspapers not only reflect the reading preferences of their time but also offer a window into the shifting intellectual landscapes that characterize different eras. By analysing which books were highlighted by reviewers and journalists, we can gain insights into the evolving cultural and literary tastes of society.

Notably, to date, no published research specifically targets the extraction of book titles from OCR-scanned historical newspapers. Previous studies, such as those by Do et al. (2012) and Sarimehmetoğlu and Erdem (2023), have focused on extracting book titles from visual sources like book cover images and video content. These methods, while innovative, rely on visual attributes and are fundamentally different from our text-based approach.

The method currently employed by the thesis supervisor to identify book titles in historical newspaper texts is predominantly rule-based. This approach involves selecting segments of 600 characters, identified by a set of rules as those most likely to contain a book title, and comparing them against a database of known book titles to ascertain matches. This method suffers from several drawbacks: it yields suboptimal accuracy, it is dependent on the completeness of the database, it requires extensive manual verification, and specific rules are tailored for a particular newspaper, making it potentially hard to generalize to other newspapers. These limitations compromise the efficacy of cultural analysis over time and restrict our understanding of historical intellectual trends.

Despite its drawbacks, the current method has successfully produced a substantial dataset, serving as a valuable foundation for developing an improved model. In response to these challenges and leveraging the existing dataset, this study proposes a novel approach utilizing advancements in natural language processing (NLP). By harnessing NLP models, we aim to develop a more robust and autonomous system capable of accurately extracting book titles from a vast corpus of newspaper text.

Eisenstein (2019) describes natural language processing (NLP) as a set of methods for making human language accessible to computers. Key applications of NLP include language translation, sentiment analysis, speech recognition, text summarization, and named entity recognition. Named Entity Recognition (NER) is the process of identifying named entities in text. Commonly used entities are people, locations, and organizations (Jurafsky & Martin, 2023).

This thesis explores the innovative idea of categorizing book titles as a specialized form of named entity, a concept that, until now, has not been thoroughly investigated in the context of historical newspapers. Which leads us to the research question of this thesis: *To what extent can Named Entity Recognition be utilized to autonomously extract book titles from OCR-scanned historical newspapers, thereby facilitating deeper cultural and literary analyses?*

The remainder of this thesis is organized as follows:

* **Chapter 2 Concepts and Literature**: This chapter introduces all the relevant concepts necessary for understanding the thesis. It also reviews the literature on these concepts and compares various methods based on existing research.
* **Chapter 3 Data**: This chapter details the process of creating the datasets used in this study, including a comprehensive data analysis.
* **Chapter 4 Methodology**: This chapter describes the methodology employed to identify the most effective method for extracting book titles from historical newspapers.
* **Chapter 5 Results**: This chapter presents the results obtained from the applied methodology.
* **Chapter 6 Discussion**: This chapter discusses and interprets the results, providing insights into why the results turned out as they did.
* **Chapter 7 Conclusion**: The final chapter concludes the thesis by addressing the research question and summarizing the key findings.

**Chapter 2: Concepts and Literature**

This chapter outlines the key concepts and literature for understanding this thesis. We start with Named Entity Recognition (NER) in Natural Language Processing (NLP), covering its challenges, tagging schemes, and evaluation metrics. We then discuss various NER approaches: rule-based, unsupervised, and supervised, focusing on supervised learning. The chapter further highlights the evolution from traditional machine learning to transformer-based Large Language Models. Finally, we delve into the methods of text representation in NER models, including word embeddings.

**Named Entity Recognition**

As introduced in the "Introduction" chapter, Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that involves identifying and classifying entities in text into predefined categories, in most cases: names of people, organizations, and locations. According to Jurafsky & Martin (2023), a named entity is defined as any item that can be distinctly identified by a proper name.

Bird et al. (2009) emphasises two major challenges in NER. The first challenge is the ambiguity of many named entity terms. They illustrate this issue with the examples "May" and "North," which could be part of named entities for dates and geographical locations. However, these terms often refer to a person's name. The second challenge concerns multi-word named entities, which necessitates determining the boundaries of an entity, specifically identifying its beginning and end.

Further in this chapter, we discuss a concept that tackles the ambiguity challenge in section "Text Representation." In section "Models for Named Entity Recognition," we explore models that learn to identify the beginning and end of multi-word named entities to address the second challenge."

**Tagging schemes**

To perform NER in a structured way, tagging schemes are used to label and categorize named entities in text. These schemes determine how each token is marked to indicate its role within a named entity. Bird et al. (2009) describe tokens as sequences of characters, which can include words, subwords, or multiple words. NER operates as a token classification task, with each token assigned a specific label.

The BIO tagging scheme, developed by Ramshaw & Marcus in 1995, is the standard method for sequence labeling. It uses 'B' for the beginning of a named entity, 'I' for tokens inside the entity, and 'O' for tokens outside any entity. Other schemes include the simpler IO and the more complex BIOES, which adds tags for entity endings and single-token entities.

**Evaluation Metrics**

To evaluate the performance of a NER system, evaluation primarily relies on classification metrics. These metrics are best understood through the concepts of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). True Positives represent instances where the model accurately identifies a token as belonging to a specific entity class. False Positives are instances where the model erroneously labels a token as belonging to the entity class. Conversely, True Negatives refer to instances where the model correctly identifies a token as not belonging to the entity class. Lastly, False Negatives occur when the model fails to recognize a token as part of the entity class when it actually is.

**Accuracy**

The simplest evaluation metric is the accuracy score (Equation 1). Accuracy measures the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.

Accuracy = TP + TN / TP + TN+ FP + FN (Equation 1)

When dealing with datasets where certain classes are significantly underrepresented compared to others (known as imbalanced datasets), relying on accuracy as an evaluation metric fails to accurately reflect the true performance of models (Grandini, Bagli, & Visani, 2020). For instance, in our dataset, only approximately 1.4% of all tokens represent book titles (Section “Data Analysis”). A model could achieve an accuracy of 98.6% by simply predicting "NO BOOK TITLE" for every token. Despite this seemingly high accuracy score, the model's actual performance in identifying book titles would be inadequate, as it would fail to detect any of the book titles present in the data.

**Precision and Recall**

In contrast, precision (Equation 2) and recall (Equation 3) are more informative metrics for evaluating model performance in this imbalanced context. Precision measures the proportion of correctly identified book titles out of all tokens predicted as book titles (Grus, 2019). This helps us understand the accuracy of the positive predictions made by the model. Recall, on the other hand, measures the proportion of actual book title tokens that were correctly identified by the model, indicating the model's ability to capture true positives (Grus, 2019).

Precision = TP / TP + FP (Equation 2)

Recall = TP / TP + FN (Equation 3)

**F1 score**

A more sophisticated metric that makes use of precision and recall is the F1 score (Equation 4). The F1 score is the harmonic mean of precision and recall, providing a single score that balances both the precision and the recall (Grus, 2019).

F1 Score = 2 \* (Precision \* Recall) / (Precision + Recall) (Equation 4)

**Approaches to Named Entity Recognition**

To determine the most suitable NER approach for this thesis, this section explores the high-level conceptual methodologies of NER. In their NER survey, Jehangir et al. (2023) categorize NER approaches into three primary categories: rule-based, unsupervised learning, and supervised learning.

**Rule-based approach**

Jehangir et al. (2023) described the rule-based approach as one where a predefined set of rules is used by the system to identify and classify entities within a text. An example of this is approach is the method currently used by the thesis supervisor to extract book titles from newspaper archives. Furthermore, Jehangir et al. (2023) identified two major drawbacks of rule-based approaches: their poor generalizability, as the rules are often domain-specific, and the requirement for advanced programming skills and significant human effort to develop these rules. This domain-specific nature means that rules effective for one newspaper may not be applicable to another, necessitating custom sets of rules for different sources.

**Unsupervised approach**

An unsupervised machine learning approach to NER is typically employed when dealing with data that lacks labels (Jehangir et al., 2023). The common methods for unsupervised learning are association and clustering. However, since this thesis has access to a large, labelled dataset, we will not further explore unsupervised methods.

