

Master Thesis

NER Classification for old and modern
Dutch biographies: A comparative study
of finetuned BERT models and
out-of-the-box tools

Felix den Heijer

*a thesis submitted in partial fulfilment of the
requirements for the degree of*

MA Linguistics
(Text Mining)

Vrije Universiteit Amsterdam

Computational Lexicology and Terminology Lab
Department of Language and Communication
Faculty of Humanities



Supervised by: prof. dr. Antske Fokkens, Angel Daza
2nd reader: Pia Sommerauer

Submitted: January 30, 2022

Abstract

In this thesis we investigated the task of Named Entity Recognition (NER) on Dutch biographies by assessing the performance of various models. These models include out of the box tools such as Stanza and Flair, a BERTje model finetuned on CoNLL, along with another BERTje and a GijsBERT model finetuned on historical Dutch NER data.

More specifically, this thesis investigated two things; for one it aimed to discover the better out-of-the-box tool for NER on dutch Biographies, and secondly it aimed to with assess the differences between a more historically pretrained model such as GijsBERT and one with a more general purpose Dutch pretraining when finetuned on the same datasets.

This project contains several test partitions. Two of these were more historical test partitions sourced online. This source was also identical to the training data of the finetuned models. Additionally, an additional high-quality test partition was annotated based on a national archive of Dutch biographical data. Furthermore, extra annotations were made to create two more test partitions to more qualitatively evaluate the BERT based models.

As for the out-of-the-box systems, we found that Flair consistently provided the best results, outperforming both Stanza and the BERTje based model that was finetuned on CoNNL by a large margin. Concerning the finetuned models, we measured similar performance and found that both made similar mistakes on both historical and modern biographical test partitions.


Declaration of Authorship

I, Felix den Heijer, declare that this thesis, titled *NER Classification for old and modern Dutch biographies: A comparative study of finetuned BERT models and out-of-the-box tools* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: **29-06-2023**

Signed:



List of Figures

2.1	Example of how embeddings were pooled [Akbik et al. (2019)]	14
3.1	Bar plot of the label distribution in the train partition	23
3.2	Wordcloud with NER labels for the training set	24
3.3	Stanza generated label distribution for the complete Biography set	25
3.4	Wordcloud for Named entities within the Biography test set	28
4.1	Training losses for each model	33
4.2	Validation losses for each model	34
4.3	Confusion matrix for stanza on NHA partition	35
4.4	Confusion matrix for stanza on SA partition	36
4.5	Confusion matrix for stanza on biography partition	37
4.6	Confusion matrix for Flair on NHA partition	38
4.7	Confusion matrix for Flair on SA partition	39
4.8	Confusion matrix for Flair on biography partition	40
4.9	Confusion matrix for baseline model on NHA test partition	41
4.10	Confusion matrix for baseline model on SA test partition	42
4.11	Confusion matrix for baseline model on biography test partition	43
4.12	Confusion matrix for finetuned BERTje on NHA test partition	44
4.13	Confusion matrix for finetuned BERTje on SA test partition	45
4.14	Confusion matrix for finetuned BERTje on biography test partition	46
4.15	Confusion matrix for finetuned GijsBERT on NHA test partition	47
4.16	Confusion matrix for finetuned GijsBERT on SA test partition	48
4.17	Confusion matrix for finetuned GijsBERT on biography test partition	49
7.1	Word cloud showing the most popular entities in the biography test partition	64
7.2	Word cloud showing the most popular entities in the NHA partition	65
7.3	Word cloud showing the most popular entities in the discarded RHC partition	66
7.4	Word cloud showing the most popular entities in the SA partition	67
7.5	Donut plot showing the percentage of biographies per source in the full biography net	67

List of Tables

2.1	An example of BIO scheme annotation for Named Entity Recognition	8
2.2	Middle Dutch case system from [Hüning and Vogl]	17
3.1	Overview of token counts for each partition in the tag de tekst dataset	22
3.2	Overview of label counts for each partition in the tag de tekst dataset	22
3.3	Most popular PERSON entities found in biography	26
3.4	Entities in the biography test set	27
3.5	Agreement between the three annotators for the biography test partition	28
3.6	All sources with known dates and the dates of their biographies	32
4.1	Classification report for Stanza on NHA partition.	35
4.2	Classification report for Stanza on SA partition.	36
4.3	Classification report for Stanza on Biography test partition	37
4.4	Classification report for Flair on the NHA partition	38
4.5	Classification report for Flair on the SA partition	39
4.6	Classification report for Flair on Biography test partition	40
4.7	Classification report for baseline model on the NHA test partition	41
4.8	Classification report for baseline model on the SA test partition	42
4.9	Classification report for baseline model on biography partition	43
4.10	BERTje on Tag de Tekst model performance for Partition NHA	44
4.11	BERTje on Tag de Tekst model performance for Partition SA	45
4.12	BERTje on Tag de Tekst model performance for Partition Biography	45
4.13	Performance for GijsBERT on NHA test partition	46
4.14	Performance for GijsBERT on SA test partition	47
4.15	Performance for GijsBERT on Biography test partition	48
4.16	Label distribution overview for qualitative partitions	50

Contents

Abstract	i
Declaration of Authorship	iii
List of Figures	v
List of Tables	v
1 Introduction	1
2 Theoretical Framework	7
2.1 The task of Named Entity Recognition	7
2.2 NER before Transformers	8
2.3 Neural Networks	9
2.4 A transformer based Neural Language Model; BERT	10
2.4.1 BERTje; a Dutch BERT Model	11
2.4.2 GijsBERT, a BERT model for Historical Dutch	12
2.4.3 Flair	13
2.4.4 Stanza	14
2.5 Domain Adaptation	14
2.6 Domain Adaptation for BERT	15
2.7 The Historical Dutch Domain	16
2.8 NLP In Historical Texts	18
2.9 The Dutch Biography Portal Domain	18
3 Methodology	21
3.1 Data Overview	21
3.1.1 “Tag de Tekst” Data	22
3.1.2 “Biography” data	24
3.2 Domain Adaptation	28
3.3 Pre- and post- processing	29
3.3.1 Models	30
3.3.2 Experiments	31
4 Results	33
4.0.1 Model Selection	33
4.0.2 Stanza Model	34
4.0.3 Flair Model	38
4.0.4 Baseline Model	40

4.0.5	Qualitative Evaluation	49
5	Discussion	53
5.0.1	Interpretation of findings	53
5.0.2	Limitations and Future Research	54
6	Conclusion	57
7	Appendix	63
7.1	Word Clouds	63
7.2	BiographyNet plots	63

Chapter 1

Introduction

Every person lives a life riddled with experiences. Experiences such as having parents, going to school, or starting a career are shared amongst many, but other experiences are only held by a few. It can be argued that the documentation of human history describes the interplay of these experiences; arranging them in a network of interactions to tell stories.

The world wide web (WWW) has emerged as a medium for storing information, and is thus also purposeful for storing the experiences of many. Social media for example, makes it easy for everyone's experiences to be collected, recorded and shared. Moreover, advanced search methods have enabled the intelligent cataloging of this information, making it easy for people to find and access experiences. These methods however, are limited to digital data and cannot be applied to information that has been recorded in writing. This whilst the majority of human experience stems from the non-digital era.

The Dutch biography portal project (Renders et al., n.d.) was an initiative from the Institute of Dutch History (ING) for producing and retrieving digital biographical information. Their collection now contains 200,000 Dutch biographies that discuss approximately 80,000 individuals. Some of these stem from existing biographical dictionaries that were digitized by the ING, others were produced by the ING in digital form directly. In these biographical dictionaries names can be looked up, and all the found biographies in which they are discussed are listed below.

The massive array of digitized biographical information is more useful when it can be found. Intelligent searching within documents can help bring a reader to the information that they are looking for. Because of this it is valuable to deploy Natural Language Processing (NLP) techniques. Notably, NLP techniques aim to enable computers to interpret human language, and NLP can be applied to a wide range of tasks. The current project aims to explore a NLP task on biographical data called Named Entity Recognition.

Named Entity Recognition (NER) is a specific task within NLP that involves identifying named entities, such as people, organizations, locations, and dates, in text data. In modern day implementation, this is performed through machine learning techniques. Machine learning can be performed for NLP tasks by 'training' a system with an algorithm. Training works by feeding the system organized data, and by having it make

predictions. In supervised machine learning, this data consists of both the 'query' and the 'label'. The query is what the system sees as input, and the label is what the system is trying to predict. In the case of named entity recognition for example, this query will consist of a series of words, called tokens, and the label the system is trying to predict would be the entity, which for example could manifest in the form of "PER" or "LOC"; marking whether a sequence of tokens is a person or location. It is important to note that the system will 'guess' in the first stages of training, whereas later the true labels are revealed to the system, and the system will be punished for its mistakes by the algorithm, allowing it to learn and improve as it witnesses more examples. This is typically how machine learning works when performing classification, but there are a wide range of algorithms that can be applied. Once the algorithm has been trained, we can save it for later use.

Traditional machine learning systems attempted to solve NLP problems by preparing their models with additional information which is informative for the task at hand (Collobert et al., 2011). If a model knows that a word is a proper noun for example, it might also be able to understand that it is more likely for a given token to be an entity. Such information can be added to the representations of text in the form of 'features', and are carefully crafted and selected by humans for each task.

Many of the more recent machine learning models implement 'deep learning'. Deep learning means that deep neural networks are used. In deep neural networks there is a hidden layer that contains nodes that grade importance of the input to determine the output. This means that these systems are capable of understanding the underlying syntactic structure of a sentence for example, so that it can use this knowledge to make its predictions. The field of NLP has widely experimented with various Deep Learning models, one of which will be investigated in this project.

In the context of the biography portal, language models can use NER in order to extract named entities from the digitized biographies and categorize them in a way that makes it easier for users to find specific information. If it is known that a user is looking for a person for example, then NER will aid in finding all the persons the user is looking for. Additionally, it is worthwhile exploring NER for this task because it can be used for a wide range of other applications. Enriching data with NER tags, for example, can help identify patterns and trends in the data, such as the geographical distribution of individuals with a certain occupation or the changing popularity of certain names over time.

Even though the biographies from the biography portal have already been digitized, they still provide challenges for a trained NER model. It is especially difficult to make predictions on older texts. This is for the most part due to the differences between modern day Dutch in comparison to 17th and 18th century Dutch, which is exacerbated in script. 'Opzigtelyk' (meaning something that easily comes to attention) for example, is a word that is no longer recognized in modern Dutch. If it were to be orthographically consistent with modern spellings however, it would be spelled 'opzichtelijk'. Such differences in spelling are accompanied by other aspects of language that change over time, such as grammar and lexis. For example, there may be differences in the way that certain grammar rules were applied in 18th century Dutch, which could make it

difficult for modern day Natural Language processing models to accurately interpret the texts. There are also phenomena such as semantic drift, as the meaning of words over time is also subject to change. "Kapot", for example, originally meant "broken" in Dutch, but over time its meaning has shifted to also include "exhausted" or "drained". For Natural Language Processing models, this is difficult to take into account, as a language model trained on more modern texts does not take phenomena such as semantic drift and orthographic changes into account.

Besides diachronics there are also other concerns when using language models on older texts, namely due to limitations of the systems that convert them to a digital medium. Many older texts are converted to a digital form through the means of Optical Character Recognition (OCR). OCR is a technology used to convert scanned images of text, such as a scanned document or a scanned page of a book, into a digitized form, that is editable and searchable. The process involves analyzing the image to identify the characters and then converting them into a machine-readable format, such as ASCII or Unicode. OCR software can be used to process text in multiple languages and can even recognize handwriting.

One issue however, is that handwriting is still difficult for machines to interpret. Especially because it manifests in many variations, with each person having different handwriting. Handwriting can for example be cursive or not, and there are many variations in each style that lead to unique handwriting. In some cases, even people struggle to read handwriting. In the same way that people have difficulty interpreting handwriting, written text is also difficult for OCR systems to consistently interpret, and therefore models make mistakes when transcribing some of this data to digital form. Because of this, OCR systems often produce data that is filled with noise, such as unwanted symbols, letters and punctuation.

Because of these limitations with OCR, a large effort has led to the manual transcription of Biography Portal data, resulting in clean text that can be effectively used by language models. Such clean data was used for NLP projects such as BiographyNet; which aimed to use connect the people mentioned in these biographies to biographical texts from other sources [Fokkens et al. \(2018\)](#). This four year project which lasted from 2012 to 2016, has demonstrated that the quality of the data is sufficient for NLP systems to work with.

With regards to language models, which this thesis will use in order to explore the NER task, the current common solution is to use a transformer based model. A transformer is a neural network architecture that captures the context and meaning of words. This is done via a mechanism called self-attention which measures the importance of each word. The measurement works through a dot-product operation between the input embeddings and attention weights. The training task determined how the weights of these models are updated.

These transformer based models are first pre-trained to create a contextual representation of the data. Pretraining is done on unsupervised tasks, an unsupervised task does not require any manual annotation. One common unsupervised task for example, is having a model try to predict a masked word in the sentence. After pre-training, the

models often need to be adapted to perform a target task. This adaptation is called fine-tuning, where an extra training task is supplied that helps the model learn what it should ultimately do. This can be a task such as NER.

Many of the well known transformer models are publically available. Another characteristic of these models is that they have millions of parameters. This is the result of pre-training these transformer based systems with a large and often costly dataset. One example of such a model is BERT. BERT is pretrained on the unsupervised tasks where it has to predict tokens that are randomly masked in the sentence, in addition to predicting the correct next sentence. If this BERT model would need to perform the NER task, it would need to be finetuned with another training partition that contains NER labels.

The pretraining process is very costly. The base BERT model contains 110 million parameters resulting from pre-training on over 800M tokens. Retraining such a model is therefore an expensive process but can be worthwhile if the target data is sufficiently different. Specialized models exist for domain specific use such as GysBERT (Manjavacas and Fonteyn, 2022), which is a model that is pretrained on the Dutch Historical domain for example. A more general version of a Dutch BERT model is BERTje, which was pretrained on a variety of books and multi-genre reference corpora.

In this thesis we explore the finetuning of both BERTje and GysBERT on historical publicly available OCR data. We then evaluate their performance on a test partition of that same data in addition to a high quality dataset containing biographical data.

We formulate two research questions that will guide our investigation.

- RQ1: How does BERTje finetuned on a generic dataset perform on the NER task in comparison to out-of-the-box tools such as Stanza and Flair?
- RQ2: How well does BERTje finetuned on historical data perform against GijsBERT that was pretrained on the historical domain?

