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

07/07/2024

Niels Bijl

**Chapter 1: Introduction**

Background: Introduce the topic, relevance, and timeliness of your research.

Problem Statement: Clearly state the problem your thesis addresses.

Objectives: Outline the specific goals of your research.

Thesis Structure: Briefly describe the structure of the thesis.

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 analyzing 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 newspaper texts is predominantly rule-based. This approach involves selecting segments of approximately 200 words, 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 highly dependent on the completeness of the database, it requires extensive manual verification, and specific rules are tailored for a particular newspaper, making it hard to generalize to other newspapers. These limitations compromise the efficacy of cultural analysis over time and restrict our understanding of historical intellectual trends.

In response to these challenges, this study proposes a novel approach leveraging advancements in natural language processing (NLP). By utilizing 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 central research question of this thesis: *How can Named Entity Recognition be utilized to autonomously extract book titles from OCR-scanned historical newspaper, thereby facilitating deeper cultural and literary analyses?*

**Chapter 2: Background**

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 highlights the evolution from traditional machine learning to deep learning, particularly the impact of transformer-based models. Lastly, we examine Large Language Models (LLMs), their architecture, training, and text representation.

**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, including 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. They further explain that NER is a subset of a broader category of tasks called sequence labelling, which involves assigning a label to each word in a sequence.

Bird et al. (2009) emphasises two major challenges in Named Entity Recognition (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.

**Tagging schemes**

Tagging schemes in NER provide a structured method 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**

NER is a task focused on token classification, where the 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. 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

When dealing with 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 cases where only 1% of all tokens represent an entity, a model could achieve an accuracy of 99% by merely predicting the majority class. Despite this seemingly high accuracy score, the model's actual performance would be quite inadequate.

**Precision and Recall**

Precision and recall are popular classification metrics, commonly employed as a foundational component for more sophisticated metrics. Precision measures the proportion of correctly predicted entities out of all the entities predicted by the model (Grus, 2019). It addresses the question: "Of all the tokens the model predicted as entities, what proportion was correctly identified as entities?"

Precision = TP / TP + FP

Recall, on the other hand, focuses on the proportion of actual entities that were correctly identified by the model (Grus, 2019). It seeks to answer: "Of all the actual entities, what proportion did the model correctly identify as entities?"

Recall = TP / TP + FN

**F1 score**

A more sophisticated metric that makes use of precision and recall is the F1 score. 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)

When adapting the F1 score for multi-class classification tasks, it can be calculated using two principal methods: micro-averaged and macro-averaged F1 scores (Grandini, Bagli, & Visani, 2020). The micro-averaged F1 score aggregates the counts of true positives, false negatives, and false positives across all classes to compute overall precision and recall. In contrast, the macro-averaged F1 score computes the F1 score for each class separately and then averages these scores. This approach gives equal importance to each class, making it particularly useful for datasets with imbalanced class distributions, as it ensures that each class contributes equally to the overall metric.

**Approaches to Named Entity Recognition**

In their survey on Named Entity Recognition, Jehangir et al. (2023) classified approaches to Named Entity Recognition into three categories: rule-based, unsupervised learning, and supervised learning.

**Rule-based approach**

They 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.

**Unsupervised approach**

Unsupervised learning in Named Entity Recognition (NER) utilizes algorithms that can identify and categorize entities from text without any labelled training data. One such algorithm employs a dictionary of words, often known as a gazetteer-based approach. Bird et al. (2009) describe the gazetteer-based method as involving the examination of each word in the text to determine if it matches an entry in a predefined dictionary of named entities. If a match is found, the word is assigned to the corresponding entity category. This technique relies on direct lookup and comparison, making it straightforward yet dependent on the quality and completeness of the dictionary used. An example of this approach is the study by Toral et al. (2005), which explored the automatic creation and maintenance of gazetteers using Wikipedia. This research demonstrated the method's efficacy in constructing NER gazetteers for location and person categories.

**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 and unsupervised 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 Named Entity Recognition is presented in the section ‘Models for Named Entity Recognition'.

**Models for Named Entity Recognition**

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 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.

**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 understand and interpret large and complex text datasets.

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). Jurafsky & Martin (2023) provide a comprehensive explanation of their 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 counter 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 LSTM (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 model 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) 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**

Chockalingam et al. (n.d.) define Large Language Models (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 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. However, the recent development of the transformer architecture has made it the preferred choice for LLMs (Chockalingam et al., n.d.).

As described in the section “Models for Named Entity Recognition,” RNNs face limitations when processing longer sequences, as early inputs can be "forgotten" due to the limited capacity of the hidden state. While attention mechanisms can be added to RNNs and LSTMs to mitigate this issue, doing so often introduces significant complexity. Transformers, on the other hand, are inherently more suitable for LLMs due to their built-in attention mechanisms, which facilitate the capture of long-range dependencies.

Additionally, the previously described transformers’ ability to parallel compute results in faster training times, the capability to train models with more parameters, and the handling of larger datasets, all leading to better performance on various natural language processing tasks (Amaratunga, 2023).

**Transformer-based Large Language Models**

Transformer-based Large Language Models differ primarily in their architecture and training methods (Amaratunga, 2023). Therefore, we first need to discuss the transformer training process. Jurafsky & Martin (2023) explain that the training of transformer models is divided into two phases: pre-training and fine-tuning.