**Supervised approach**

Géron (2017) explains the concept of supervised machine learning, where models are trained using labelled data. This means that each piece of input data is associated with a corresponding expected output, allowing the model to learn how to map inputs to outputs during the training process. Due to the limitations inherent in rule-based methods and the availability of a substantial labelled dataset, this thesis will concentrate exclusively on the supervised approach. A deeper exploration of supervised models for NER is presented in the section ‘Models for Named Entity Recognition'.

**Models for Named Entity Recognition**

To identify suitable NER models for this thesis, this section examines several widely-used supervised machine learning models.

In their survey on Named Entity Recognition in historical documents, Ehrmann et al. (2023) distinguish between two types of supervised machine learning approaches: traditional machine learning and deep learning. They refer to the traditional machine learning approach as "pre-neural machine learning," which primarily utilizes algorithms that do not involve neural network architectures. Meanwhile, Jurafsky & Martin (2023) define deep learning as the application of modern, multilayered neural networks to model complex patterns and relationships in data.

**Traditional machine learning**

Ehrmann et al. (2023) identified the Conditional Random Field (CRF) classifier as the most prevalent traditional machine learning model for NER. Meanwhile, in a separate survey focused on Named Entity Recognition, Jehangir et al. (2023) explored additional traditional machine learning models, including Decision Trees, Naive Bayes, Hidden Markov Models, Maximum Entropy models, and Support Vector Machines. Ehrmann et al. (2023) concluded that the performance of traditional machine learning approaches in NER is generally significantly lower than that of deep learning approaches. Supporting this, Wang et al. (2016) demonstrated superior performance of deep learning in their study, where a Recurrent Neural Network significantly outperformed a traditional CRF model. They also highlighted how traditional models rely heavily on manual feature engineering, which can be very time-consuming when creating a custom NER system.

**Deep learning**

Collobert et al. (2008) was one of the first studies that applied deep learning for Named Entity Recognition. They successfully used a simple Convolutional Neural Network (CNN) for several Neural Language Processing tasks, including Named Entity Recognition. Géron (2017) provides a detailed explanation of CNNs, noting that in addition to their application in NLP, they have been utilized in computer vision since the 1980s. This connection is logical given that CNNs emerged from the study of the brain’s visual cortex. Convolutional Neural Networks are utilized in NLP to process text by applying convolutional layers. These layers employ multiple filters that slide over the input text data sequentially. Each filter is designed to capture specific local patterns or features within the text, such as suffixes, prefixes, and combinations of words that are indicative of named entities (Keraghel et al. 2024). This enables the CNN to extract meaningful features from the text, and by aggregating these local features, CNNs can effectively extract named entities from large and complex texts.

Another category of deep learning models employed for NER includes Recurrent Neural Networks (RNNs). In contrast to CNNs, RNNs are specifically designed to process sequential data (Keraghel et al. 2024). Sequential data refers to data where the order of elements matters, such as a sequence of words from a text. Jurafsky & Martin (2023) provide a comprehensive explanation of RNNs functionality. RNNs process a sequence of words by iterating through the words one at a time while maintaining an 'internal memory' (hidden state) that captures information about the sequence seen so far. They also note a key limitation of RNNs: the vanishing gradient problem. As the length of the data sequence increases in RNN, the issue of vanishing gradients often arises. This occurs when the gradients, which represent how much and in what direction the weights of the neural network should be adjusted during training, become excessively small. This extremely small gradient effectively prevents the weights from changing, thereby stalling the network's learning process. To address this issue, the Long Short-Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997) was developed. The LSTM uses gating mechanisms to tackle the vanishing gradient problem. These gates decide which information is important to keep or discard in its ‘internal memory’, avoiding the exponential shrinking that is characteristic of the vanishing gradient problem.

Yang and Xu (2020) highlight that the BiLSTM-CRF model, an adapted LSTM network, was considered the state-of-the-art for NER as of 2020. The BiLSTM-CRF network modifies the traditional LSTM architecture by processing data both forwards and backwards, a method known as bidirectional (Jurafsky & Martin, 2023). The outputs from the BiLSTM are then fed into a Conditional Random Field (CRF), which serves as the final output layer.

Apart from the vanishing gradient problem, traditional RNNs also suffer from what is known as the bottleneck problem (Jurafsky & Martin, 2023). In an RNN, the sequential input is fed in one by one, with the hidden state being updated at each step. By the time the final input of the sequence is processed, the hidden state is expected to encapsulate the entire context of the sequence, regardless of its length. However, as the RNN processes longer sequences, early inputs can be "forgotten" because of the limited capacity of the hidden state to retain information over time. The attention mechanism was developed as a solution to this bottleneck problem. Instead of relying solely on the final hidden state to encode the entire sequence, the attention mechanism allows the model to access information from all the hidden states throughout the sequence. This enables the RNN to focus on different parts of the input sequence as needed, thereby retaining important details and improving performance on tasks involving long sequences.

Out of the idea of the attention mechanism introduced in RNNs, the mechanism called self-attention was developed. While traditional attention mechanisms in RNNs improved performance, they still relied on sequential processing, limiting parallelization and efficiency. The self-attention mechanism, however, allows each token in the input sequence to directly interact with every other token, enabling parallel processing and better capturing long-range dependencies. This shift led to a new deep learning architecture known as the transformer architecture introduced by google in the paper “Attention is All you Need” (Vaswani et al., 2017) and revolutionized the field of NLP.

While Yang and Xu (2020) highlighted the BiLSTM-CRF as state-of-the-art as of 2020, Labusch et al. (2019) had already demonstrated the superior performance of an early transformer model, BERT (Devlin et al., 2018), over the BiLSTM-CRF in NER tasks involving historical OCR-scanned German text. Furthermore, Ehrmann et al. (2023) noted in their survey on NER in historical documents that transformer-based networks are surpassing BiLSTM models in the deep learning landscape. Similarly, Sun et al. (2021) studied NER in the biomedical domain, noting that although BiLSTM-CRF models were once considered state-of-the-art, transformer-based models have since surpassed them in performance within the biomedical domain. The importance of Sun et al. (2021) lies in the fact that the biomedical domain, like this study, focuses on very specialized custom named entities.

**Large Language Models**

If we analyse the most popular models on Hugging Face, a collaborative platform for machine learning models and datasets, we find that Large Language Models (LLMs) are the preferred method for NER. Chockalingam et al. (n.d.) define LLMs as “deep learning algorithms that can recognize, extract, summarize, predict, and generate text based on knowledge gained during training on very large datasets.” They further explain how LLMs differ from all other language models that can perform NLP tasks. LLMs are considered large because of two reasons: they are trained on large amounts of data, and they comprise a huge number of trainable parameters.

In the past, LLMs were predominantly based on architectures such as RNNs and LSTM networks. However, the recent development of the transformer architecture has made it the preferred choice for LLMs (Chockalingam et al., n.d.).

In the "Deep Learning" section, we discussed the limitations of RNNs when processing longer sequences. RNNs tend to "forget" early inputs due to the limited capacity of the hidden state. Although adding attention mechanisms to RNNs and LSTMs can alleviate this issue, it often introduces significant complexity. In contrast, transformers are inherently better suited for LLMs because their built-in attention mechanisms effectively capture long-range dependencies.

Furthermore, as mentioned in the "Deep Learning" section, transformers excel due to their ability to perform parallel computations. This results in faster training times, the capability to train models with more parameters, and the ability to handle larger datasets. Consequently, making the large language models even larger. These advantages lead to superior performance in various natural language processing tasks (Amaratunga, 2023).

**Transformer-based Large Language Models**

There are numerous transformer-based Large Language Models, each differing from one another primarily in their architecture and training methods (Amaratunga, 2023). To understand these differences, we first need to explore the transformer’s training process, as outlined by Jurafsky and Martin (2023). The training of transformer models occurs in two main phases: pre-training and fine-tuning.