Stanza and Flair are the out of the box tools that we will explore. With Flair using document level embeddings (FLERT) and Stanza being built with a neural network for NER classification. BERTje is based on BERT which stands for Bidirectional Encoding Representation of Transformers. The baseline BERTje model that we will use is an already finetuned model for NER that was trained on a generic dataset (CoNLL), and we will compare this to our own model that is finetuned on in-domain data.

We hypothesize that the BERTje model, finetuned on historical data, will surpass the performance of the Stanza and Flair models because of its WordPiece tokenization. WordPiece breaks down words into subwords based on the most frequent substrings in a corpus. The tokenizer maintains a vocabulary of subwords, and the input text is converted into a sequence of subword tokens. Because out of domain data will likely yield new tokens, we predict this to be an advantage for BERTje.

Regarding the second research question, we anticipate that the GijsBERT will surpass the performance of BERTje and that this will be exacerbated on historical data.

We believe this to be the case because it is pre-trained on historical text.

We will present this thesis by first outlining the relevant work in the theoretical framework, we then discuss our data and experimental setup in the methodology section, and then present the results, discussion and conclusion.

Chapter 2

Theoretical Framework

This thesis aims to explore the performance of various machine learning models on the task of Named Entity Recognition (NER) in Dutch text. This theoretical framework serves the purpose of providing relevant background information on this topic.

We will introduce the topic by first explaining what NER is. Furthermore we discuss approaches that can be used for coming to such a system where we predominantly go over the algorithms that were used for this task in the past.

We then go over the systems that are used in the project. It is important to mention that the project involves training two models and testing two out-of-the-box tools that can be used for NER. Considering that our systems will need to perform this task on Dutch text, we also choose to discuss some of the linguistic history of this language and its present-day form. This is especially relevant because the training of the models is purposed to help them adapt to this domain.

2.1 The task of Named Entity Recognition

NER is a task in natural language processing that involves the identification of real world things. This is typically done by giving labels to proper nouns that belong in a category. Popular NER datasets use tags such as persons, locations, and organizations. Other data may choose to specify different categories such as time or book for example. These categories are often different depending on the task, with fields that apply NER for the biomedical domain using a different subset of categories in comparison to what historians might use.

One of the earliest shared tasks made for NER is the CoNLL 2002 initiative (Tjong Kim Sang, 2002). A shared task can be interpreted as a benchmark for data-science; where a standardized dataset is used in order to evaluate the performance of models. CoNLL provides test and validation data to measure this performance. Throughout the years, the performance of many systems have been assessed on CoNLL, and it remains commonly used today.

Next to datasets, CoNLL also standardized a method of annotating data, namely in the BIO (or sometimes referred to as OIB) scheme. This is a word-level style of an-

notation, where every word receives a label. If the word falls in no particular category, it will receive the 'O' label. If it does fall into a category, it will be marked by a 'B' if it is the onset of that category, or an "I" if it not the first word within that category. This is then followed by the appropriate label. This is further illustrated in the table below.

Token	Label
My	O
name	O
is	O
Felix	B-PER
den	I-PER
Heijer	I-PER

Table 2.1: An example of BIO scheme annotation for Named Entity Recognition

2.2 NER before Transformers

As briefly mentioned before, transformer based models are currently leading to state-of-the-art performance on many NLP tasks. Before we discuss these models however, we want to also introduce some more traditional approaches. These traditional approaches for NER primarily relied on combination of heuristical and statistical methods. Two well known statistical approaches that introduced some relevance for even more modern architectures are Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs).

HMMs are generative models that can be used for a variety of tasks such as part-of-speech tagging, NER, and syntactic parsing. More generally, HMM's are used for tasks with sequential classes. HMMs are based on the Markov assumption that the probability of a sequence of states depends only on the current state and the previous state. Because of this a 'first order' hidden markov model would only look at the previous token and label to predict the next class. HMMs are composed of two main components: the observation model, which describes the probability of observing a sequence of words given a sequence of states, and the state transition model, which describes the probability of transitioning from one state to another. Similarly to the later discussed newer architectures, HMM's train by trying to predict a class for a masked token. That is in the case of NER, by trying to predict whether the tag for the next word in the sequence. The performance of HMM's however is limited due to its sequential assumption. When people process language they are able to hold a sequence into working memory and relate back to it to develop understanding (Schoonen et al.) [p. 5], but by modeling language sequentially this can not be taken into account. Discriminative models such as CRF's attempt to solve that problem.

CRFs are a type of discriminative model that can be used for tasks such as sequence labeling. CRFs model the likelihood of a sequence of labels occurring when given a sequence of observations. CRFs are particularly useful for tasks such as NER, where the goal is to identify and classify named entities within text. CRFs can be used to

model dependencies between labels, such as the probability of a word being labeled as a named entity given the labels of the previous words in the sequence. Both HMMs and CRFs have been used for NER, but CRFs have been found to perform better in most cases. CRFs have been shown to be more robust to errors in the training data and to perform better when the data is limited (Finkel et al. (2005)).

NER systems could combine these CRF and HMM models and improve their accuracy with heuristical knowledge. One example of such knowledge would be through gazetteers. A gazetteer is a list of known named entities, such as people, places, and organizations. Gazetteers can be used to identify named entities by looking up words in a text and seeing whether they co-occur in a gazetteer. Having a gazetteer containing common names for example, can make it easier for a system to identify people.

Regular expressions are another tool that can be used for heuristics based NER. These are basically 'advanced search queries' that are good for looking up specific patterns such as dates, email-addresses and phone numbers. The date format "MM/DD/YYYY" would for example look like "\d2/\d2/\d4". Where the regular expression is initialized with the first backslash, where 'd' represents that it is looking for a digit, and where the number thereafter specifies how long this sequence of digits should be.

These methods will not extensively be addressed in this project, but are common approaches nonetheless. The majority of the systems used in this project are based on neural networks.

2.3 Neural Networks

Neural networks address the fundamentals behind modern machine learning algorithms, before we can explain transformers, it is important to first understand what neural networks are and how they work. Neural networks are based on neurons, which is a node in the brain that can send a synaptic signal if activated. Neural networks mimic this principle by using arrays of artificial neurons. The artificial neuron takes an input and processes this in order to produce an output (Goodfellow et al., 2016).

The basic structure of a neural network involves a set of artificial input neurons, connected to several more layers of neurons that have 'hidden' states, and then an output layer. Hidden states are states that can not be observed or adapted manually. The input layer will take the information you would like to process, which in the case of NER is the information associated with words. The hidden states process that information, and the output layer will yield a result; so in the case of NER would yield a label or sequence thereof.

Neural networks are typically trained based on an algorithm called Stochastic Gradient Descent (SGD). Each neuron is initialized with a weight, which in the beginning is set to random, unless specified to be otherwise. The basic idea behind SGD is to iteratively update the weights of the connections between the neurons in the direction of the negative gradient of the error function (Ruder, 2016). This means that a model will update weights until it has minimized 'loss' or in other words, minimizing the

errors it makes. The gradient of the error function tells us the direction in which the error is increasing, and by moving in the opposite direction, we can decrease the error (Ruder, 2016). These neural networks are the learning algorithms that lie behind the transformer based systems that are used in this project.

2.4 A transformer based Neural Language Model; BERT

In this section we get into our first transformer based model, which is the one that we address in this project. It is called BERT. BERT stands for Bidirectional Encoding Representation of Transformers. The concept of transformers refers to an architecture that involves two main components, the encoder and decoder. The encoder is a representational system for the language, where words and how they relate to one another is inferred by the model, whereas the decoder is a productive component capable of producing sequential output.

Before BERT can apply statistical techniques, it builds a lexicon of words that it sees in the pre-training data. In order to increase matches with words, and to ensure that future queries will fit into this lexicon, it splits words into something called wordpieces. These wordpieces are meant to take Out Of Vocabulary (OOV) words into account (Devlin et al., 2018). One example of how this is done can be shown with a word such as “inconsequential” where it can be split as ‘in’, ‘##con’, ‘##se’, ‘##quen’, ‘##tial’. In many other cases however, BERT chooses to not tokenize a word into these subwords. BERT Frequently used words are not subword tokenized, whereas rare words are split into subwords.

Because BERT is a representation of encoders only, it only understands the relationships between words in a language, but does not yet have the ability to produce any information on its own. It therefore solely computes a numerical representation for a sentence and the words within it. It does this by first calculating the importance of related words through a mechanism called self-attention.

Self-attention works by first creating a set of “keys,” “values,” and “queries” for the input sequence (Vaswani et al., 2017). The keys and values are used to represent the input, while the queries are used to determine which parts of the input are most relevant for a given prediction. The attention mechanism then computes the dot product of the queries with the keys, and applies a soft-max function to the resulting values to obtain the attention weights. These attention weights are then used to weigh the values when computing the weighted sum that is output by the attention mechanism. One key property of self-attention is that it allows the model to attend to any part of the input sequence, rather than just the parts that are adjacent in the sequence (Bahdanau et al., 2014). Because of this it is better able to take semantic processing into account, where it can be argued that these attention mechanisms somewhat mimic the role of human processing in language processing.

After computing self attention, the BERT model trains itself based on two tasks; masked language modeling (MLM), and next sentence prediction (NSP). In the MLM task, a percentage of the tokens in the input are randomly masked (replaced with a

special [MASK] token), and the model is trained to predict the original values of the masked tokens. This task is intended to train the model to understand the context in which words appear, as it must use the surrounding words to predict the missing token. The NSP task involves predicting whether a sentence B follows another sentence A in the input. This task is intended to train the model to understand the relationship between sentences, as it must determine whether sentence B is a valid continuation of sentence A.

After MLM and NSP the weights are saved and can be passed onto the next encoder. Once all encoders have processed their weights, the pre-training process is complete. In order to then use a BERT model for a specific task it can then be finetuned for that task.

With regards to NER, the BERT architecture has led to state of the art performance. With the current best model performing NER on the CoNll-2003 dataset being a modified variant of BERT [Baeovski et al. \(2019\)](#). For the purpose of our project, we use two variants of BERT that work on Dutch text; BERTje and GijsBERT.

2.4.1 BERTje; a Dutch BERT Model

The first developed BERT models were multilingual models, though the pre-training technique could of course also be used for training language-specific BERTs. BERTje is an example of such a language-specific BERT. It is essentially a BERT model that was pre-trained on Dutch data, allowing it to achieve better performance on Dutch texts [de Vries et al. \(2019\)](#). This makes BERTje more interesting than a general purpose multilingual version for the current project, as we solely address Dutch texts.

BERTje was pretrained on news articles, books, and Wikipedia data. Additionally, the authors of BERTje have already fine-tuned this model on a wide range of NLP tasks, and has been found to be effective in improving the performance of Dutch NLU systems [de Vries et al. \(2019\)](#).

BERTje was primarily pre-trained on Dutch literature (4.4gb), followed by news corpora (2.4gb), a multi-genre reference corpus called SoNar 500 (2.2gb), news websites (1.6GB), and wikipedia data from 2019 (1.5GB). The pre-training procedure was slightly adapted for BERTje, after some criticism that was written on the original BERT architecture. The NSP task for example, was adapted so that the model would not only try to predict the next sentence, but also the previous sentence [de Vries et al. \(2019\)](#). Instead of NSP this is called Sentence Order Prediction (SOP) where the model is tasked to say whether the sentence is the previous or the next iteration. The MLM task was also adapted in the pre-training of this model. Not only as the model trying to predict the next wordpiece, but it aimed at predicting consecutive wordpieces belonging to the same token.

In order to evaluate the model they first fine-tuned it to be able to do three NLU tasks; NER, Part of Speech tagging, and Semantic Role labeling. For the purpose of this project we will focus on their NER results. The NER performance was evaluated on CoNLL 2002 data. CoNLL is a well-known format where each token has a label annotated with Beginning Inside Out (BIO) scheme annotations. This means a word

will get assigned the label O if it does not get a label, a 'B' at the onset of a label, and an 'I' for a consecutive token that is assigned the same label that was given at the onset of the entity. CoNLL is a shared task that serves the purpose of a benchmark as many models are tested on this data [Tjong Kim Sang \(2002\)](#).

The performance of multilingual BERT was at 80.7% for the NER task and increased with the BERTje version to an f1 score of 88.3%. The state of the art at the time of BERTje development achieved an f1-score of 90%. This was achieved with a well fine-tuned multilingual BERT model for NER [Wu and Dredze \(2019\)](#). [de Vries et al. \(2019\)](#) propose that applying their fine-tuning approach and scale to BERTje would likely have led to even better results.

The fine-tuning approach used by [Wu and Dredze \(2019\)](#) involved adding a linear classification layer with softmax on top of multilingual BERT. When masking predictions, they mask the post-initial wordpieces, considering the word-level labels that CoNLL uses.

In this project we choose to use this BERTje model as one of our base models that we can further finetune on with domain specific data.

2.4.2 GijsBERT, a BERT model for Historical Dutch

GijsBERT is another example of a pre-trained BERT model that we can use for fine-tuning purposes. GijsBERT distinguishes itself because it is trained on more historical Dutch texts. It is important to mention that the Dutch data used in this thesis can originate from as far back as the 18th century. This makes it extra challenging for language models to deal with, considering that the majority of their training data is likely modern as a limited amount of information has been transcribed from this time period.

GijsBERT is a very recently developed BERT based model that was developed by researchers from Leiden University. As mentioned, it aims to better handle historical Dutch data, and the problems that often come along with it, such as noisy OCR instances for example [Manjavacas and Fonteyn \(2022\)](#). As of to date, it is the only Dutch BERT model that has been specifically developed to handle historical data.

Architecturally, GijsBERT is similar to the original proposed BERT model from [Devlin et al. \(2018\)](#). With this is meant that it is also pretrained with the NSP and MLM objectives. The main difference of course with GijsBERT to this other model stems from its pretraining data. The pretraining data for GijsBERT comes from two large databases. These databases are Delpher; containing primarily historical newspapers, books and journals, and the Digital Library of Dutch Literature (DBNL); which contains digitized literature from Dutch and Flemish libraries. It is important to note, that the contents from Delpher, are primarily OCR'ed, whereas the content from DBNL was transcribed by linguistic researchers, resulting in high quality historical data.

Because some of the content was not of sufficient quality, most notably due to OCR, part of the work from [Manjavacas and Fonteyn \(2022\)](#) consisted of training another

model to detect text that was of insufficient quality. This was a character level model called *KenLM* trained on a part of the higher quality DBNL data to determine quality estimates. Ultimately, this model was used to determine what selection from Delpher was suitable for use.