According to Jurafsky & Martin (2023), pre-training of transformer models differs from most supervised machine learning models discussed earlier in the “Models for Named Entity Recognition” section. Instead of supervised training, pre-training relies on self-supervised learning, where the model learns to predict parts of its input data from other parts without explicit human-labeled data. One pre-training task is Masked Language Modeling (MLM), where a percentage of the input tokens are masked, and the model is trained to predict these masked tokens. Another task is Next Sentence Prediction (NSP), where the model is given pairs of sentences and must determine if the second sentence follows the first in the original document. Lastly, Causal Language Modeling (CLM) is another pre-training task, where the model is trained to predict the next word in a sentence.

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.

BERT (Devlin et al., 2019), short for Bidirectional Encoder Representations from Transformers, is one of the most popular transformer models and revolutionized NLP (Ravichandiran, 2021). 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**

Text representation bridges human language and machine learning algorithms by converting text into a numerical format that models can interpret and use. 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 embeddings or word vectors.

Word vectors 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 vectors 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. Notable examples of word vectors 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 dataset used in our 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 these titles did not always match exactly within the text. Often, the given book title included the main title, subtitle, author, and sometimes the genre. To address this, an algorithm was developed to accurately 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, then searched for these segments within the text. This step addressed cases where the title was only referenced by the main title, without the inclusion of subtitles and authors.

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, the precise text representing the book title in the newspaper was compiled. This process enabled the determination of the exact 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 (NER). 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.

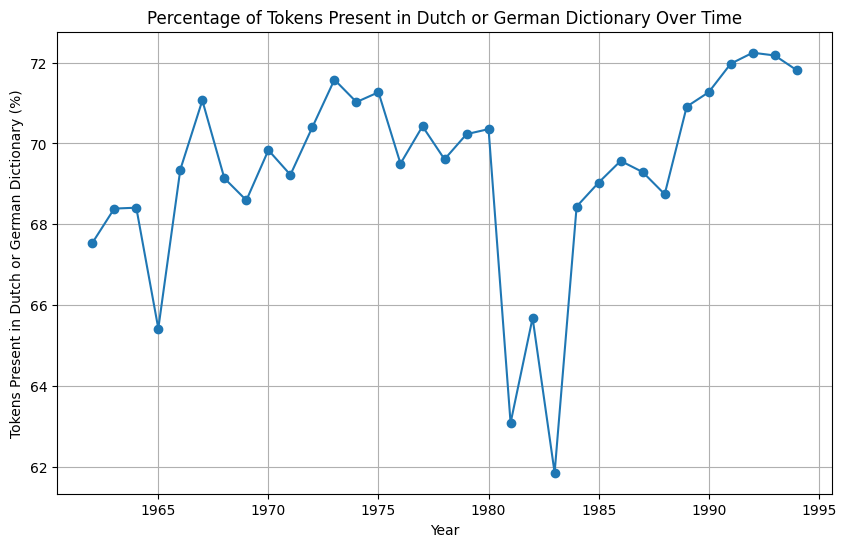
To implement this, the subsequent step involves tokenizing the newspaper text by splitting it into individual words, resulting in an array of separate words. Additionally, an array of labels is generated based on the location of book titles within the text. Each word in the array is assigned a label: "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”].

**Data Analysis**

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, which included 23,529 book titles. Specifically, this dataset contains 7,643,958 tokens in total, with 110,018 of these tokens representing book titles. Consequently, only 1.4% of the total tokens are book titles, highlighting a substantial class imbalance.

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.



**Chapter 3: Methodology**

* Limitations and restrictions
* Research procedure
  + Evaluation
  + Experiments

**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.

**Baseline**

In developing a Named Entity Recognition (NER) system for extracting book titles from historical newspapers, we started with a baseline model. This simple model establishes a performance benchmark, ensuring that any improvements from more complex models are meaningful. Jurafsky & Martin (2023) emphasize the use of a “Most Frequent Class Baseline” in token classification. This baseline model assigns each token to the class it occurred in most often in the training set, in our case the “O” class, meaning no-book-title. By setting this baseline, we can measure the effectiveness of more advanced models.

**Spacy**

In the pursuit of extracting book titles from historical newspapers, we utilized spaCy, an open-source software library designed for NLP. SpaCy provides pre-trained models that perform various NLP tasks, including NER. In this study, we specifically utilized spaCy's Dutch pre-trained NER model, which is proficient in identifying a wide array of entities within Dutch texts. One of the key entity categories it recognizes is "WORK\_OF\_ART," which includes titles of books, songs, and other artistic works. Although this 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 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.

**LSTM-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 significant class imbalance. Consequently, this study also incorporates the predecessor state-of-the-art in NER, the LSTM-CRF model, to address this specific context and provide a comparative analysis.

Training the LSTM-CRF model necessitates converting the text into a numeric representation, as detailed in the "Text Representation" section. This study focused exclusively on specific 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) resulted in 24% of the tokens being marked as unknown. In contrast, FastText embeddings (Bojanowski et al., 2017) had only 2% unknown tokens. Consequently, FastText was chosen for the text-to-numeric transformation in this study.

**Transformer-based**

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, 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).

**Research procedure**

TODO

**Evaluation**

TODO

**Chapter 4: Results & Discussion**

Model Performance: Present the effectiveness of your model in identifying book titles.

Comparisons: Compare your results with existing methods, if applicable.

Discussion: Interpret the results and discuss their implications.

**Chapter 5: 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.

**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, Inc.”

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>

**Appendices**

A. Hardware specifications

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