During the pre-training phase, transformer models use self-supervised learning rather than the supervised training methods discussed in the "Models for Named Entity Recognition" section. In self-supervised learning, the model learns to predict parts of its input data from other parts without relying on explicitly human-labelled data. Two common pre-training tasks are Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, a percentage of the input tokens are masked, and the model is trained to predict these masked tokens. In NSP, the model is given pairs of sentences and must determine if the second sentence follows the first in the original document.

After pre-training, the transformer model is fine-tuned on human-labeled data for specific tasks such as Named Entity Recognition. The purpose of pre-training is to teach the model the meanings of words and their relationships within the language, enabling easier learning of the specific task during fine-tuning. This concept is an instance of transfer learning, where knowledge gained from one task improves performance on a different but related task.

One of the most popular transformer models that revolutionized the field of NLP (Ravichandiran, 2021) is BERT, short for Bidirectional Encoder Representations from Transformers. BERTs training process relies on both MLM and NSP during pre-training. Its architecture is based on the original transformer model introduced by Vaswani et al. (2017).

Liu et al. (2019) identified that BERT was significantly undertrained and introduced an improved version called RoBERTa (Robustly optimized BERT approach). Major enhancements in RoBERTa include training on a much larger dataset and eliminating the NSP task from the pre-training process.

**Text representation**

To understand how text is processed in NER models, we will explore how text is transformed into a representation suitable for machine learning algorithms.

The concept of text representation bridges human language and machine learning by converting text into numerical formats that models can interpret and utilize. Patil et al. (2023) discuss how early techniques, such as One Hot Encoding, Bag of Words, Term Frequency, and Inverse Document Frequency, focused on word frequency. These methods struggled with high-dimensional vector representations in large vocabularies. This limitation was addressed by advanced representations, called word embeddings or word vectors.

Word embeddings are derived using neural networks trained in a self-supervised manner, such as predicting the next word. This method uses a continuous vector space to represent words as low-dimensional arrays of real numbers. These word embeddings capture both the semantic and syntactic aspects of words by considering their context. This allows them to identify relationships such as synonyms, antonyms, and analogies (Patil et al., 2023). Notable examples of word embeddings include Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017).

The previously mentioned word embeddings are categorized as static word embeddings. Static word embeddings map each word to a single fixed vector representation, meaning they cannot account for the different meanings a word might have in various contexts. This limitation is addressed by dynamic word embeddings, which produce context-dependent representations, allowing the same word to have different vector representations depending on the context.

BERT utilizes WordPiece tokenization (Devlin et al., 2019). WordPiece tokenization breaks down words into subwords, which helps manage out-of-vocabulary words and handles rare or complex words by decomposing them into more frequent subword units. These subwords are then converted into numerical representations through a lookup table that maps each subword to a unique vector. During pre-training, BERT trains its own embedding layer, which, after training, transforms the numerical input into dynamic word embeddings. The exact implementation of WordPiece remains unknown, as Google has never open-sourced its implementation (WordPiece Tokenization - Hugging Face NLP Course, z.d.).

**Chapter 3: Data**

This chapter outlines the data used in this study, detailing its collection, preparation, and transformation. Additionally, we analyse the dataset's composition and evaluate the quality of OCR accuracy.

**Data Collection**

The dataset provided was derived from the Leeuwarder Courant (LC), a Dutch newspaper with a digital archive from 1945 until 1995. Despite some gaps, the LC provides a rich source for studying book reviews due to its national prominence and relatively good machine readability compared to other digitized newspapers. The newspapers were digitized using Optical Character Recognition (OCR). Notably, the accuracy of OCR decreases for older issues in the archive (section “Data Analysis”).

The book title extraction process involved three main steps. First, a classical algorithm identified consistent tokens within each book review, extracting a 600-character segment (title pericope) likely containing the book title. This segment was compared to the 'Nederlandse Bibliografie Totaal' (NBT) database. Matches were identified based on the intersection of words between the title pericope and the NBT titles.

Second, to address false negatives from faulty OCR, the title pericopes were processed using a Large Language Model, specifically ChatGPT 4. The results were again matched with the NBT titles, improving the identification rate.

Finally, manual correction ensured the accuracy of the dataset. This involved verifying the identified titles and checking for false positives and negatives. This thorough and time-consuming process resulted in a precise dataset of book titles extracted from the LC.

**Locating Book Titles in Newspaper Text**

A significant challenge in this study was transforming the book titles to their locations in the newspaper text. The given book titles were the official titles from the NBT database, but often these titles did not always match exactly within the text. These titles frequently included the main title, subtitle, and sometimes the author and genre. To address this, an algorithm was developed to locate and match the book titles within the newspaper content.

The first step involved preprocessing the text. Both the newspaper content and the book titles were cleaned to remove extra spaces and were converted to lowercase for consistency. Following this, the algorithm attempted direct matching. Initially, it looked for the exact given title within the newspaper text. If the exact title was not found, the algorithm checked for partial matches. It split the title at various delimiters such as colons, semicolons, equals signs, and commas to extract the main title, in the hope that this main title was present in the text.

For titles that did not match through direct methods, a fuzzy matching technique was employed. The algorithm split the title and the newspaper text into tokens and then formed segments from the text with the same number of tokens as the title. Each segment was compared to the title using a fuzzy matching score, which measured the similarity between the two strings. The segment with the highest similarity score was selected as the best match.

To ensure the accuracy of this fuzzy matching approach, these matched titles were manually verified. The matched text sometimes missed one or two words from the newspaper text, but these instances were retained to maintain data quantity. However, matches that missed too many words of the title or were almost unrecognizable due to faulty OCR were marked and later excluded from the dataset. Some representative examples obtained from the fuzzy matching approach are shown in Table X.

Consequently, this process enabled the determination of the location of the book title within the given newspaper text, which was a necessary step to transform the dataset into a Named Entity Recognition format.

|  |  |
| --- | --- |
| Main book title | Fuzzy matched text in newspaper content |
| het weerlicht op de kimmen | het weerhcht op de kimmen; |
| kosmos vogelveldgids van europa | ders: „vogelveldgids van europa". |
| beter blote jan dan dode jan, en andere uitspraken van louis paul boon | beter blote jan dode jan en andere "ltsp,eng\_ran louis paul boon. |
| knotsgekke uitvindingen van de 19e eeuw | knotsgekke uitvindingen van de 19de eeuw. |

Table x: Examples how the main book title was fuzzy matched in the newspaper content.

**Formatting Data for Named Entity Recognition**

After obtaining the locations of book titles in the newspaper text, the data can be formatted for Named Entity Recognition. Various formatting schemes are discussed in the "Tagging Schemes" section. Archana et al. (2023) examined methods for handling imbalanced data in NER and demonstrated the effectiveness of the IO format in addressing this issue. Given that this study also faces substantial data imbalance (as discussed in the "Data Analysis" section), the IO tagging format will be utilized.

To implement this, the subsequent step involves tokenizing the newspaper text by splitting it into individual words, resulting in an array of separate words. Additionally, an array of labels is generated based on the location of book titles within the text. Each word in the array is assigned a label: "I" (Inside) if the word is part of a book reference, or "O" (Outside) if it is not. For example, in the sentence "Recent heb ik het boek De paarden van oranje gelezen." the words would be tokenized and labeled as follows: ["Recent", "heb", "ik", "het", "boek", "De", "paarden", "van", "oranje", “gelezen”] would have the corresponding labels ["O", "O", "O", "O", "O", "I", "I", "I", "I", “O”].

**Labels from another newspaper**

In addition to the book titles provided from the newspaper "De Leeuwarder Courant", I manually labeled book titles from other newspapers to evaluate the developed NER system's generalizability to other newspapers. This involved detecting articles containing book reviews from the newspapers "Het Parool" and "De Trouw". Articles from both newspapers were merged into a new test dataset. Due to time constraints, only the articles that were most clearly book reviews were selected, resulting in 115 book reviews from "De Trouw" and 193 from "Het Parool." An article was clearly a book review if it contained the characters "blz," "ISBN," and "ƒ". The labeling process was streamlined using a Label Studio environment to facilitate NER labeling.