The resulting set for pretraining GijsBERT consisted of 7.1B tokens. 5.8 of these were from Delpher and 1.3 were from DBNL. The dates from which these texts originated range from 1500 to 1950.

2.4.3 Flair

Alongside these BERT models the current project also addresses two out-of-the-box tools. One of them is called Flair.

Flair is a Natural Language Understanding (NLU) model developed by Zalando Research. It is a fairly modern framework for building state-of-the-art NLP systems. Flair allows users to easily apply pre-trained embeddings, such as BERT or ELMO, to their text data, and also provides a simple interface for training and evaluating models on a variety of NLP tasks, including named entity recognition, part-of-speech tagging, and text classification.

With regards to NER, the authors of Flair state an important problem; namely the problem of underspecified contexts. Classifying 'Baskel' in a sentence such as

- John Baker (Netherlands) Baskel
- B-PER I-PER B-LOC B-ORG

would allow for it to be interpreted as either a person or as an organization, assuming the word is not already present in the models lexicon. The paper from [Akbik et al. \(2019\)](#) proposes a solution to this problem where they opt for Pooling Contextualized Embeddings. A contextualized embedding is what is already produced by models such as BERT, it is a contextual representation of a sentence. Pooling combines these representations to form a new one that can be given to a NER classifier to give it additional context in relation to the rest of the text.

More specifically, the method works by holding each instance of a word in memory, next to its context and updating the pooled embedding at every new occurrence of the word in the data. In the end, the original contextual embedding is concatenated to the pooled embedding, preserving the original context. This process is shown in Figure 1.

This process is repeated and memory is cleared at every epoch during training. On the CoNLL 03 dataset, performance of this model (Flair) was compared to that of other models such as BERT from [Devlin et al. \(2018\)](#). On the NER task Flair achieved a f1 score of 93.18, which outperformed BERT by .32%. Because of the impressive performance for Flair on this task, and its up-to-date libraries, we chose to include it in this project.

Algorithm 1 Compute pooled embedding**Input:** *sentence, memory*

```

1: for word in sentence do
2:    $emb_{context} \leftarrow$ 
        $embed(word)$  within sentence
3:   add  $emb_{context}$  to  $memory[word]$ 
4:    $emb_{pooled} \leftarrow pool(memory[word])$ 
5:    $word.embedding \leftarrow$ 
        $concat(emb_{pooled}, emb_{context})$ 
6: end for

```

Figure 2.1: Example of how embeddings were pooled [Akbik et al. \(2019\)](#)

2.4.4 Stanza

The second out of the box tool that we address in this project is called Stanza.

Stanza is a Python natural language processing (NLP) library developed by the Stanford NLP Group. It provides a suite of NLP tools, for this project the most notable of which is a NER module that classifies persons, locations, organizations along with miscellaneous entities. The NER module uses a bidirectional long short-term memory (LSTM) recurrent neural network (RNN) to encode the input text, and a conditional random field (CRF) layer to make predictions about the named entities present in the text [Qi et al. \(2020\)](#).

A bi-LSTM is a type of LSTM that processes the input sequence in both forward and backward directions, which allows it to capture contextual information from both the past and future of each word. This is useful for NER, as the named entities in a text may be influenced by words that come before or after them.

The encoded text is then passed through a CRF layer, which makes predictions about the named entities present in the text based on the encoded input. The CRF layer uses the encoded representations of the input words and the transitions between them to make these predictions.

In addition to the NER module, Stanza also includes tools for other NLP tasks such as part-of-speech tagging, dependency parsing, and language modeling. One of the reasons that Stanza was included in this project, is because its NER system has been created via some of the aforementioned traditional machine learning approaches.

2.5 Domain Adaptation

The current project addresses Dutch data that can stem from many different time periods. Because of this we need to try to get the models that we train to also perform well on older Dutch text. This can be addressed via Domain Adaptation.

Considering that artificial intelligence based NLP models train based on a vocabulary, an important aspect to consider is whether the vocabulary used to train the model overlaps with the vocabulary of the target. Additionally, the distribution of the data might be different; certain words or grammatical features in one domain should be regarded as more important in comparison to those same features in another domain. Data sourced from social media for example, will use different words and grammar in comparison to data sourced from news articles. In order to better help models account for these domain differences, domain adaptation techniques are implemented.

Feature based methods of domain adaptation involve transforming the features of the data in the target domain to align with the features of the data in the source domain (Ganin and Lempitsky (2015)). In other words, a grammatical feature such as part of speech might be more important in a NER classification task in one domain, in comparison to NER in another domain, this needs to be taken into account. Feature based domain adaptation is typically performed by observing the features in both domains, and then by applying methods to transform the features that manifest differently in the training domain to the target (Ganin and Lempitsky (2015)).

Instance-based domain adaptation is also common. Instance-based domain adaptation can be performed by finding instances in the training data that match the target domain. Adaptation can thus occur by training on the union of test sets, giving less importance to, the tokens that do not occur across domains (Daume III and Marcu (2006)). If one domain for example involves social media data A , and the target involves news articles B , we would want to give less importance to all the elements that do not occur in $A \cup B$. This could be applied in NLP by, for example, finetuning a language model on only instances that occur on both domains.

Another approach is called self-training. In self-training the models own predictions are used to generate new training data (Liu et al. (2019)). The new training data, which is then partially generated by the model can be used to further improve the models performance, as this technique, despite lacking a certain standard of label quality, allows models to increase their lexicon and thus often also improves performance.

Weight transfer is another method that can be used for domain adaptation. Weight transfer involves transferring the learned parameters of a model trained on the source domain to a new model for the target domain (Ganin et al. (2016)). The idea behind this method is that the shared features between the source and target domain will be captured by the learned parameters of the model and can be reused for the target domain.

2.6 Domain Adaptation for BERT

Considering that models such as BERT already require adaptations in order to perform NLU tasks such as NER, one simple way to adapt to the target domain is to start the fine-tuning process with similar data. By doing this, you retain the models understanding from the pre-training phase, and help it adapt to the target data while it simultaneously learns a new task.

The SciBERT initiative for example, was aimed to provide a version of the BERT base model that was deemed more suitable for scientific tasks. (Beltagy et al., 2019) compared the difference between training a BERT model with scientific data from scratch, with finetuning a version of BERT-base that with a smaller subset of in-domain data. They achieved the best results with pre-training a model on in domain data, however fine-tuning also improved performance for all tasks.

Another approach added in conjunction to fine-tuning for BERT was adversarial training. This first involves fine-tuning a model for a task such as NER, and then training an adversarial model which serves the purpose of trying to fool the finetuned classifier. It fools the classifier, because it is trained to predict instances where the finetuned NER classifier makes mistakes. The adversarial model then helps create new training examples that are difficult for the fine-tuned NER classifier to take into account, which are labeled to further improve performance (Ebrahimi et al., 2021). The adversarial task can be applied for domain adaptation by helping it use target-domain instances where the NER model is expected to make a misclassification.

2.7 The Historical Dutch Domain

Because some of our target data is written in historical dutch it is also relevant to provide some background on phenomena such as semantic drift along with changes in the orthography. Discussion regarding change in the Dutch language over time typically has refers to either old-Dutch, middle Dutch or modern Dutch. In order to summarize this component, we will discuss changes in the declension system along with orthographic changes that occurred from middle-Dutch to modern-Dutch.

Middle Dutch refers to Dutch spoken between 1100 and 1550. Four grammatical cases were known in the Dutch language in middle Dutch however only remnants of it exist today. These cases are the genitive, nominative, dative, and accusative. In the 17th century these case systems did not exist in the productive form, but in writing, they were often still used. The use of the case system in writing lasts up until the 19th century, and remnants of it still exist in modern dutch today. Presumably its existence was extended for so long, due to prescriptive efforts of grammarians to preserve the case system.

The middle Dutch case system is characterized by a complex system of inflections which indicate the grammatical function of nouns, pronouns, and adjectives in a sentence (Hüning and Vogl). All cases are marked morphologically through a suffix. As for nouns, it is important to note that the inflection rules change dependent on whether the noun is categorized as 'strong' or 'weak'. A singular noun ending on -e for example, would typically fall under the 'weak' category, whereas nouns ending on a consonant are likely to indicate strong inflection. The table below from (Hüning and Vogl) shows some examples of these strong and weak inflections, and how they would be changed for each case. As can be seen from the table; the expression of gender, strength, case, and number of articles are all dependent on the noun.

Masculine		strong	weak
Sg. Nom	die goede	gast	mensche
Gen	des goets/goeden	gast(e)s	menschen
Dat	dien goeden	gaste	mensche
Accu	dien goeden	gast	mensche
Pl. Nom.	die goede	gaste	menschen
Gen.	der goeder	gaste	menschen
Dat	dien goeden	gasten	menschen
Acc.	die goede	gaste	menschen
Neuter		strong	weak
Sg. Nom	dat goede	hof	herte
Gen	des goets/goeden	hoves	herten
Dat	dien goeden	hove	herte
Accu	dat goede	hof	herte
Pl. Nom.	die goede	hove	herten
Gen.	der goeder	hove	herten
Dat	dien goeden	hoven	herten
Acc.	die goede	hove	herten
Feminine		strong	weak
Sg. Nom	die goede	daet	siele
Gen	der goeter	daet/dade	siele(n)
Dat	der goeder	daet/dade	siele(n)
Accu	die goede	daet	siele
Pl. Nom.	die goede	dade	sielen
Gen.	der goeder	dade	sielen
Dat	dien goeden	daden	sielen
Acc.	die goede	dade	sielen

Table 2.2: Middle Dutch case system from [Hüning and Vogl](#)

In the table from [\(Hüning and Vogl\)](#) we also get some sense of the Dutch orthography. In middle Dutch, there was not yet a formal orthographic system, and Dutch spelling was often characterized by how the words would sound phonologically. In 1804, a new spelling system was introduced as defined by Matthijs Siegenbeek [\(Siegenbeek 1804\)](#), which eventually was approved by the Batavian Republic. This introduced several orthographic changes that still exist today, such as the change of the 'y' to the long 'ij' form. It must be noted that this spelling system, despite being approved by the republic, did not achieve widespread use, as there were also many criticisms.

Later the Jan Frans Willems spelling was proposed, which was similar to Siegenbeek's system, but had a solution for some of the problems considering critique regarding how some of the diphthongs were to be spelled. There were, for example, discussions regarding how the diphthong /ei/ should be spelled; 'ei', or 'ey', amongst many others. [Vosters \(2009\)](#). This system would be officially adopted in the Netherlands in 1888.

The newer spelling continued to foster disagreement amongst linguists, as the /e:/ sound would have an open and closed spelling. The word 'heeten' uses the 'e' twice,

whereas 'lezen' applies it once, despite both resembling the same phoneme. This led to another reform, leading to a new spelling called; 'De Kolloewijn Spelling' (Kolloewijn, 1891), which was adopted in 1904. was adopted mainly in South Africa, where the system was simplified to unify such spelling differences.

Despite adoption in South Africa, the Kolloewijn spelling was not popularized in the Netherlands. Ultimately the Marchant system was a combination of the prior mentioned systems, which was proposed and adopted in 1934 Trouille (1987) and is very close to the modern day Dutch spelling system.

2.8 NLP In Historical Texts

One of the problems with applying language models on historical texts, described by Labusche et al. (2019), is that older texts are often less standardized and are often preprocessed with Optical Character Recognition (OCR) techniques which do not always provide accurate transcriptions. This would for example lead to noisy data, which would need to be filtered out to obtain best results.

Labusche et al. (2019) opted to fully pre-train a BERT model specifically for OCR'ed texts in historical-german. This was done with a similar method to the original BERT mode from Devlin et al. (2018). The difference however was that they trained it on more noisy OCR'ed data, though several amendments were made in order to reduce the noise. One of these noise reduction techniques was the removal of 'failed' OCR pages, where it was decided it was incomprehensible and not fit for training. This would commonly manifest through excessive punctuation in combination with several incorrectly transcribed graphemes. Ultimately (Labusche et al., 2019) found that pre-training on the historical German dataset improved performance in that domain, but worsened performance on other domains.

2.9 The Dutch Biography Portal Domain

The crux of this project is to evaluate how well a finetuned BERT model performs on Biography data. The target data of this project is within the biography portal domain. The dutch biography portal has collected a approximately 145,000 biographies from 80,000 individuals. The biographies were collected and organized by the Institute of Dutch History (ING), along with the joint initiative of a number of other organizations Renders (2023).

They claim to collect biographies from Dutch 'men' and 'women', which in their interpretation, means that they collect biographies from both people who were born in the Netherlands, but also biographies of people who were not born in the Netherlands but who became active in the area during their lives. The biography portal has described a selection of categories, in which the people described in their data fall. This can include nobility or famous artists, along with several other categories. There are also people whom do not fall into any of the categories that are placed in the miscellaneous category. Besides categories, additional metadata that is collected from the

biographies includes date of birth, date of death, gender, religion, and the full and family names of the described persons in question.

When indexing for a particular individual, the biography portal will link to the text in which the biography describing this person was found. This means that no biographies are rewritten, and primary source versions are retained, resulting in a combination of older, and newer texts.

Chapter 3

Methodology

This chapter discussed the methodology of this project. Here we discuss the data used in this project, domain adaptation, and the pre-and-post processing that is needed for our models to work with the data. This is followed by a description of the systems used in this experiment.

3.1 Data Overview

This project applies the task of Named Entity Recognition to two datasets. The first dataset was retrieved from IVDNT (Institute for the Dutch Language), the test is publicly available and called “AI Trainingset, Tag de Tekst for Named Entity Recognition (NER)” (Dutch Language Institute, 2022). It was produced by 150 volunteers from the “Tag de Tekst” crowdsourcingproject. The set contains labels for persons, locations, and time entities, and was sourced on texts originating between the 17th and 19th century. It involves notorial texts from the Dutch cities of Haarlem and Amsterdam along with documents from the Dutch East India Company.

The second dataset is inherited from the Dutch biography portal project (Renders et al., n.d.). As was explained in chapter 2, this includes 200,000 biographies containing information about approximately 80,000 individuals. The biography portal project aggregates biographical data from various sources and covers a wide range of time periods; ranging from the 18th to 21st century.

In this thesis, we will compare the performance of several NER models using the “Tag de Tekst” dataset as our evaluation metric. This procedure applies to the out-of-the-box models which include a baseline BERTje model finetuned on CoNLL, along with the most recent large distributions of Flair and Stanza. The reason for choosing this dataset is that there were no gold labels available for the biography data during the starting phases of this project. Additionally, since the “Tag de Tekst” data was labeled by human annotators we assume that it is of sufficient quality. We are however unable to verify this because the “Tag de Tekst” project was initiated through the means of crowdsourcing, there is no information such as inter-annotator agreement that can help us understand the true quality of this data.

One challenge when working with different models is that they may have different labeling conventions. For example, one dataset may include a TIME label, while a pre-

trained system may not classify such a label. In addition, the way entities are labeled can also vary. For example, one dataset or model may label a “person” entity as “Mr. Snow,” while a model may omit the title and simply label the entity as “Snow” or vice versa. To address this issue, we have made several choices regarding our selection of entities for evaluation.

3.1.1 “Tag de Tekst” Data

The data from Tag de Tekst consists of several subdocuments. Most, but not all, are used for the purpose of this experiment. It contains a training partition, a development, or validation partition, along with other partitions titled “Noord Hollands Archief” (NHA), “Regionaal Historische Centra” (RHC), and “Stadsarchief Amsterdam” (SA). Table 3.1 gives an overview of the amount of tokens and sentences in each partition. The sentences were segmented with a out-of-the-box tool called Spacy (Hon-nibal et al., 2020), where we loaded their nl-core-news-sm model, which is their smallest model (12MB) for segmenting sentences that was trained on news data. Because of this being automatically performed, it is possible that the amount of sentences may not be completely accurate. The original source does not provide much information regarding the origin of each specific partition, however it is known that they are all sourced from Dutch national archives.

Partition	Amount of tokens	Amount of sentences
test_NHA_cleaned	9730	1044
test_SA_cleaned	18296	2333
train_cleaned	1880206	256930
validation_cleaned	467089	53401
test_RHC_cleaned	523	57

Table 3.1: Overview of token counts for each partition in the tag de tekst dataset

From the table above it can be seen that the RHC partition is by far the smallest, only containing 57 sentences. Because of this, we have chosen to omit this particular partition from this project. Additionally, we see that train and validation are by far the largest subsets. Our test sets remain with a combined 28,000 tokens. More important than sole token count however, is the distribution of the labels. Table 3.2 gives an overview of the label distribution for each partition.

File	B-TIME	I-TIME	B-LOC	I-LOC	B-PER	I-PER
test NHA	109	255	252	154	352	483
test SA	255	428	257	54	781	847
train	32073	57807	47724	14475	98521	110330
validation	7310	13197	11027	3035	24213	26297

Table 3.2: Overview of label counts for each partition in the tag de tekst dataset

It is important to note that the “tag the text” data acknowledges three types of entities, “PER”, “LOC”, and “TIME”, considering that the cardinality of the union

of the “tag the tekst” set along with the label outputs for stanza and flair is only two; the labels “PER” and “LOC”, we have oriented this project towards the classification of only these two labels.

The most popular label in every partition is PER, and the distribution of labels is approximately the same for all partitions. The only real difference is that the RHC partition is the only partition to have more B-LOC labels than I-TIME labels. This though does not say anything about the amount of TIME entities in the dataset, instead, it suggests that the span of these TIME entities is longer in this particular partition. Because we are not interested in the TIME entity for the purpose of this project however, this is unlikely to become an issue.

In terms of distribution, it can be argued that the train partition should be paramount. This is because this distribution could explain biases that a model might develop when trained. Because the distributions of each label are approximately the same for every partition, inspecting the training data is in this case also insightful for the entirety of the tag-de-tekst set. Figure 3.1 describes the distribution of our training partition data.

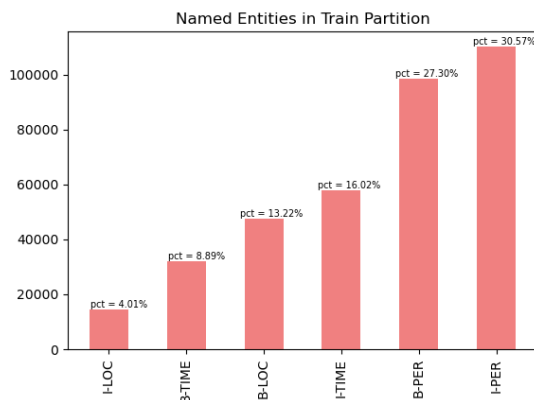


Figure 3.1: Bar plot of the label distribution in the train partition

From the 1.880 million tokens in this CoNLL style partition, we can see that almost 60% of the labels contain the “PER” label, in comparison to 8% tokens being assigned a “LOC” label. The majority of these “LOC” labels include names of cities. More specifically, The city of Haarlem is mentioned 1,798 times, whereas the city of Amsterdam and Batavia are also popular. As for the “PER” label, it consists of typical Dutch names as can be expected. An overview of the entities found in the “Tag de Tekst” data can be more clearly seen in the wordcloud from Figure 3.2.

The larger elements in the wordcloud from figure 3.2 indicate that they are more common. This figure thus shows us the most popular entities in the partition. Most notably, person names attract attention, though this is largely due to more person entities being present in the data, it is still somewhat surprising that Dutch cities are not marked larger than they are considering the origins of the partition. Additionally, names are a much more specific entity, which, especially for this dataset, typically

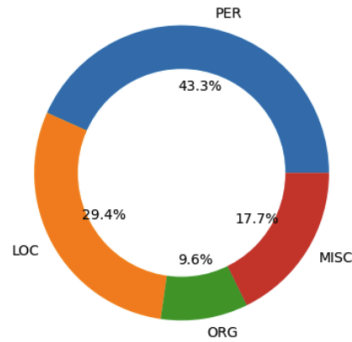


Figure 3.3: Stanza generated label distribution for the complete Biography set

raised. Whereas another domain, such as the legal domain for example, will talk about people with respect to the crimes they are accused of, not giving much importance to the cities in which they reside.

Further inspection of the biography labels have led us to notice that there is a skewed distribution, with some locations occurring more frequently than others. This is shown in Table 3.1.2. Amsterdam is the most common location, with over 55,000 occurrences, while other locations like Utrecht and Leiden also have relatively high frequencies. Nederland, Rotterdam, and Den Haag also occur frequently among the most frequent labels, while other locations like Groningen and Haarlem occur more sparsely.

Location	Count
Amsterdam	55217
Utrecht	21501
Leiden	18048
Nederland	13604
Rotterdam	13563
Den Haag	12997
Groningen	10764
Haarlem	10300
Antwerpen	7507
Dordrecht	6

The distribution of the most frequent person labels in the dataset also appears to be skewed, with some names occurring much more frequently than others. This can be seen in Table 3.3, which shows the top 10 most popular location entities. Showing more in an illustration would quickly become redundant, as the majority of locations only occur once. Jan and Kramm are the most common names, each with over 3,000 occurrences, while Willem and II also have relatively high frequencies among the most frequent labels. Maria, Johannes, and Vader also occur frequently among the most frequent labels.

Person	Count
Jan	3111
Kramm	3110
Wagenaar	2989
Willem	2771
II	2629
IV	2158
Maria	2124
Johannes	2046
Muller	2000
Kok	1898

Table 3.3: Most popular PERSON entities found in biography

Biography test partition

The biography data annotations were sourced by three annotators. Unlike other datasets in this project, the annotation guidelines are known, informing us what conditions were used for annotating each label. The subpartition of biography data that was annotated by humans contained 35,132 tokens and 2,264 sentences, making it the third largest human annotated partition in this project, behind the Tag de Tekst train and validation partitions.

Annotation Guidelines for BiographyNet test partition

The annotation guidelines were written in English, in the description it is stated that the annotators must annotate entities as a category within a set of all proper noun phrases in the text. The annotation guidelines are directed at labeling persons, locations, organizations, and artworks as possible categories. Considering the present project is directed at the classification of persons and locations, we will focus on describing the annotation guidelines for only these categories.

The category of persons is marked as PER. These entities are limited to humans, and should consist of only proper names referring to unique individuals. Titles should be included in annotation, if you have a person name with a preceding title for example, “Prof. Dr. Piek Vossen”, both “Prof” and “Dr” belong to this entity along with the first and last name that follows. Pronouns along with family referants should not be considered a person entity.

The annotation guidelines list several examples of person entities.

- Horner J. Simpson
- IBM Lawyers
- The Secretary of State
- Saudi Arabia’s Crown Prince

The category of locations is marked as LOC. Location entities consist of official locations, and should exclude region descriptors determined by social groups or communities. Locations can therefore include countries, cities, regions with an official name. Additionally, street and building names should also be considered a location, under the condition that an event is happening there.

The annotation guidelines list several examples of location entities.

- Soviet Union
- Haarlem
- municipality of Amsterdam

The annotation guidelines also describe special cases that should be considered when annotating. These describe coordination, name suffixes, coreference, ambiguity, and agency. When a sequence of titles are used in coordination for example, only the ultimate title should be used when annotating. If you have “Mr and Mrs. den Heijer” for example, “Mr” should be omitted from the person category.

Entity	Count
B-PER	1161
I-PER	1301
B-LOC	809
I-LOC	209

Table 3.4: Entities in the biography test set

In the test partition, we expectedly see a similar distribution of PER and LOC labels compared to the full biography data. This indicates that in terms of label distribution, our test partition is somewhat consistent with the intended target of our model. Figure 3.4 illustrates a wordcloud of these entities, where we can take a closer look at the annotated labels. Here we can see that names are still prominent and repeated in this dataset, such as “Charlotte Sophie”. Additionally, we also see ‘cities’ such as ‘Den Haag’ being popular entities. Notably, cities that were popular in the train partition such as ‘Amsterdam’ or ‘Haarlem’ do not appear, giving us an insight that the people described within this subset probably do not stem from those areas.

This implies that the distribution of locations, in terms of popularity, does not entirely align with our training data. Nevertheless however, it is experimentally still worthwhile to explore how the models will deal with these entities, given their different pre-training and origins.

The quality of the annotations is assessed in Table 3.5. Each iteration one annotator is taken as “ground truth” and then the F1 score of the other 2 annotators is measured against it as if they were classifiers.

Table 3.5 shows that the agreement is good to high for all annotators, suggesting that the quality of this set is sufficient for evaluation purposes.



Figure 3.4: Wordcloud for Named entities within the Biography test set

Annotator	3	2	1
3	1	0.9	0.8
2	0.84	1	0.8
1	0.74	0.79	1

Table 3.5: Agreement between the three annotators for the biography test partition

3.2 Domain Adaptation

Domain adaptation is typically performed to help a language model trained to make predictions on one type of data, also work when making predictions on another type of data.

This problem is closely related to the interdependence assumption, which states that the training and test data are drawn from the same underlying distribution. In other words, it assumes that features in the training data are distributed in the same way in target data of the model. This however, is in practice rarely the case, as the test data often contains information that was not seen by the model in training.

For this experiment for example, we use a BERTje model that was pretrained on Dutch literature and news, along with the SoNar-500 set. Our target domain contains data from BiographyNet, which consists of biography portal data with several other sources containing biographical information. Because we know that some of the biographical information is written in older Dutch, we know that the interdependence assumption does not hold, since BERTje’s training features (derived from the self-attention mechanism) do not contain older Dutch text.

Our solution to this is to therefore finetune the BERTje model on older Dutch text originating from Tag de Tekst. This is done because we assume that the data from Tag

de Tekst is closer to our biographical target.

Another model that we use in this thesis is called GijsBERT, this model deals with the other end of the spectrum as it is trained on historical dutch news, books and journals. GijsBERT will likely better fit the Tag-de-Teskt datasets as its pretraining data stems from similar time periods. Alternatively, it has less modern training data that it might encounter in the biography portal data.

Just like BERTje, we also finetune GijsBERT on the historical Tag de Tekst set, however this is not done for domain adaptation purposes, but solely for allowing the model to perform the NER task.

3.3 Pre- and post- processing

We used data from two different origins, “Tag de Tekst” and “Dutch biographical Portal”. The biography data was made available for use through a JSON, whereas the Tag-de-Tekst data was made available through several TSVs of which two partitions were selected for cleaning. Both partitions contained data that was labeled through CoNLL conventions, where every token is assigned a label, with each token placed on its own line. CoNLL format indicates that each label receives a “beginning”, “in”, and “out” indicator along with the corresponding label.

The biography data was delivered to us in JSON format, with metadata already containing various processed formats such as Stanza tokens and labels. Because of this, no additional preprocessing had to be done in order to use this data for our project.

The “Tag de Tekst” data did have some preprocessing steps, as there was for example no indication marking sentence endings, which is important for when we train our finetuned model. Because of this extra efforts had to be made for marking sentence boundaries. This was done by implementing a sentence segmentation tool from Spacy. After sentence segmentation, further inspection led to finding large arrays of text that were not segmented. This was presumably because the unsegmented chunks are transcribed documents from legal origin, or transcribed conversations that do not mark clear sentence boundaries. In order to combat this, some manual efforts were made to segment the largest chunks, which contained over 6000 tokens. Furthermore, a preprocessing system was written that segments any sentence containing more than 400 tokens into chunks of 130 tokens. The reason for segmenting in this way is because we wanted to reduce sentence size, and simultaneously, limit biases that would develop if we were to for example, split all sentences into chunks of 130.

After preprocessing, the systems were ready for use and generated output. Considering that not all of our system generated the same labels, some adaptations had to be made. The post-processing for the out-of-the-box systems such as Flair and Stanza involved removing some of the labels from the predictions. This was done in order to compute the evaluation metrics with only PER and LOC labels present, which are the only two named entity tags that are used by all systems.

For the baseline BERTje and the finetuned BERTje system, we reconstruct the

wordpieces back into tokens. Additionally, we applied some steps to ensure that the tokenization output was identical to that of the original data, if this was not done, it would be impossible to evaluate the performance of the model.

3.3.1 Models

In order to conduct our experiments, we propose several out of the box systems; Stanza and Flair, along with a baseline BERT based system; BERTje. Additionally, we finetune two models on historical data and evaluate its performance. The models we will finetune are BERTje and GijsBERT.

The Stanza system is evaluated without any major modifications. In order to compute the NER results we simply load stanza with the default pipeline for dutch 'nl' and have it make predictions on the datasets. The same is done for a tool such as Flair, where we simply load it with the largest dutch pipeline available "ner-dutch-large" and have it make predictions.

The baseline BERTje is the same model that was used for the experiments from [de Vries et al. \(2019\)](#), which was available on huggingface; a platform used to store datasets and trained machine learning models. As mentioned by [de Vries et al. \(2019\)](#) this model is BERTje finetuned on data from CoNLL 2002. Initializing BERT based models requires two important aspects, a tokenizer, and the weights and vocabulary of the finetuned model. As the names suggest, the tokenizer performs tokenization, splitting the words into wordpieces for BERT to predict, and the finetuned model can look at these wordpieces and use the learned weights to make predictions.