A major difference between this dataset and the one from “De Leeuwarder Courant” is that this dataset is specifically labeled in a NER format. Instead of matching the given title to the most likely position in the book review, this approach labels each occurrence of the title directly in the text. Consequently, in this dataset, the title always includes the full book title along with the subtitle. In contrast, the “De Leeuwarder Courant” data sometimes required discarding the subtitle to accurately identify the title's position. Additionally, if a title appears multiple times in “Het Parool” or “De Trouw”, it is logically labeled each time it is mentioned. However, this was not the case for the “De Leeuwarder Courant” dataset due to difficulties determining the book title positions.

**Data Analysis**

**Leeuwarden Courant**

As explained in section "Locating Book Titles in Newspaper Text," we manually selected 729 fuzzy matches of book titles as insufficient. Consequently, 594 book reviews were removed from the initial dataset. If a book title within a review was marked as insufficient, the entire review was excluded. This step is crucial to avoid a dataset where some tokens that represent book titles in the review text were incorrectly annotated as non-book titles, which would have misled the model and potentially decreased its performance.

From the Leeuwarder Courant dataset, we compiled a total of 12,535 book reviews, encompassing 23,529 book titles. This dataset contains a total of 7,643,958 tokens, with 110,018 of these tokens being book titles. As a result, book titles represent only 1.4% of the total tokens, indicating a significant class imbalance.

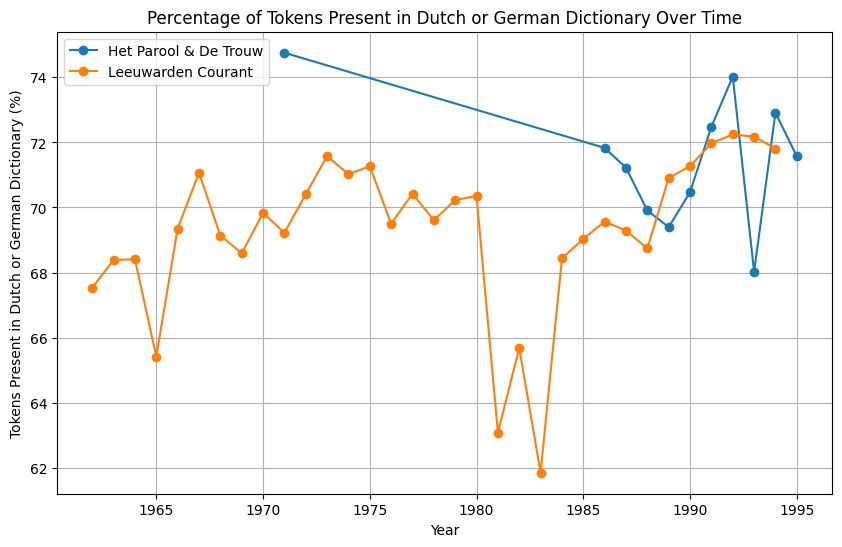
**The Trouw & Parool**

In the section “Labels from Another Newspaper”, we discussed the creation of a second dataset, which includes 115 book reviews from "De Trouw" and 193 from "Het Parool". This dataset contains 221,237 tokens in total, with 4,264 of these tokens representing book titles. Here, too, we observe a substantial class imbalance, as book titles constitute only 1.9% of the total tokens.

**Optical Character Recognition Quality**

To further assess the quality of our dataset, we attempted to quantify the OCR accuracy. This was done by splitting each review into individual words and determining the percentage of words that are present in dictionaries. For Dutch words, we used the dictionary from opentaal.org, which contains a total of 164,105 words. Additionally, since some German books are reviewed, we included a German dictionary from openthesaurus.de, containing 164,913 words. Figure X shows the results of the percentage of words found in the dictionaries over time. This analysis provides an indication of the OCR accuracy, but it is important to note that the dictionaries do not include every Dutch and German word or their derivatives. These were the most complete open-source dictionaries available for download.

Figure X indicates that the OCR performance is generally better for Parool & Trouw compared to Leeuwarden Courant. This may be attributed to the method used distinguish book review articles from other articles in Parool & Trouw, which relies on the presence of the key terms 'blz', 'ISBN', and 'ƒ' in the text. Articles with OCR errors in these key terms are ignored, resulting in the exclusion of many poorly OCR'd articles.



**Chapter 4: Methodology**

This chapter outlines the experiments conducted to identify the most effective NER model for extracting book titles from book review articles and the methods used to evaluate model performance. Initially, the models employed in these experiments are described in detail.

**Models**

In this section, we introduce the Named Entity Recognition models used in this study, starting with the simplest model, and progressing to the most complex. Additionally, we discuss the loss function used in the training process.

**Baseline and SpaCy**

To develop a NER system for extracting book titles from historical newspapers, we initially established a baseline model. This simple model serves as a performance benchmark, ensuring that improvements from more complex models are meaningful.

For our baseline, we utilized spaCy, an open-source software library specifically designed for NLP. SpaCy offers pre-trained models capable of performing various NLP tasks, including NER. In this study, we utilized spaCy's “nl\_core\_news\_lg” model, which is trained on Dutch text, including news articles. This model is capable at identifying a diverse range of entities within Dutch texts. For our purposes, we focused exclusively on the "WORK\_OF\_ART" entity, which includes titles of books, songs, and other artistic works.

While this baseline model may not achieve the highest performance compared to more specialized or custom-trained models, its ease of use and accessibility make it a valuable initial tool for our research.

In addition to using the pre-trained SpaCy NER model, we took advantage of spaCy's capability to fine-tune their pre-trained models on our own custom training data. This adaptability could potentially increase performance in recognizing our custom entity. However, while spaCy offers a streamlined and user-friendly interface, it also imposes certain limitations. The simplified training process restricts the ability to customize various aspects of the training, potentially limiting the performance improvements that could be achieved through more detailed and specific adjustments. The underlying architecture of spaCy's NER model is based on a Convolutional Neural Network (CNN), as detailed in spaCy's official documentation.

**BiLSTM-CRF**

Although the literature (section "Deep Learning") identifies transformer-based models as the state-of-the-art in NER, there is a notable gap regarding their performance on this specific context with a substantial class imbalance. Consequently, this study also incorporates the predecessor state-of-the-art in NER, the BiLSTM-CRF model, to address this specific context and provide a comparative analysis.

Training the BiLSTM-CRF model necessitates converting the text into a numeric representation, as detailed in the "Text Representation" section. This study focused exclusively on pre-trained Dutch word embeddings for this transformation. Various word embeddings were evaluated based on the proportion of unknown tokens in the training data. The largest GloVe embeddings (Pennington et al., 2014) classified 24% of the tokens as unknown. In contrast, FastText embeddings (Bojanowski et al., 2017) marked only 2% of tokens as unknown. Consequently, FastText was chosen for the text-to-numeric transformation in this study.

**Transformer-based Large Language Models**

In addition to utilizing the predecessor state-of-the-art in NER, this research also employs the current state-of-the-art: transformer-based models. This study specifically focuses on models that are not only pre-trained but also already fine-tuned for NER tasks. Liu et al. (2021) demonstrated that fine-tuning transformer models on a specific domain's NER task yields better performance if the model has already been fine-tuned on a NER task from another domain. Hugging Face, a collaborative platform, hosts numerous fine-tuned NER transformer-based models. For this study, several popular transformer models from Hugging Face, which had already been fine-tuned on Dutch NER tasks, were further fine-tuned to enhance their performance on our specific dataset.

The first model employed is the "WikiNEuRal" from Tedeschi et al. (2021). This is a multilingual BERT model that has been fine-tuned on a NER task across nine languages, including Dutch and German.

Another model used is the “xlm-roberta-large-finetuned-conll03-english”, an XLM-RoBERTa model (Conneau et al., 2019). This model is a multilingual extended version of RoBERTa and has been fine-tuned on the CoNLL-2003 dataset (Sang et al., 2003).