Some slight amendments had to be made in regards to re-aligning the output of the BERTje models. This is because BERTje uses wordpieces, so it makes a prediction for each wordpiece, though in the baseline BERTje model, the tokenizer had a different vocabulary size than the finetuned model, meaning that they would not agree upon the wordpieces to split on; resulting in an error. Because of this we cloned the model from huggingface locally and amended its vocabulary manually. We removed 78 subtokens from the vocabulary of the tokenizer, which mostly consisted of letter markings (such as an umlaut) and other symbols. After this both the tokenizer and the finetuned model had 30,000 tokens or subtokens in its vocabulary so that the predictions could be assigned correctly.

Our finetuned BERTje model is the base BERTje finetuned on the "Tag de Tekst" train partition. For the finetuning process we implemented the same proposed strategy from the original BERT authors [Devlin et al. \(2018\)](#). This is where we apply a linear feed forward neural network layer on top of the embeddings produced by BERTje. There are several parameters of interest: The learning rate tells the model how aggressively to update weights, a high learning rate, would for example mean that it will aggressively change the weight and thus how important certain tokens are. Too high of a learning rate would typically result in the model not learning anything, or even becoming worse during training. A low learning rate on the other hand, would cause the model to learn too slowly, or learn nothing at all. Our selected learning rate was 1e-5, which, for most purposes, is considered to be low. This learning rate is passed

to the second important parameter, called the optimizer. For this project we used the Adam optimizer. This is typically known to be a safe option, because it is capable of adjusting the learning rate during training (Kingma and Ba (2014)). We then train the model for the NER task on our tag de tekst data for eight epochs with a batch size of eight. The batch size refers to the number of samples used to update the model parameters during each iteration of the training process. The epochs refer to the amount of times that the model will see the complete training data.

3.3.2 Experiments

Our research questions are pointed at investigating the performance of our systems. In order to evaluate the performance we will assess this with quantitative and qualitative methods.

Some models required choices to be made. The out of the box flair and stanza models, there were different vocabulary options, for both models we picked the largest available vocabulary. The baseline BERTje model did not leave room for any choices, as it was the only CoNLL finetuned BERTje model available on the huggingface repository from the authors of (de Vries et al. (2019)). For our own finetuned BERTje however, we created choice by training it on three random seeds. These seeds determine the starting weights, allowing our experiment to be reproduced. We used '1234500', '1482500', and '4365691' as seeds for our three versions. During training we save a version of each model after each epoch. We are then left with eight versions (one for each epoch) for each of our three models. Model selection then requires us to pick the the best model seed and epoch. We perform such selection by graphing the training and validation losses. Training losses determine how well the model is fitting the training data, whereas validation losses give some indication on the extent to which a model is fitting unseen data. In our case, our training losses will be based on the Tag de Tekst train partition, whereas our validation loss is determined by the Tag de Tekst validation partition. We then select the model that has the lowest training or validation loss that has seen the most number of epochs as the best. During the span of this experiment, the GijsBERT model was introduced to this project. The GijsBERT model was finetuned with the same procedure as BERTje, however only one random seed was used. This made the selection process simple as it just involved taking the epoch with the best validation loss.

After model selection we perform our quantitative analysis. For this, each model will be run on the "Tag de Tekst" test partitions: NHA and SA, in addition to the "Tag de Tekst" data, we also evaluate quantitative performance on our aimed target; the Biography Portal Domain. The dataset used for evaluating on the target domain is the seperate human-annotated biographynet partition. Quantitative evaluation involves presenting a classification report with the precision, recall, and F1 scores for each label. In this report we will remove the 'O' label, because it is extremely common in comparison to the other entities and not our main area of interest. In addition to this classification report, we create confusion matrices, which indicate which labels the model confuses for others. These two metrics will give us a general indication of how the model performs. Additionally though we also aim to understand the types of errors this model creates, which will determine our qualitative evaluation.

Our qualitative evaluation involves annotating an extra small subset of Biography Portal data. Because we do not know the origin of our already annotated Biography Portal partition, we choose to annotate a small partition of data. The annotation procedure is as follows: we select a very small subset of approximately 1000 tokens from old and new domain, and review misclassifications made by the BERTje finetuned on CoNLL, BERTje finetuned on Tag de Tekst, and GijsBERT finetuned on Tag de Tekst. An overview of the sources with known dates are given below in table [3.6](#).

Source key	Date written
blnp	1968-2006
bwg	1998-2009
bwn	1979-2008
bwsa	1986-2003
elias	1903-1905
glasius	1851-1856
knaw	1857
nbw	1964-2011
nnbw	1911-1937
vdaa	1852-1878
wikipedia	2000
weyerman	1729-1769

Table 3.6: All sources with known dates and the dates of their biographies

Our small qualitative evaluation subset will sample on articles from old sources ranging from 1729 up until 1878, and new sources ranging from 1900 to present. We will then evaluate the mistakes made by the our models.

The evaluation of mistakes will try to identify systemic patterns. This means that misclassifications that are similar, or are made more than once, will primarily be discussed.

This results in two new datasets seen in [3.3.2](#)

Table name	Number of tokens	Number of sentences
Qualitative test old	1411	116
Qualitative test new	1223	80

The old qualitative evaluation set contains 1411 tokens and 116 sentences, and the new qualitative evaluation set contains 1223 tokens and 80 sentences. With old we mean that the sources contained within are from 1878 at the newest. Annotations for these partitions were done by a single annotator with the same annotation guidelines as mentioned in [3.1.2](#).

Chapter 4

Results

Our results consist of two parts. The first of which is a quantitative assessment, where we will assess the performance of each system on the partitions from “Tag-de-Tekst” and on the annotated biographynet data. For this we will provide statistical insights about the models we trained, along with information regarding each systems performance on the data partitions. The second part involves a qualitative assessment, for which we which is our biographynet partition, to evaluate patterns in misclassifications that the model made.

4.0.1 Model Selection

Model selection only applies to selecting the best version of our finetuned BERTje models. This involves the three BERTje seeds. For comparison, we have also graphed our finetuned GijsBERT model to see how in performs in figure [4.1](#).

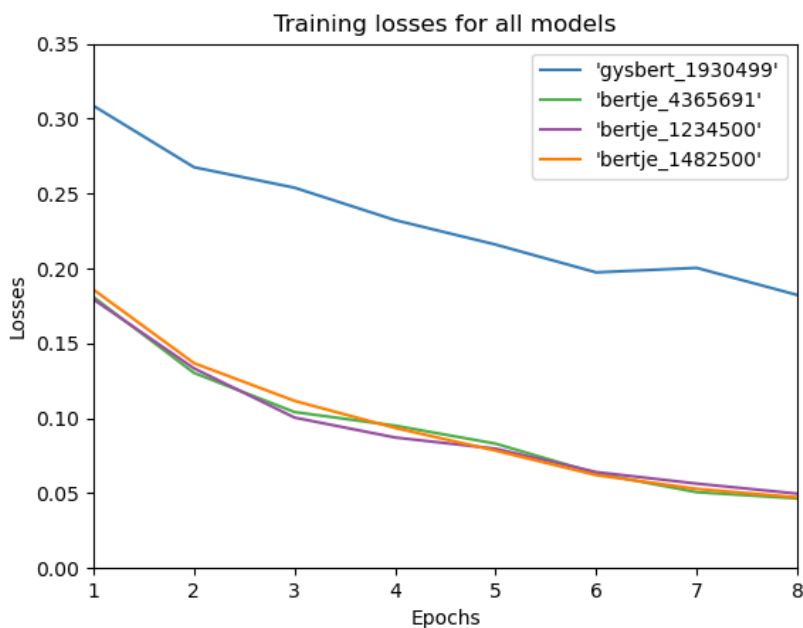


Figure 4.1: Training losses for each model

The training losses show that the models are all improving on the training set as the epochs continue. Since our goal during training is to minimize the loss function, this is a good sign.

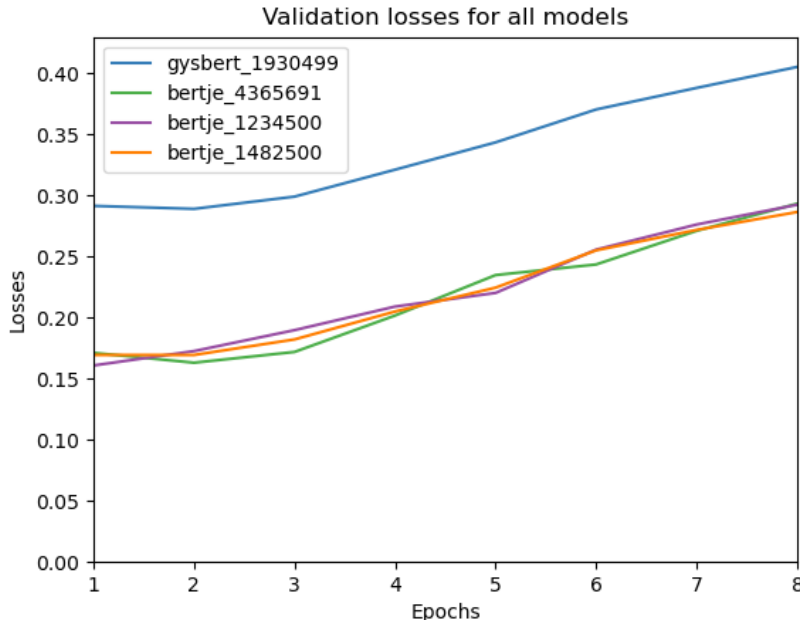


Figure 4.2: Validation losses for each model

When evaluating the losses on the validation partition however, we see that the models very slightly lower their loss for the first two epochs, and then start increasing their loss. Considering that the training loss is still going down, this can be considered typical for overfitting for the train dataset.

For our selection we noted that the BERTje model with the seed of '4365691' has the lowest validation loss, and therefore we have chosen to evaluate on this model.

4.0.2 Stanza Model

The first out of the box tool that we will evaluate is Stanza. The performance metrics on the NHA test partition from the Tag de Tekst set can be seen in table 4.1. For interpretation of these classification reports, we need to remember that precision refers to the amount of true positives from the model over the total number of predictions made by the model. Recall refers to how well the model is able to identify all positive samples in the dataset. It is the ratio of true positives to the total number of actual positives in the dataset.

On this Tag de Tekst test partition, we find that Stanza does not achieve high performance. The I-LOC label only achieves 0.238 precision and 0.064 recall. The B-LOC label is the highest performing label, with a precision of 0.638 and recall of 0.385. The B-PER label is the second highest performing label, with a precision of 0.566 and recall

Label	Precision	Recall	F1-Score	Support
I-LOC	0.238	0.064	0.102	154
B-LOC	0.638	0.385	0.480	252
I-PER	0.704	0.596	0.646	483
B-PER	0.566	0.545	0.556	352
Micro Avg	0.623	0.473	0.538	1241
Macro Avg	0.537	0.398	0.446	1241
Weighted Avg	0.594	0.473	0.519	1241

Table 4.1: Classification report for Stanza on NHA partition.

of 0.545. The highest performing label is I-PER, with a precision of 0.704 and recall of 0.596. The overall performance of Stanza on this test partition is 0.623 precision and 0.473 recall.

Overall, the classification report implies that Stanza does not perform well on this test partition. The precision and recall scores are relatively low, indicating that the model is not able to accurately identify the labels in the dataset. This could be due to the fact that the model is not trained on this particular dataset, and therefore does not have the necessary knowledge to accurately identify the labels.

In figure 4.3 we see that confusion typically occurs with B and I variations of tags. I-LOC is for example frequently confused with B-LOC and I-PER is typically confused with B-PER. It is important to note however that the I-LOC labels are also often confused with tags such as I-PER. One plausible reason as to why this might be occurring is due to LOC labels being more typically out of vocabulary for Stanza.

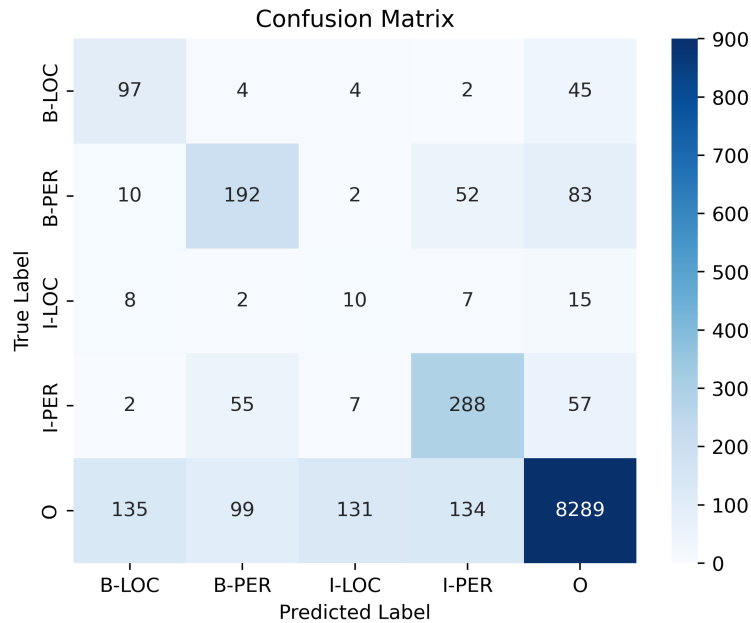


Figure 4.3: Confusion matrix for stanza on NHA partition

Label	Precision	Recall	F1-Score	Support
I-LOC	0.050	0.019	0.027	54
B-LOC	0.548	0.331	0.413	257
I-PER	0.700	0.649	0.674	847
B-PER	0.571	0.658	0.612	781
Micro Avg	0.618	0.593	0.605	1939
Macro Avg	0.467	0.414	0.431	1939
Weighted Avg	0.610	0.593	0.596	1939

Table 4.2: Classification report for Stanza on SA partition.

The performance for Stanza on the Tag de Tekst SA test partition is similar to its performance on NHA. The I-LOC label precision and recall are particularly low at 0.050 and 0.019 respectively. The B-LOC scores are slightly higher at 0.548 precision and 0.331 recall. This indicates that the Stanza model is better at identifying the onset of location entities and is comparatively worse at identifying its span. The PER labels achieve higher precision and recall. With I-PER precision and recall at 0.700 and 649 and B-PER precision and recall at 0.571 and 0.658. For all labels the precision score is higher than the recall, likely indicating that the model is conservative with choices other than O when making a prediction.