The study also incorporates the “robbert-v2-dutch-ner”, developed by Delobelle et al. (2020). This model is based on RoBERTa and fine-tuned specifically for Dutch NER tasks.

Lastly, the “BERTje”, developed by De Vries et al. (2019) at the University of Groningen, is included. This model is BERT-based and pre-trained on Dutch data, including contemporary and historical fiction novels, the Multifaceted Dutch News Corpus (Ordelman et al., 2007), and a collection of Dutch news articles. After pre-training, it was then fine-tuned on the CoNLL-2002 dataset (Sang, 2002).

**Loss Function**

The loss function is a component necessary for training neural networks, as it measures how well the model’s predictions align with the actual target values. During training, the neural network adjusts its weights with the goal to minimize the loss function. Initially, when testing the transformer-based models and the BiLSTM-CRF model, they failed to learn effectively. The default loss function, Cross-Entropy, did not address the significant class imbalance present in the training data (section "Data Analysis" in chapter "Data"). Consequently, the models always predicted that all tokens were "no book" and nothing else.

Nemoto et al. (2024) proposed a solution to this issue by developing a loss function capable of handling class imbalance. They compared several loss functions frequently used for NER tasks with class imbalance and introduced their own loss function, called "Majority or Minority (MoM)". Their comparative analysis on several NER datasets demonstrated that the MoM loss function outperformed the other loss functions commonly used for NER with class imbalance.

Consequently, the MoM loss function was used during training for all transformer-based models and the BiLSTM-CRF model. However, the “user-friendly” SpaCy model did not easily allow for a custom loss function.

**Research Experiments**

This section outlines the experiments conducted to identify the top-performing model in each category of NER models. Subsequently, these top-performing models were compared to determine the best overall NER model. Detailed descriptions of the models used in each experiment can be found in the previous section, "Models."

The datasets used for these experiments are further detailed in the "Data" chapter. The "Leeuwarden Courant" dataset was divided into training (70%), validation (15%), and test (15%) sets. This division allows for a realistic evaluation of the model's performance. The training set is used to train the NER models, the validation set is utilized to determine the hyperparameters and for model selection to prevent overfitting, and the test set provides an unbiased evaluation of the final model's performance. The "Parool & Trouw" dataset was retained as a single unit to serve as a secondary test set, aimed at evaluating the generalizability of the NER models to different newspapers.

For each experiment involving a training procedure, the maximum batch size was used to optimize computational time. The computational resources utilized for these experiments are detailed in Appendix A.

**Tuning the BiLSTM-CRF**

The first experiment in this study involved a hyperparameter search for the number of memory units in the BiLSTM-CRF model. Three variations of the model were trained for 20 epochs with a batch size of 64, each with a different number of memory units: 50, 100, and 200. For each model, the F1 score on the validation set was recorded at each epoch. This allowed us to determine the optimal number of training epochs through a callback mechanism. The final BiLSTM-CRF model selected was the one that demonstrated the overall best performance on the validation set.

**Comparing the Transformer-based Large Language Models**

The second experiment evaluated multiple transformer-based large language models (LLMs) to identify the best performer. Each transformer-based LLM was trained for 20 epochs with batch sizes as shown in Table x. During training, the F1 score on the validation set was recorded at each epoch to determine the optimal number of epochs through a callback mechanism. Among these multiple LLMs, the one with the highest overall F1 score on the validation set was chosen as the final transformer-based LLM for this study.

|  |  |
| --- | --- |
| Model | Batch size |
| WikiNEuRal | 16 |
| xlm-roberta-large-finetuned-conll03-english | 2 |
| robbert-v2-dutch-ner | 16 |
| BERTje | 16 |

Table x: Batch sizes for each transformer-based large language model

**SpaCy**

No experimentation was needed to optimize the SpaCy approaches. The pre-trained SpaCy model (baseline) required no hyperparameter tuning. For the further fine-tuned SpaCy model on our training data, the default settings were used.

**Final Comparison**

The final experiment was conducted after selecting the best transformer-based large language model and the best BiLSTM-CRF model, both determined using the validation set from the "Leeuwarden Courant" data. In this experiment, all final NER models were compared: the SpaCy model (baseline), the fine-tuned SpaCy model, the best transformer-based LLM, and the optimized BiLSTM-CRF model. These comparisons were made using both the test set from the "Leeuwarden Courant" and the "Parool & Trouw" dataset. The evaluation methods for model performance are detailed in the next section, "Performance Evaluation."

**Performance Evaluation**

This section outlines the methods used to evaluate the model's performance, assessed through two primary approaches: token classification performance and the accuracy of book title extraction. For the experiments aimed at identifying the best performing NER model, only token classification performance was considered. After selecting the best NER model, the performance of the actual extracted book titles was estimated.

**Token Classification Performance**

The token classification evaluation measures how accurately tokens are identified as either "I" (indicating a book title) or "O" (indicating no book title). This assessment utilizes the metrics of Recall, Precision, and F1 score, which are detailed in Chapter "Concepts and Literature", under the section "Evaluation Metrics".

**Extracted Book Title Performance**

Beyond token-level performance, this study primarily focuses on the accurate identification of book titles within book review articles. This evaluation involves matching the extracted tokens to the most similar book title in the Nederlandse Bibliografie Totaal (NBT) database. Using the "Leeuwarden Courant" dataset, which contains the exact titles from the NBT for each book review, Recall, Precision, and F1 scores were calculated by comparing the set of predicted book titles with the set of actual book titles.

**Chapter 5: Results**

In this chapter, the results of the experiments are presented.

**BiLSTM-CRF**

The results from the hyperparameter tuning to determine the number of memory units for the BiLSTM-CRF are shown in Table x. The table highlights three different configurations of memory units: 50, 100, and 200. As the number of memory units increases, the total number of parameters (model size) also increases significantly from 141,010 to 804,010.

The F1 score on the Leeuwarden Courant Validation data shows a consistent improvement with the increase in memory units, going from 66.9% for 50 units to 68.8% for 200 units. This suggests that increasing the number of memory units contributes positively to the model's performance. However, the training time also varies with the number of memory units, with 100 units yielding the shortest training time of 278 minutes, whereas 50 and 200 units require 310 and 318 minutes, respectively. This varying training time is influenced by other computations running simultaneously during the training of the models, which could explain why the smallest model is not the fastest.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Memory Units | Total Number of Parameters (model size) | Leeuwarden Courant Validation F1 score (%) | Training Time (minutes) |
| 50 | 141,010 | 66.9 | 310 |
| 100 | 322,010 | 68.0 | 278 |
| 200 | 804,010 | 68.8 | 318 |

Table x: Results from hyperparameter tuning to determine the number of memory units for the BiLSTM-CRF model, including total number of parameters, F1 score on Leeuwarden Courant validation dataset, and training time.

**Transformer-based Large Language Models**

The results from training several transformer-based large language models are presented in Table X. Among the models tested, the "xlm-roberta-large-finetuned-conll03-english" model achieved the highest F1 score on the Leeuwarden Courant validation dataset, with a score of 83.9%. However, this model also required the longest training time at 624 minutes, which is substantially longer compared to the other models. This extended training duration is attributable to the model's larger number of parameters that need to be adjusted during training. Additionally, due to hardware limitations (Appendix A), the maximum batch size for training this bigger model was restricted to 2 (Table X), whereas the other models were trained with a batch size of 16. This smaller batch size further contributed to the increased training time.

|  |  |  |  |
| --- | --- | --- | --- |
| Hugging Face Transformer model | Total Number of Parameters (model size) | Leeuwarden Courant Validation F1 score (%) | Training Time (minutes) |
| WikiNEuRal | 177,264,386 | 80.3 | 108 |
| xlm-roberta-large-finetuned-conll03-english | 558,842,882 | 83.9 | 624 |
| robbert-v2-dutch-ner | 116,173,058 | 78.7 | 110 |
| BERTje | 108,548,354 | 78.5 | 111 |

Table x: Results from training several transformer-based large language models, including total number of parameters, F1 score on Leeuwarden Courant validation dataset, and training time.

**Final Comparison**

In this section, the comparison of all the final models that resulted from the previous experiments was conducted based on their token classification performance. Additionally, for the best model, an analysis was done to see how many books the model correctly identified by matching the predictions to the NBT.

**Token classification**

Table X presents the final token classification results from four final models: the baseline model, the trained SpaCy model, the best BiLSTM-CRF model, and the best transformer-based large language model.

In the Leeuwarden Courant Validation Set, the baseline model performs poorly, whereas the other models demonstrate much better performance, indicating that training on our own dataset enhances results. The BiLSTM-CRF model achieves an F1 score of 68.8%, precision of 73.8%, and recall of 64.4%, outperforming the trained SpaCy model. The transformer-based model, xlm-roberta, exhibits the highest performance with an F1 score of 83.9%, precision of 82.9%, and recall of 85.0%, demonstrating its superior effectiveness on this dataset.

In the Leeuwarden Courant Test Set, performance trends are similar to the validation set, with all models outperforming the baseline. The xlm-roberta model again shows the highest performance.

In the Trouw & Parool Test Set, the baseline model performs better compared to the other datasets but still lags behind the more complex models. The transformer-based model continues to lead, achieving an F1 score of 56.0%, precision of 78.7%, and recall of 43.3%. All models show notably high precision relative to recall. This suggests that when a model identifies a token as a book title, it is usually correct (high precision), but many book title tokens are missed (lower recall). This pattern indicates that the models are conservative in their predictions, opting to classify a token as a book title only when they are confident. This conservative approach reduces false positives but results in several true positives being missed.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Named Entity Recognition model | Leeuwarden Courant Validation | | | Leeuwarden Courant Test | | | Trouw & Parool Test | | |
| F1 (%) | Precision (%) | Recall (%) | F1 (%) | Precision (%) | Recall (%) | F1  (%) | Precision (%) | Recall (%) |
| Baseline (SpaCy) | 9.6 | 12.8 | 7.7 | 9.1 | 12.5 | 7.2 | 21.1 | 31.6 | 15.9 |
| Trained Spacy | 64.9 | 74.7 | 57.3 | 63.9 | 74.3 | 56.0 | 32.0 | 71.8 | 20.6 |
| BiLSTM-CRF (200 memory units) | 68.8 | 73.8 | 64.4 | 69.0 | 74.8 | 64.0 | 34.6 | 77.2 | 22.3 |
| xlm-roberta-large-finetuned-conll03-english | 83.9 | 82.9 | 85.0 | 84.3 | 83.4 | 85.2 | 56.0 | 78.7 | 43.3 |

Table x: Final token classification results from the best model in each category, including F1 score, precision, and recall on the Leeuwarden Courant validation set, Leeuwarden Courant test set, and Trouw & Parool test set.

**Extracted Book Titles**

The transformer-based large language model, xlm-roberta-large-finetuned-conll03-english, demonstrated superior performance on the Leeuwarden Courant validation, Leeuwarden Courant test, and Trouw & Parool test datasets. Consequently, this model was selected for further assessment. Beyond its token classification performance, the model's ability to accurately identify book titles discussed in book reviews was evaluated in Table x.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | True Positives | False Positives | False Negatives |
| Leeuwarden Courant Test |  |  |  |
| Trouw & Parool Test |  |  |  |

Table x: Performance of the xlm-roberta-large-finetuned-conll03-english model in identifying book titles in book reviews. The model's true positives, false positives, and false negatives on the Leeuwarden Courant Test and Trouw & Parool Test datasets.

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | F1 (%) | Recall (%) | Precision (%) |
| Leeuwarden Courant Test |  |  |  |
| Trouw & Parool Test |  |  |  |

Table x: Performance of the xlm-roberta-large-finetuned-conll03-english model in identifying book titles in book reviews. The model's F1 score, recall, and precision on the Leeuwarden Courant Test and Trouw & Parool Test datasets.

**Chapter 6: Discussion**

**Impact of Faulty Optical Character Recognition**

A qualitative analysis of the provided book reviews reveals the inherent imperfections of Optical Character Recognition (OCR). Although a quantitative method to assess OCR accuracy is outlined in Chapter Data, Section “Data Analysis,” pinpointing an exact accuracy rate remains challenging. The quantitative analysis offers only an approximation, indicating that while OCR accuracy is improving over time, it is still not flawless.

A relevant study conducted by Hamdi et al. (2019) investigated the impact of faulty OCR on NER tasks using an LSTM-CRF model. They introduced OCR noise to determine its effect on performance. Their findings demonstrated that increasing the Word Error Rate (WER) from 1% to 7% and the Character Error Rate (CER) from 8% to 20% in OCR outputs resulted in a substantial drop in the F1 score from 90% to 60%. WER quantifies the percentage of incorrectly recognized words, while CER measures the percentage of incorrectly recognized characters. Their study conclusively showed how faulty OCR can drastically affect NER performance.

Unfortunately, since we cannot determine the exact OCR accuracy, we cannot precisely measure the impact of faulty OCR on extracting book titles from historical newspapers. However, it is very likely that our results are impacted by the OCR.

**Manual Analysis of Predictions**

This section analyses the best performing transformer-based large language model, emphasizing its strengths and weaknesses based on a manual review of its predictions on the test datasets to explain the observed F1, recall, and precision metrics.

Firstly, it stands out how the model accurately classifies tokens as book titles, resulting in very few false positives, which aligns with its high precision score.

Secondly, the model consistently identifies book titles without duplicating them when they appear multiple times in the text. However, it marks only one instance of each title, reflecting the structure of the training data, which annotates each book title just once (Chapter Data, section Locating Book Titles in Newspaper Text). Consequently, the annotated positions do not always match where the model marks the title in the text. This observation is contributing to the model’s lower recall score.

Thirdly, the tokens predicted as book titles often exclude the full title, frequently omitting subtitles. This is again logical, as subtitles were frequently omitted to accurately locate the book title in the text. Again, this observation contributes to the lower recall score.

Lastly, the model's performance differs substantially between the Leeuwarden Courant test dataset and the Parool & Trouw dataset. This discrepancy arises because the Leeuwarden Courant data is not annotated directly in a NER format, whereas the Parool & Trouw dataset is. In the Parool & Trouw dataset, each occurrence of a book title is annotated, leading to multiple annotations for the same title, unlike the Leeuwarden Courant data. Consequently, the model is trained to predict only a single instance of each title, causing a much lower recall score for the Parool & Trouw dataset compared to the Leeuwarden Courant dataset. Despite this, the precision score remains high for both datasets, likely because the single instance predicted by the model tends to be accurate.

To address these shortcomings, a potential solution is to annotate the training data directly in a NER format, rather than inferring the most likely position of the book titles based on the given titles from the NBT.

Matching NER output to NBT database

When using title4 (the actual labels used to train the NER), thus if the NER model had optimum performance. Then we found F1 score 65.41%, precision 65.73%, and recall 65.10%.

**Chapter 7: Conclusion & Future Work**

Summary of Findings: Recap the key findings and their significance.

Contributions: State the contributions of your research to the field of applied data science.

Limitations: Acknowledge any limitations encountered during the study.

Future Research: Suggest areas for future investigation.

Future work:

* Clean the OCR

**References**

Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

Eisenstein, J. (2019). Introduction to Natural Language Processing. MIT Press.

Do, Y., Kim, S. H., & Na, I. S. (2012). Title Extraction from Book Cover Images Using Histogram of Oriented Gradients and Color Information. International Journal Of Contents/Journal Of Contents, 8(4), 95–102. <https://doi.org/10.5392/ijoc.2012.8.4.095>

Sarimehmetoğlu, B., & Erdem, H. (2023). Extracting Book Titles From Book Recommendation Videos Using a Deep Learning Approach. MANAS Journal Of Engineering, 11(2), 229–234. <https://doi.org/10.51354/mjen.1369636>

Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O’Reilly Media.