The confusion matrix for this partition, seen in [4.4](#) confirms this suspicion. The model confuses O more than other labels, though this is to be expected because O is also the most common true label in the dataset. In this partition we also see that there is a lot of confusion between B-PER and I-PER labels. Notably, the partition only correctly classified a single location, and was likely to confuse them for the I-PER label.

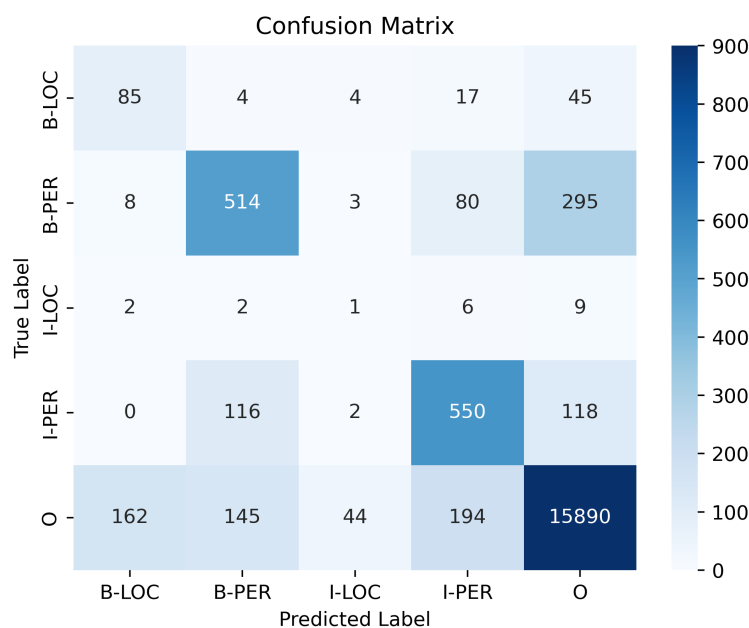


Figure 4.4: Confusion matrix for stanza on SA partition

Label	Precision	Recall	F1-score	Support
I-LOC	0.432	0.416	0.424	209
B-LOC	0.734	0.844	0.785	809
I-PER	0.782	0.803	0.792	1301
B-PER	0.679	0.883	0.768	1161
Micro Avg.	0.714	0.816	0.762	3480
Macro Avg.	0.6572	0.7369	0.692	3480
Weighted Avg.	0.715	0.816	0.760	3480

Table 4.3: Classification report for Stanza on Biography test partition

The Stanza model achieves much better performance on the biography test partition in comparison to the Tag de Tekst partitions. For I-LOC, Stanza achieves a 0.432 precision and 0.416 recall. For B-LOC stanza achieves 0.734 precision and 0.844 recall. For I-PER we get a 0.782 precision and 0.803 recall. The B-PER label achieves 0.679 precision and 0.883 recall. This finding somewhat supports the prior suggestion, namely that the Stanza model had difficulty with the strong domain differences and noise found in the Tag-de-Tekst partitions.

The confusion matrix for this partition also reflects these results. This partition finds relatively less confusion between B-PER and I-PER labels when compared to Stanza performance on Tag-de-Tekst. We do however still see that the model is particularly conservative with the I-LOC label, though it only occurs a total of 209 times in this Biography test partition.

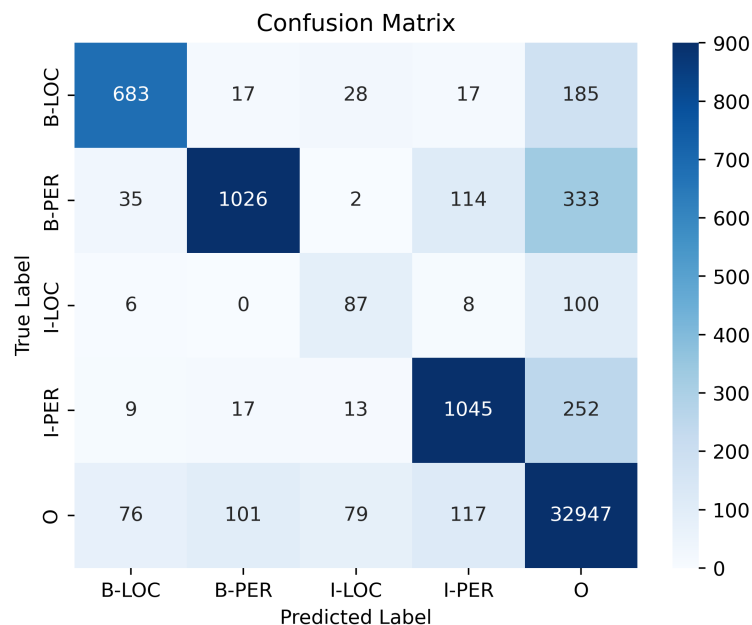


Figure 4.5: Confusion matrix for stanza on biography partition

4.0.3 Flair Model

Next, we look at the performance of our Flair model.

Label	Precision	Recall	F1-score	Support
B-LOC	0.7258	0.5357	0.6164	252
B-PER	0.6060	0.6903	0.6454	352
I-PER	0.7495	0.7433	0.7464	483
I-LOC	0.3137	0.1039	0.1561	154
Micro avg.	0.6741	0.6068	0.6387	1241
Macro avg.	0.5987	0.5183	0.5411	1241
Weighted avg.	0.6499	0.6068	0.6181	1241

Table 4.4: Classification report for Flair on the NHA partition

The Flair model gets better scores than Stanza on the test NHA partition. It achieves higher precision than recall for the B-LOC label. With 0.752 precision and 0.535 recall. For I-LOC it achieves 0.313 precision and .103 recall, thus implying that location span is more difficult for the model to interpret. For the PER label, the model reaches a similar F1 score for onset and span, with the I label achieving a slightly higher score of 0.74 compared to 0.64 for the B-PER label.

Similarly to the Stanza model, we can see that LOC seems to be the most difficult label to interpret. In the confusion matrix shown on [4.6](#) we can see that this model also predicts very few I-LOC locations, and conservatively classifies them as O. Similarly to stanza, we see the most PER confusion occurring with the B and I variants of that tag.

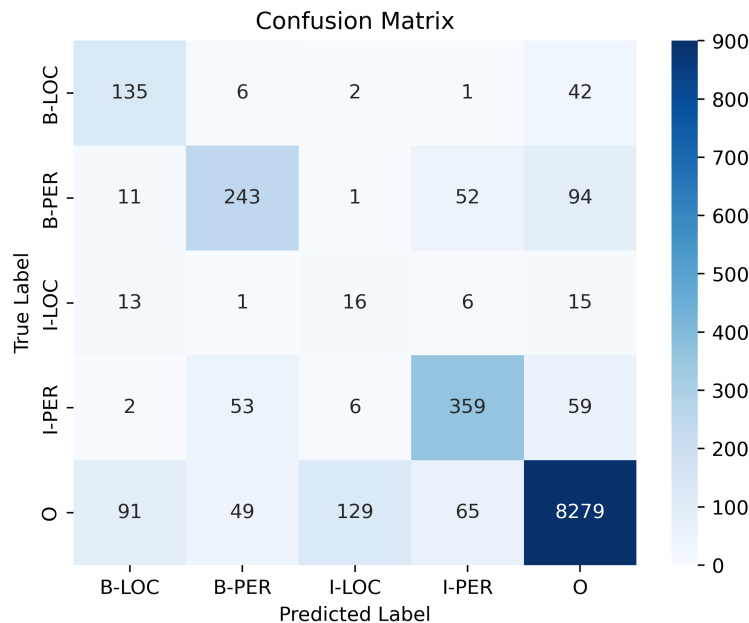


Figure 4.6: Confusion matrix for Flair on NHA partition

Label	Precision	Recall	F1-score	Support
B-LOC	0.5529	0.5486	0.5508	257
B-PER	0.6330	0.7375	0.6813	781
I-PER	0.7614	0.7308	0.7458	847
I-LOC	0.0435	0.0556	0.0488	54
Micro avg	0.6541	0.6906	0.6719	1939
Macro avg	0.4977	0.5181	0.5066	1939
Weighted avg	0.6620	0.6906	0.6745	1939

Table 4.5: Classification report for Flair on the SA partition

On the SA partition, Flair achieves a worse B-LOC F1-score at 0.55. Similarly to the other data, the I-LOC performs worse than B-LOC with a F1 score of just 0.04. I-PER does slightly better on this partition with a F1 score of 0.74 compared to 0.68 for the B-PER label. This implies that the SA partition is not much easier for this model to the extent that it was easier for Stanza.

The confusion matrix for this model [4.7](#) shows that I-LOC is typically not guessed by the model, confusing it for O 62 times. We also see that the model confused the O label for B-PER and I-PER tags. This is not entirely surprising because the distribution of PER labels is also typically higher, as we indicated in the data overview section. As we saw in the other partitions, we again see that B-PER and I-PER tags were often frequently confused by one another.

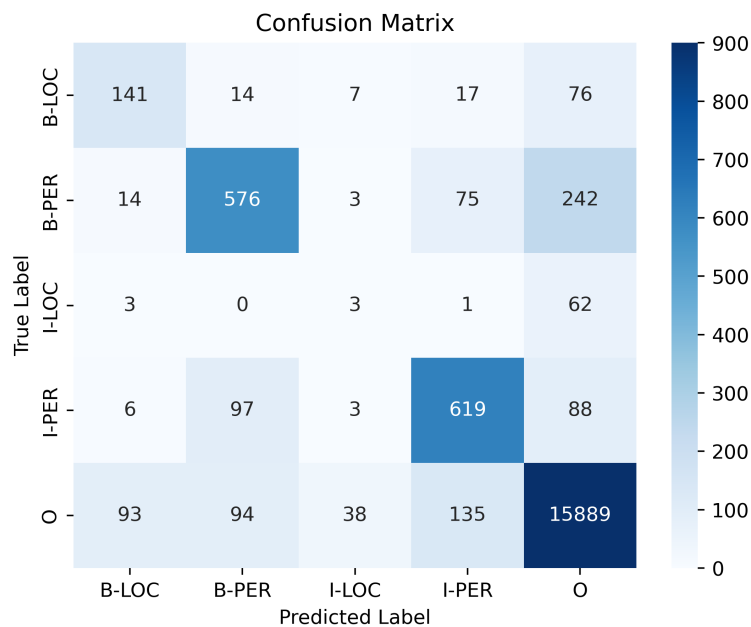


Figure 4.7: Confusion matrix for Flair on SA partition

On the Biography test set, Flair does better in comparison to the Tag de Tekst data. This is likely due to the domain difference and low amount of noise in this partition.

Class	Precision	Recall	F1-score	Support
B-LOC	0.7786	0.8999	0.8349	809
B-PER	0.7054	0.8889	0.7866	1161
I-PER	0.7921	0.7879	0.79	1301
I-LOC	0.6124	0.5215	0.5633	209
Micro Avg	0.7478	0.8316	0.7875	3480
Macro Avg	0.7221	0.7745	0.7437	3480
Weighted Avg	0.7493	0.8316	0.7857	3480

Table 4.6: Classification report for Flair on Biography test partition

All of the scores have an F1 above 78, with the exception of I-LOC, achieving an F1 score of 0.56.

In the confusion matrix [4.8](#) we can again see that the model is more likely to guess a PER label when the true label is O. Additionally, we see that there is little confusion about the LOC label for other labels than O. Again however we see that a true B-PER label is often confused as I-PER by the model.

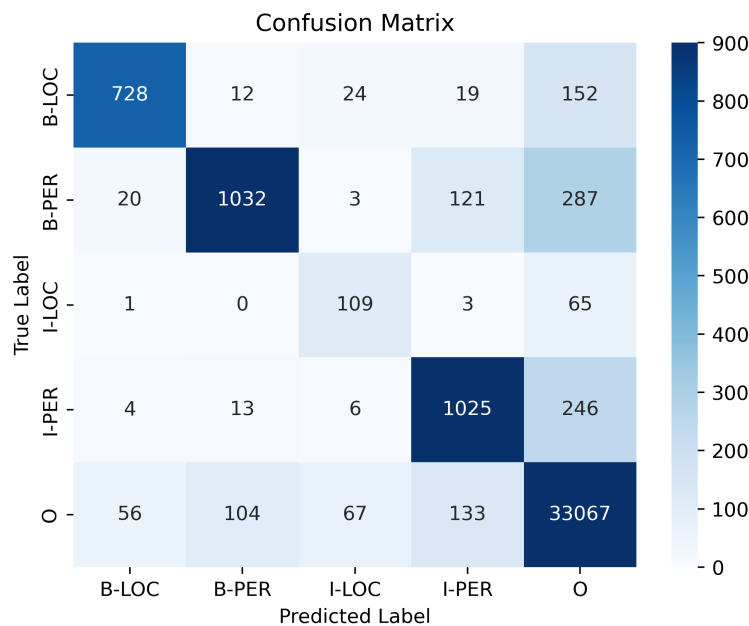


Figure 4.8: Confusion matrix for Flair on biography partition

4.0.4 Baseline Model

Our baseline model was BERTje trained on CoNLL data. From the results we quickly notice that it achieved poor results on our data.

As can be seen from [4.7](#) Baseline BERT model does not achieve great NER per-

Label	Precision	Recall	F1-score	Support
B-LOC	0.61	0.26	0.37	252
B-PER	0.61	0.31	0.41	352
I-PER	0.55	0.29	0.38	483
I-LOC	0.21	0.05	0.08	154
Micro avg	0.56	0.26	0.35	1241
Macro avg	0.49	0.23	0.31	1241
Weighted avg	0.54	0.26	0.35	1241

Table 4.7: Classification report for baseline model on the NHA test partition

formance on the test NHA partition. This model achieves a 0.41 F1-score on B-PER, a 0.38 F1-score on I-PER, a 0.08 F1-score on I-LOC and a 0.37 F1-score on B-LOC. Similarly to the other models, it seems that I-LOC is the most difficult to interpret, though for the baseline BERT model, this score is lower in comparison to Stanza and Flair. Additionally, span is harder for the model to identify in comparison to onset.

When we see what this baseline model confuses in the confusion matrix from [4.9](#) we primarily see the high amount of O guesses. This implies that this model takes an extremely conservative approach while guessing. Furthermore, it reflects similar confusion as the other models with B-PER and I-PER. The low amount of guesses by this model do not lend itself to clear interpretations to be made from this matrix.