Ramshaw, L. A., & Marcus, M. P. (1995). Text Chunking using Transformation-Based Learning. arXiv.org. <https://arxiv.org/abs/cmp-lg/9505040v1>

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.

Grus, J. (2019). Data Science from Scratch: First Principles with Python. O’Reilly Media.

Jehangir, B., Radhakrishnan, S., & Agarwal, R. (2023). A survey on Named Entity Recognition — datasets, tools, and methodologies. Natural Language Processing Journal, 3, 100017. <https://doi.org/10.1016/j.nlp.2023.100017>

Toral, A., & Munoz, R. (2006). A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia. In Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources.

Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. <http://cds.cern.ch/record/2699693>

Ehrmann, M., Hamdi, A., Pontes, E. L., Romanello, M., & Doucet, A. (2023). Named Entity Recognition and Classification in Historical Documents: A Survey. ACM Computing Surveys, 56(2), 1–47. <https://doi.org/10.1145/3604931>

Wang, W., Bao, F., & Gao, G. (2016, November). Mongolian named entity recognition with bidirectional recurrent neural networks. In 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 495-500). IEEE.

Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning (pp. 160-167).

Keraghel, I., Morbieu, S., & Nadif, M. (2024). A survey on recent advances in named entity recognition. arXiv preprint arXiv:2401.10825.

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica. D, Nonlinear Phenomena, 404, 132306. <https://doi.org/10.1016/j.physd.2019.132306>

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Yang, G., & Xu, H. (2020). A Residual BiLSTM Model for Named Entity Recognition. IEEE Access, 8, 227710–227718. <https://doi.org/10.1109/access.2020.3046253>

Labusch, K., Kulturbesitz, P., Neudecker, C., & Zellhöfer, D. (2019, October). BERT for named entity recognition in contemporary and historical German. In Proceedings of the 15th conference on natural language processing, Erlangen, Germany (pp. 8-11).

Sun, C., Yang, Z., Wang, L., Zhang, Y., Lin, H., & Wang, J. (2021). Biomedical named entity recognition using BERT in the machine reading comprehension framework. Journal Of Biomedical Informatics, 118, 103799. <https://doi.org/10.1016/j.jbi.2021.103799>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. arXiv (Cornell University), 30, 5998–6008. <https://arxiv.org/pdf/1706.03762v5>

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Chockalingam, A., Patel, A., Verma, S., & Yeung, T. (n.d.). A Beginner’s Guide to Large Language Models. NVIDIA Corporation.

Amaratunga, T. (2023). Understanding large language models: Learning Their Underlying Concepts and Technologies. Apress.

Ravichandiran, S. (2021). Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT. Packt Publishing Ltd.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A Survey of Text Representation and Embedding Techniques in NLP. IEEE Access, 11, 36120–36146. <https://doi.org/10.1109/access.2023.3266377>

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, 135-146.

WordPiece tokenization - Hugging Face NLP Course. (z.d.). <https://huggingface.co/learn/nlp-course/chapter6/6>

Archana, S. M., Prakash, J., Singh, P. K., & Ahmed, W. (2023). An Effective Biomedical Named Entity Recognition by Handling Imbalanced Data Sets Using Deep Learning and Rule-Based Methods. SN Computer Science/SN Computer Science, 4(5). <https://doi.org/10.1007/s42979-023-02068-6>

Liu, Z., Jiang, F., Hu, Y., Shi, C., & Fung, P. (2021). NER-BERT: a pre-trained model for low-resource entity tagging. arXiv preprint arXiv:2112.00405.

Tedeschi, S., Maiorca, V., Campolungo, N., Cecconi, F., & Navigli, R. (2021, November). WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER. In Findings of the Association for Computational Linguistics: EMNLP 2021 (pp. 2521-2533).

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116.

Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050.

Delobelle, P., Winters, T., & Berendt, B. (2020). Robbert: a dutch roberta-based language model. arXiv preprint arXiv:2001.06286.

De Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch bert model. arXiv preprint arXiv:1912.09582.

Ordelman, R. J., de Jong, F. M., van Hessen, A. J., & Hondorp, G. H. W. (2007). TwNC: a multifaceted Dutch news corpus. ELRA Newsletter, 12(3-4).

Sang, E. F. T. K. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. arXiv.org. <https://arxiv.org/abs/cs/0209010>

Nemoto, S., Kitada, S., & Iyatomi, H. (2024). Majority or Minority: Data Imbalance Learning Method for Named Entity Recognition. arXiv preprint arXiv:2401.11431.

Hamdi, A., Jean-Caurant, A., Sidere, N., Coustaty, M., & Doucet, A. (2019, June). An analysis of the performance of named entity recognition over OCRed documents. In 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL) (pp. 333-334). IEEE.

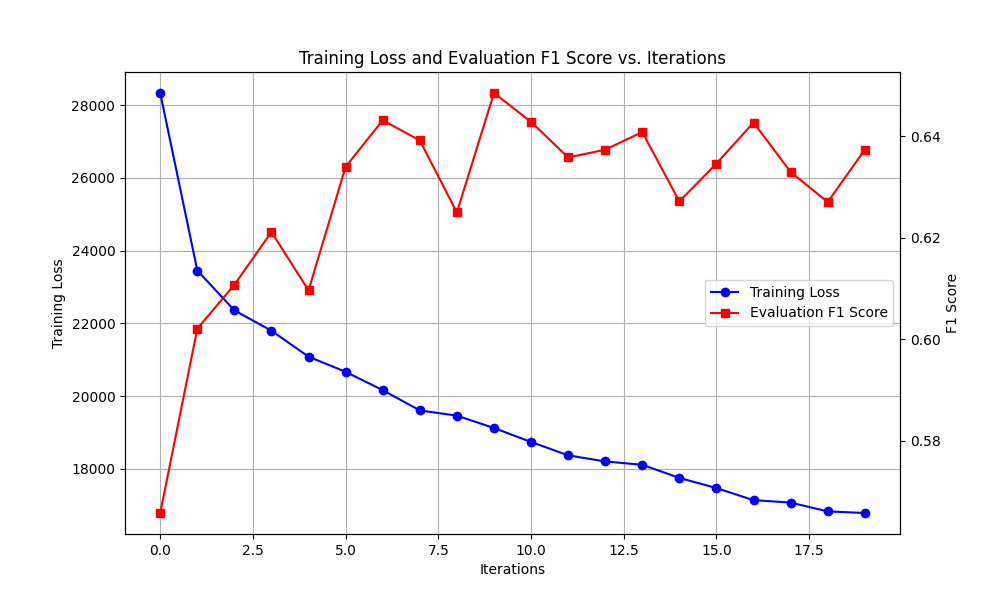
**Appendices**

**A. Hardware specifications**

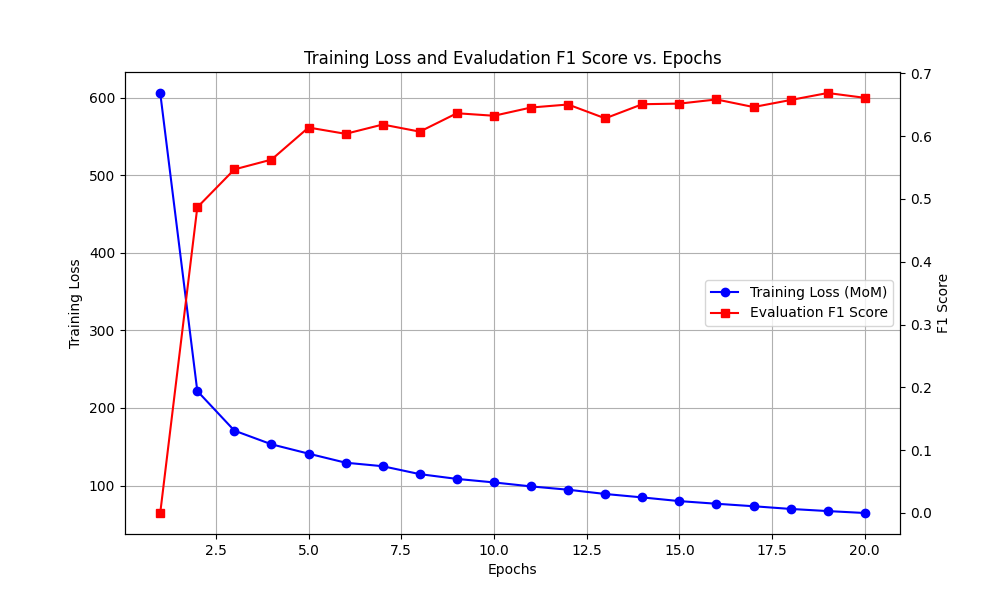
Processor: AMD Ryzen 7 3700X 8-Core   
RAM: 32 GB  
GPU: NVIDIA GeForce RTX 3060 12GB

**B. Training history**

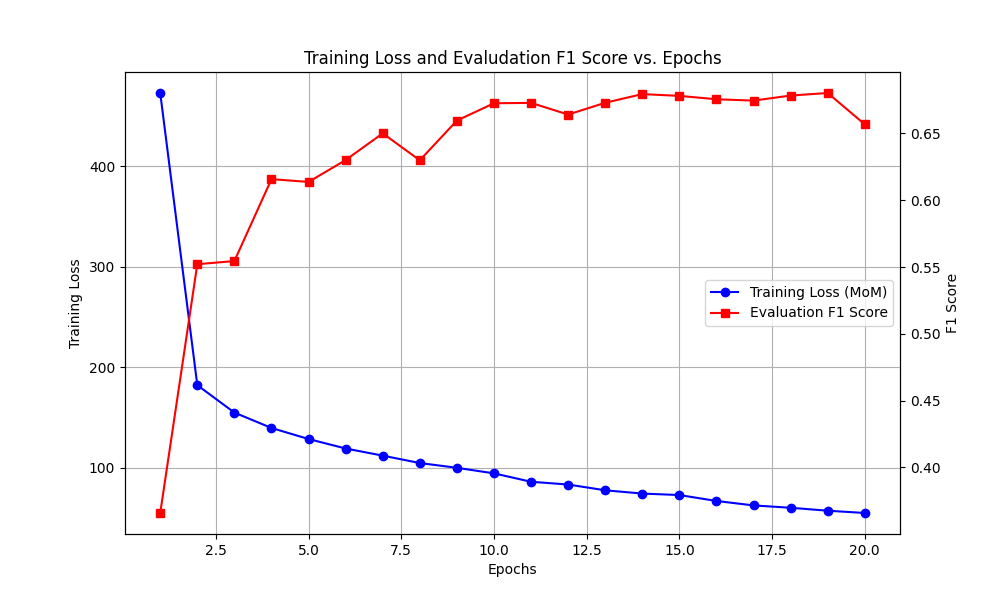
**B.1 SpaCy**



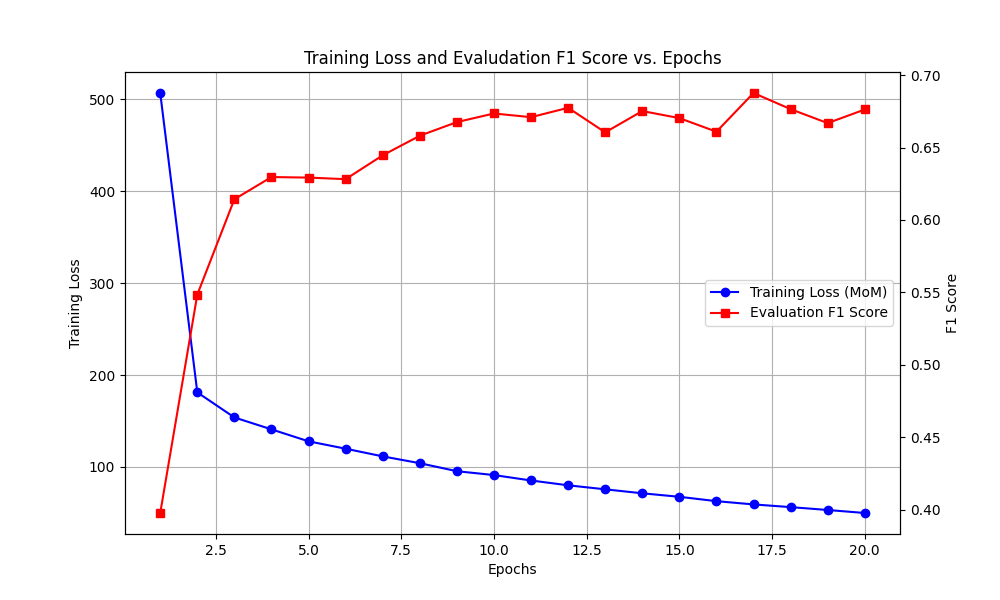
**B.2 BiLSTM-CRF**



B.2.1 50 memory units

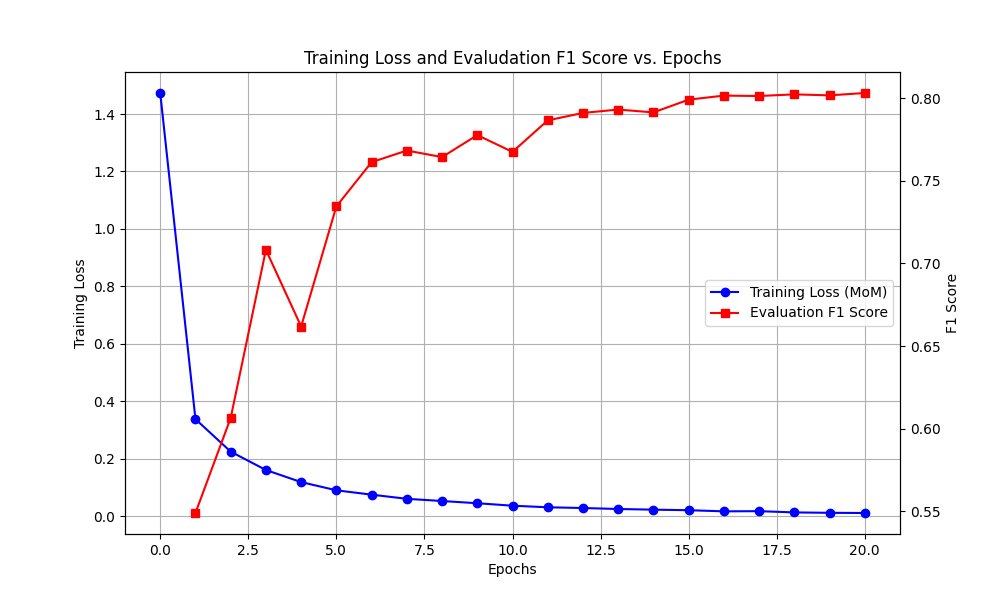


B.2.2 100 memory units



B.2.3 200 memory units

**B.3 Transformers**



B.3.1 Babelscape/wikineural-multilingual-ner



B.3.2 xlm-roberta-large-finetuned-conll03-english

A graph with a red line and blue line

Description automatically generated

B.3.3 pdelobelle/robbert-v2-dutch-ner

**C. Prediction examples on unseen data**

A screenshot of a computer

Description automatically generated

Example on from the Trouw (only the main title is predicted)

A screenshot of a computer

Description automatically generated

Leeuwarden courant

A screenshot of a text box

Description automatically generated

Leeuwarden Courant

A screenshot of a text message

Description automatically generated

Leeuwarden courant

A screenshot of a computer

Description automatically generated

Parool

A screenshot of a book

Description automatically generated

Parool

A screenshot of a computer

Description automatically generated

Parool

A screenshot of a computer

Description automatically generated

Trouw

A screenshot of a computer

Description automatically generated

Trouw

A screenshot of a computer

Description automatically generated

Trouw

A screenshot of a text box

Description automatically generated

Trouw