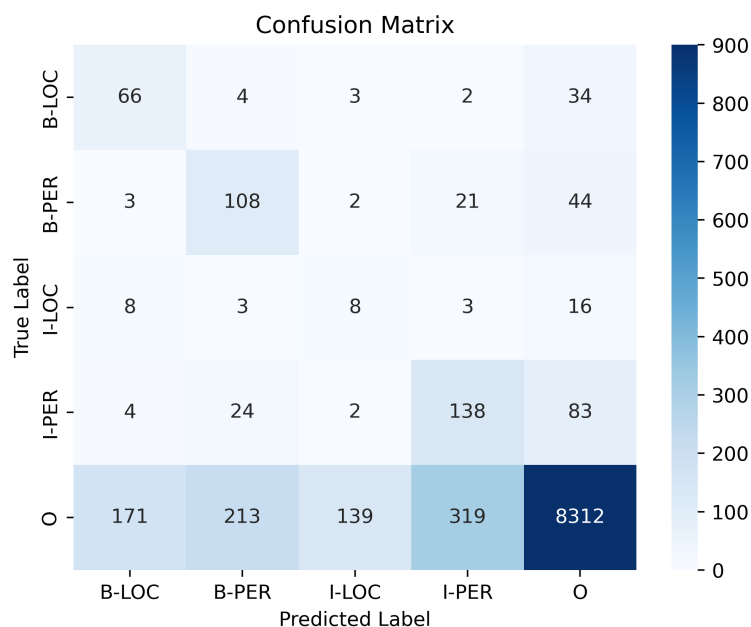


Figure 4.9: Confusion matrix for baseline model on NHA test partition

On the SA dataset, the model performs similarly, achieving an F1 score of 0.47 for B-PER, 0.33 for I-PER, 0.016 for I-LOC and 0.241 for B-LOC. Notably, B-LOC classification scores lower in comparison to this model on the NHA partition, however

Class	Precision	Recall	F1-score	Support
B-LOC	0.430	0.167	0.241	257
B-PER	0.586	0.397	0.473	781
I-PER	0.492	0.251	0.333	847
I-LOC	0.014	0.019	0.016	54
micro avg	0.499	0.292	0.369	1939
macro avg	0.380	0.209	0.266	1939
weighted avg	0.508	0.292	0.368	1939

Table 4.8: Classification report for baseline model on the SA test partition

I-LOC scores slightly better, but still bad at 0.016.

In the confusion matrix from [4.10](#) we see a similar pattern. This time the O label is centered in the figure, and its dark color immediately highlights that it is also conservative whilst guessing. We also again see that B-PER and I-PER labels are typically confused by one another, and very few I-LOC guesses were made, whilst B-LOC was frequently misclassified as O.

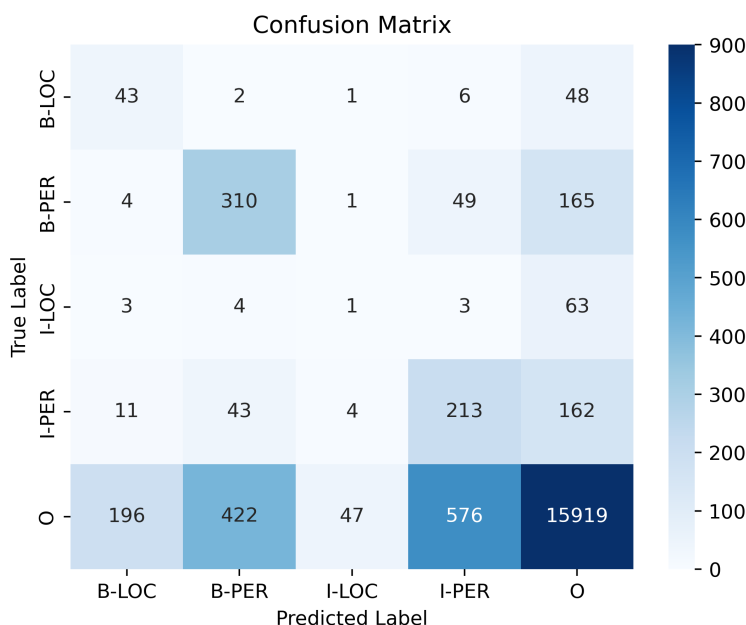


Figure 4.10: Confusion matrix for baseline model on SA test partition

The F1 scores for the baseline BERT model on the test biography partition were 0.45 for B-PER, 0.42 for I-PER, 0.31 for I-LOC and 0.60 for B-LOC. Surprisingly, B-LOC performed much better on this dataset in comparison to the baseline BERT model performance on the other test sets. Additionally, the scores for this BERT model again reflects that this partition seems to yield better results, likely due to the domain differences and less noise.

Class	Precision	Recall	F1-score	Support
B-LOC	0.738	0.509	0.603	809
B-PER	0.538	0.398	0.458	1161
I-PER	0.544	0.351	0.427	1301
I-LOC	0.391	0.258	0.311	209
Micro Avg	0.579	0.398	0.472	3480
Macro Avg	0.553	0.379	0.450	3480
Weighted Avg	0.578	0.398	0.471	3480

Table 4.9: Classification report for baseline model on biography partition

In the confusion matrix from [4.11](#) we see that this model is again very conservative with its classifications. The model again makes most of its mistakes because it confuses all of the labels for O consistently. Notably, the I-LOC label is guessed more frequently on this partition than B-LOC.

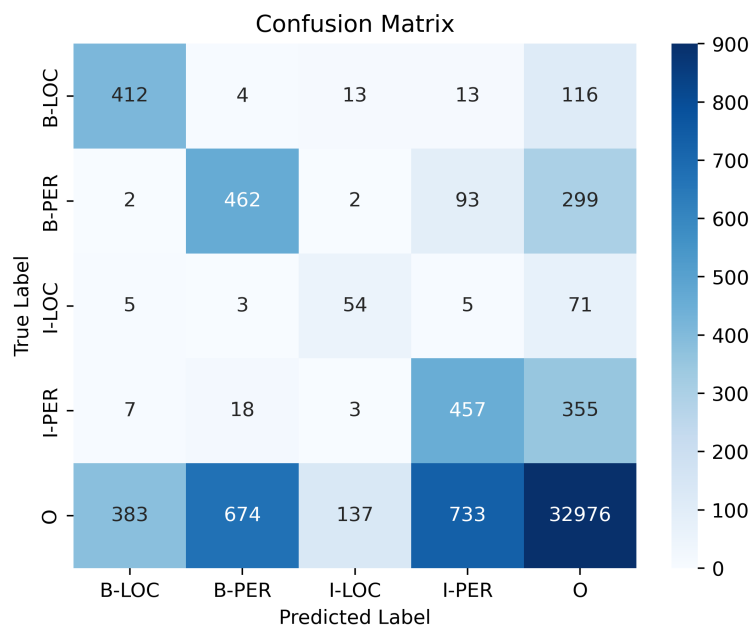


Figure 4.11: Confusion matrix for baseline model on biography test partition

Finetuned Tag-de-Tekst BERTje Model

The Tag de Tekst finetuned models achieves consistently better scores than all of the other models on the NHA and SA datasets. This is not surprising as these datasets are close to their training data. Notably this finetuned BERTje is also outperforming GijisBERT for some of the entities in these partitions.

As can be seen from [4.10](#) finetuned Tag-de-Tekst BERTje model also achieves substantially better scores than the baseline. On The NHA partition, it achieved a 0.83

	Precision	Recall	F1-Score	Support
B-LOC	0.933	0.722	0.814	252
B-PER	0.906	0.767	0.831	352
I-LOC	0.669	0.565	0.613	154
I-PER	0.831	0.917	0.872	483
micro avg	0.849	0.791	0.819	1241
macro avg	0.835	0.743	0.782	1241
weighted avg	0.853	0.791	0.816	1241

Table 4.10: BERTje on Tag de Tekst model performance for Partition NHA

F1 score for B-PER, 0.61 for I-LOC, 0.87 for I-PER and 0.81 for B-LOC. This is the best of all tested models for this partition.

The confusion matrix for this partition [4.12](#) reflects that this model outperforms the other models. The most notable confusion here is between B-LOC and I-LOC along with B-PER and I-PER. This indicates that the model might have difficulty determining the onset of tags.

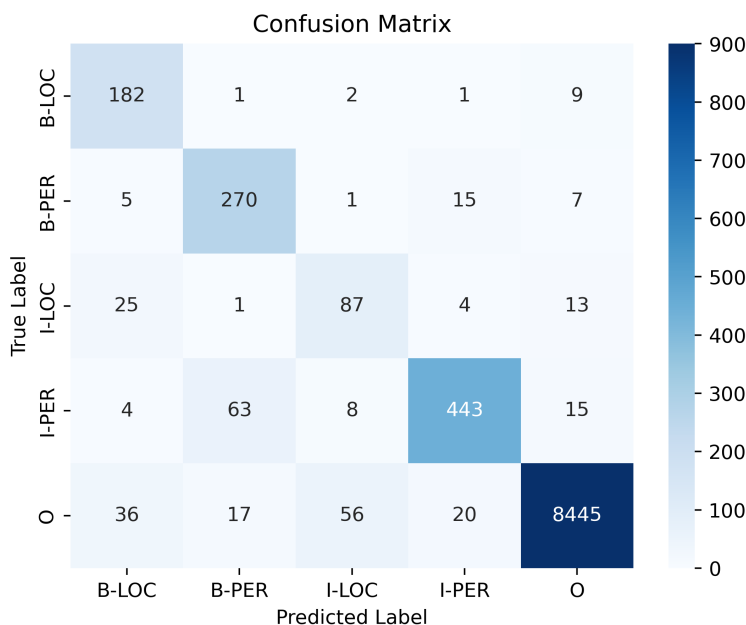


Figure 4.12: Confusion matrix for finetuned BERTje on NHA test partition

The finetuned Tag-de-Tekst BERTje model achieves similar results on the SA partition with the exception of I-LOC, which scored lower than on NHA with an F1 of 0.33. For B-PER, I-PER and B-LOC the finetuned model achieved F1 scores of 0.82, 0.85 and 0.72 respectively.

When assessing these results with a confusion matrix [4.13](#), we see that the model is actually more likely to guess B-LOC when the true label is I-LOC. Additionally we

	Precision	Recall	F1-Score	Support
B-LOC	0.833	0.642	0.725	257
B-PER	0.909	0.754	0.824	781
I-LOC	0.274	0.426	0.333	54
I-PER	0.819	0.895	0.856	847
micro avg	0.827	0.792	0.809	1939
macro avg	0.709	0.679	0.685	1939
weighted avg	0.842	0.792	0.811	1939

Table 4.11: BERTje on Tag de Tekst model performance for Partition SA

see that I-PER is often confused for B-PER which we also saw from the other models.

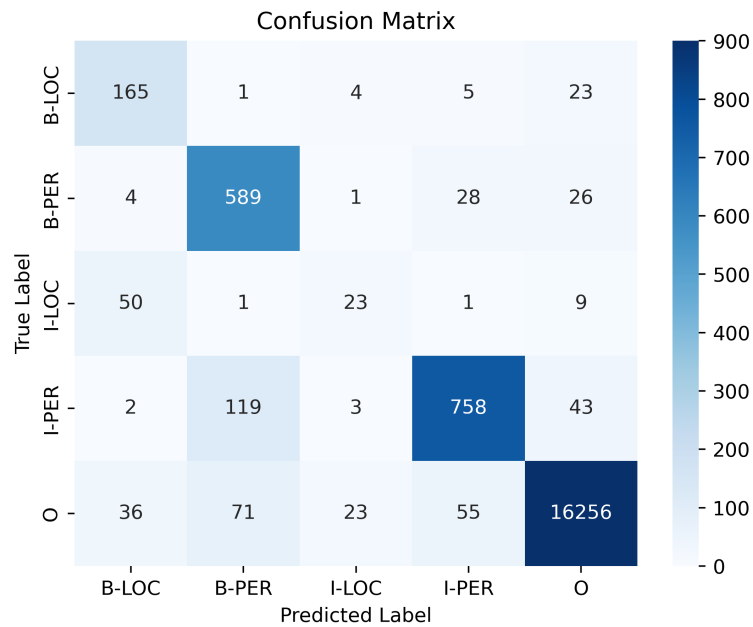


Figure 4.13: Confusion matrix for finetuned BERTje on SA test partition

Label	Precision	Recall	F1-Score	Support
B-LOC	0.759	0.787	0.773	809
B-PER	0.680	0.584	0.628	1161
I-LOC	0.352	0.522	0.420	209
I-PER	0.686	0.802	0.740	1301
micro avg	0.673	0.709	0.691	3480
macro avg	0.619	0.674	0.640	3480
weighted avg	0.681	0.709	0.691	3480

Table 4.12: BERTje on Tag de Tekst model performance for Partition Biography

On the Biography test partition, the finetuned Tag-de-Tekst BERTje model performed worse than Flair but better than the baseline. It achieved a F1 score of 0.62

for B-PER, 0.42 for I-LOC, 0.74 for I-PER and 0.77 for B-LOC. Noticeably, the I-LOC score was better at 0.42.

In the confusion matrix from [4.14](#) we see that this partitions lends itself to more confusion. We see that especially the O label is confused for other labels, suggesting that this model is not very conservative in its classification approach. Similarly to the other partitions, we see that there is confusion regarding the onset of entities such as B-PER and I-PER or B-LOC and I-LOC.

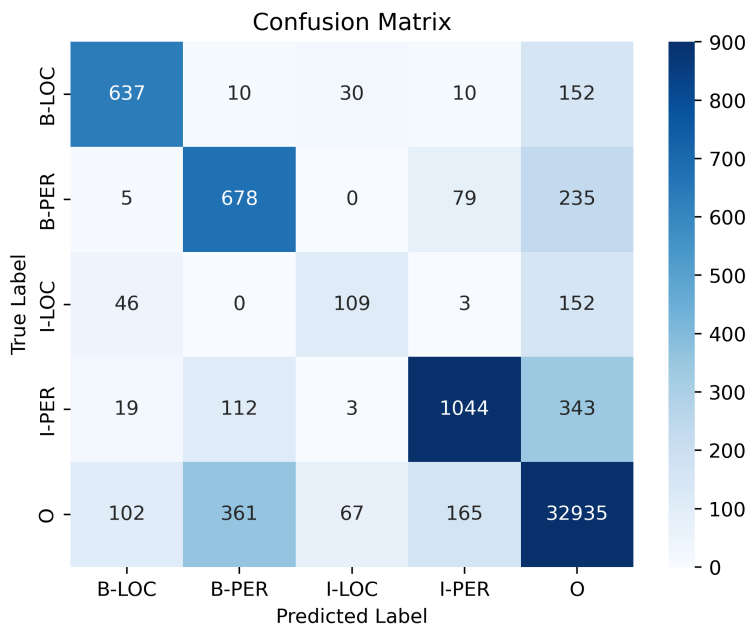


Figure 4.14: Confusion matrix for finetuned BERTje on biography test partition

GijsBERT model

GijsBERT scores similarly to the finetuned BERTje model. Surprisingly, it outperformed the finetuned tag-de-tekst BERTje model on the biography test partition, which mostly consisted of non-historical text.

Label	Precision	Recall	F1-score	Support
B-LOC:	0.932	0.810	0.867	252
B-PER:	0.869	0.639	0.736	352
I-LOC:	0.806	0.675	0.735	154
I-PER:	0.857	0.756	0.803	483
micro avg:	0.869	0.724	0.790	1241
macro avg:	0.866	0.720	0.785	1241
weighted avg:	0.869	0.724	0.789	1241

Table 4.13: Performance for GijsBERT on NHA test partition

GijsBERT performed very well on the NHA partition. It achieved an F1 score of 0.86 for B-LOC, 0.74 for B-PER, 0.74 for I-LOC and 0.80 for I-PER. From this classification report, it seems that the scores achieved here are similar to that of the Tag-de-Tekst finetuned BERTje model. The GijsBERT model outperformed the Tag-de-Tekst finetuned BERTje model slightly on the B-LOC and I-LOC tags on this partition.

When we assess these results with a confusion matrix [4.15](#), we see that there was also little confusion in this model. One thing to note however from this confusion matrix, is that the model seems to make the most guesses for O, which is also not surprising as O is the most common label.

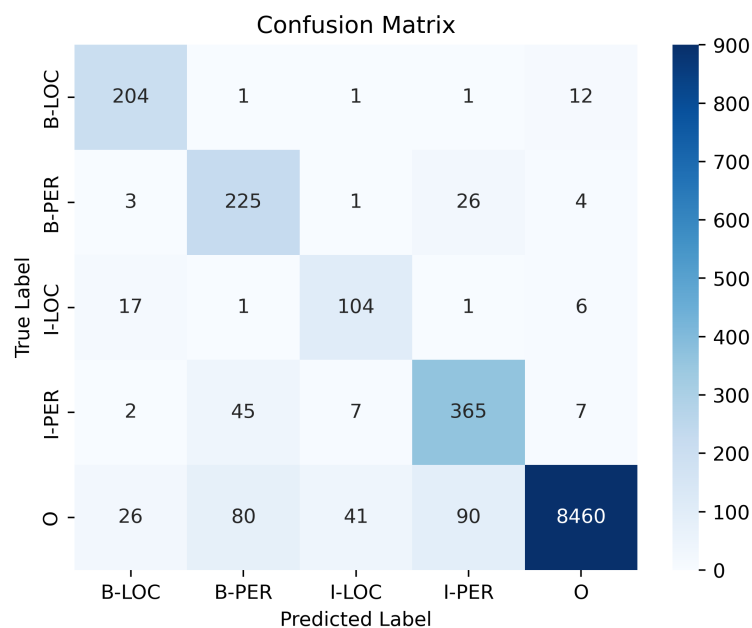


Figure 4.15: Confusion matrix for finetuned GijsBERT on NHA test partition

Label	Precision	Recall	F1-score	Support
B-LOC:	0.826	0.739	0.780	257
B-PER:	0.901	0.713	0.796	781
I-LOC:	0.412	0.389	0.400	54
I-PER:	0.838	0.842	0.840	847
micro avg:	0.846	0.764	0.803	1939
macro avg:	0.744	0.671	0.704	1939
weighted avg:	0.850	0.764	0.802	1939

Table 4.14: Performance for GijsBERT on SA test partition

GijsBERT also achieved good performance on the SA partition. It achieved F1 scores of 0.78 for B-LOC, 0.79 for B-PER, 0.40 for I-LOC and 0.84 for I-PER. The averages imply that it achieved almost equal performance when compared to the finetuned BERTje model on this dataset. One notable difference is that it achieved a better

score for I-LOC.

In the confusion matrix from 4.16, we see that the I-LOC label is rarely guessed, whilst the I-PER label is guessed more frequently than the B-PER label. This again is strange, and implies that GijsBERT also has difficulties with determining the onset of an entity, similarly to what was seen from our finetuned BERTje model. In this partition we also see that GijsBERT is slightly more conservative in its classification approach, due to the larger amount of misclassifications with O in comparison to its other classifications.

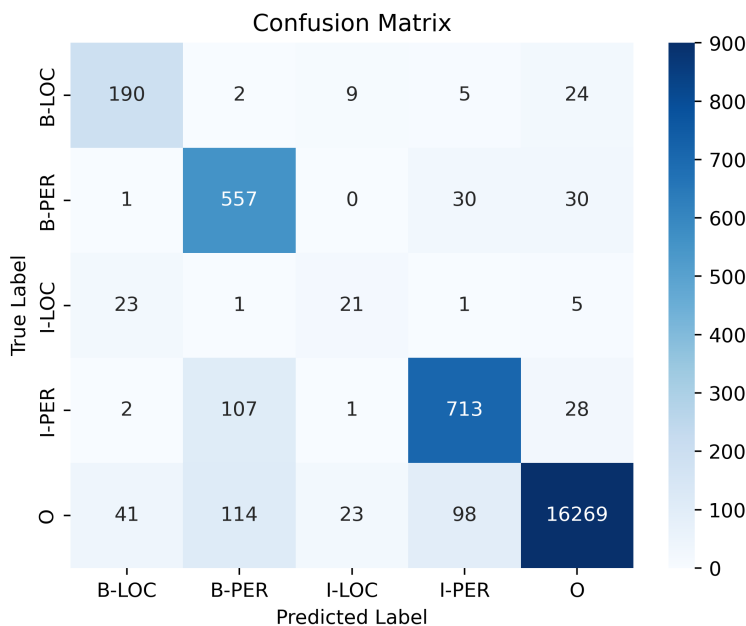


Figure 4.16: Confusion matrix for finetuned GijsBERT on SA test partition

Label	Precision	Recall	F1-score	Support
B-LOC:	0.793	0.721	0.755	809
B-PER:	0.702	0.651	0.676	1161
I-LOC:	0.510	0.254	0.339	209
I-PER:	0.749	0.739	0.744	1301
micro avg:	0.735	0.676	0.705	3480
macro avg:	0.688	0.591	0.628	3480
weighted avg:	0.729	0.676	0.699	3480

Table 4.15: Performance for GijsBERT on Biography test partition

On the biographynet partition, GijsBERT performed very similarly when compared to the Tag-de-Tekst finetuned BERTje model. It achieved an F1 score of 0.75 for B-LOC, 0.67 for B-PER, 0.34 for I-LOC and 0.74 for I-PER. GijsBERT again outperformed the finetuned BERTje on both I-LOC and B-LOC, however the difference is very small for this partition.

The confusion matrix for GijsBERT on this partition [4.17](#) shows that I-LOC and B-LOC along with B-PER and I-PER are often confused by the model. Additionally, we see the model is somewhat conservative in its classification due to the variation of predictions when the true label was O.

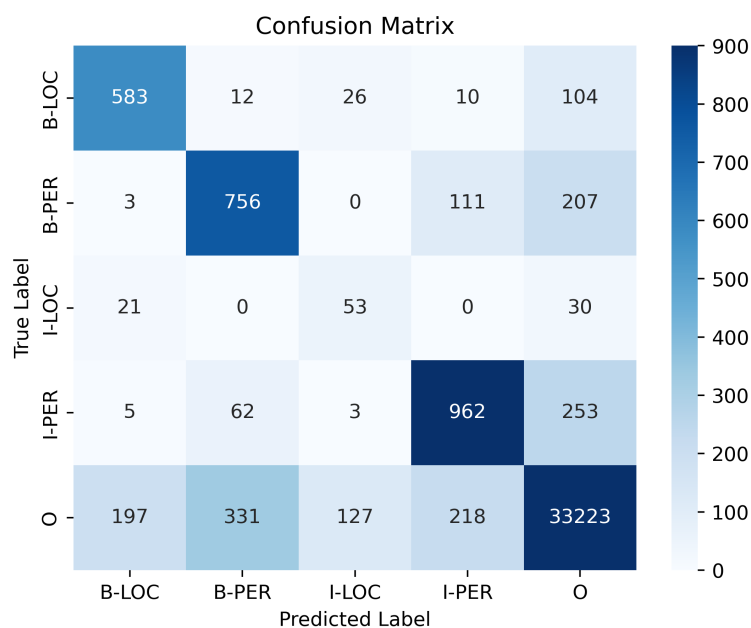


Figure 4.17: Confusion matrix for finetuned GijsBERT on biography test partition

4.0.5 Qualitative Evaluation

Our qualitative evaluation consisted of two random samples. One of these random samples was done specifically for sources that ranged from 1729 up until 1878. The other sample was for finding more recently written biographies that occurred from 1900 to present. Three biographies were selected for each sample which resulted in two small partitions for our evaluation. The older partition consisted of approximately 1000 tokens, and 77 entities, whereas the newer partition resulted in 1300 tokens with 140 entities.

In this section we will review the classifications made by our three models on an individual level. We will specifically look for misclassifications with a pattern occurring more than once, exploring the limitations of each model. In the data partitions, it must be noted that the location entity is more rare, which is the case for all our data. In the newer set are only 10 location entities, and only one of which is I-PER. This is likely the case because the quality of the data is higher in comparison to what we saw in Tag de Tekst data. In the case of Tag de Tekst, the OCR would in some cases split a location entity such as “Amsterdam” into two tokens, “Ams” and “terdam”, resulting in more I-LOC entities. Additionally, the annotation guidelines limit location entities

to official locations, which are typically only a single token marking a city or a country.

In table 4.0.5 we present the general statistics for our two sets with a table. These indicate the amount of entities found in each. We choose to present these here instead of in the data overview because our sampling method was part of the experimental setup.

Dataset	B-PER	I-PER	B-LOC	I-LOC	O
Old biographies selection	33	37	6	1	926
New biographies selection	68	41	30	1	1162

Table 4.16: Label distribution overview for qualitative partitions

Qualitative Evaluation for baseline BERTje

As was implied from the classification reports, the baseline BERTje model made many mistakes. It seemed to loosely mark instances of the token "van" as I-PER. With this is meant that it marked no instances of a B-PER anywhere around this token. "van" is a function word in Dutch, but is also commonly used in surnames. In the modern Dutch text, another function word, "is" was marked as B-PER, despite this not being a common marking for a name.

This BERTje model did correctly mark person-titles such as "Mevrouw" but missed the more unique tokens that followed thereafter. In the modern partition the name "Jeanne d' Arc" had "Jeanne" and "d" correctly identified as B-PER and I-PER but had "Arc" missed. This seemed to be a consistent finding, the baseline BERTje model did not seem to mark any person three token PER entities correctly.

As for LOC some well known cities such as Rotterdam and Leiden were missed in the middle Dutch partition, whereas other cities like Utrecht, Groningen and Amsterdam were identified correctly. "Oranjewoud" was another missed LOC entity, though this is less surprising as it is a lesser known region within the Netherlands.

Qualitative Evaluation for Tag-de-Tekst BERTje

The first model that we will evaluate is the Tag-de-Tekst BERTje model. One important thing to note is that the model chooses to not annotate person-titles. The annotators for both of the biography partitions however do annotate these person-titles. This likely indicates the reason as to why the PER scores were lower for the biographynet test partition.

For the middle Dutch data, the Tag-de-Tekst BERTje model primarily made onset mistakes with B-PER and I-PER tags, which is due to the aforementioned annotation convention differences from Tag de Tekst and biographynet. These cover a lot of the PER disagreements in the middle Dutch text. Another notable finding is that a German excerpt in the middle Dutch data with some PER entities, was correctly marked by the model.

In the new biographies selection, the model starts by overestimating the span of a PER entity. The phrases “Schilder, tekenaar. Geboren 1638, Utrecht.” marks the introduction of a person where “Schilder” is their first name and “tekenaar” is their profession. The model correctly marked Schilder as the onset of a name, but “tekenaar” was incorrectly marked as I-PER. Typically, the information that would help understand that “tekenaar” is a profession is the comma inbetween these two tokens and the lowercase ‘t’.

The model also made mistakes as some PER entities were missed. “BOELAART JAN” was a person entity that was missed by the model. Notably, a distinguishing characteristic of this entity was that it was completely written with capital letters. Another distinguishing characteristic of this entity was that it had an atypical order where the surname comes before the given name. This is likely the more important characteristic, because another PER entity (“THEODORUS STOCKUM”) in the middle Dutch text with all capital letters was correctly identified by the model. There was also an instance where a person entity only consisted of the initial “B.” which was also missed by the model.

The behaviour for identifying location entities was interesting. The Tag de Tekst finetuned BERTje model did not seem to always assign the B-LOC tag at the onset of a location. In some cases, a single LOC entity would be labeled as I-LOC. This is interesting because for other single token location entities the model identified B-LOC correctly. This particular example happened on the token “Oranjewoud” in the sentence:

- “In het bejaarde centrum Oranjewoud.”

Similar mistakes occurred in the modern biography partition. Another example where I-LOC was used instead of B-LOC occurred at the entity “Indie”. For both of these cases it occurs when there is one token inbetween the PER and LOC entity, though further evaluation would be needed to conclude that this happens in all such cases.

For LOC tags, some entities were also missed by the model. This included “italie”, and “Gelre” in the modern partition.

Qualitative Evaluation for GijsBERT

The next model we evaluate is GijsBERT. The first thing to note is that the GijsBERT model seems to perform similarly to the BERT model for the old biographies selection. Similarly, it does not mark person-titles. This is to be expected because both BERTje and GijsBERT were finetuned on the same data.

For the middle Dutch data, the Gijsbert model missed persons which the Tag-de-Tekst BERTje model marked correctly. Person entities like “J.M.M Aler” , “Jeanne d’ Arc” and “Vigny Ibsen” were missed by the model in the modern Dutch partition. Some common Dutch names such as “Hendrik” and “Willem” were missed by the model in the modern partition. This might be because these names are more modern and thus have not been seen before by the model. Notably, the GijsBERT model confused the

B-PER tag for I-PER in some instances where only a single token was present. In the modern partition, it also made token specific mistakes, where the token “Gelre” was consistently confused for a PER entity whilst being a LOC entity. Similarly to the BERTje model, the GijsBERT model correctly identified PER entities in the German excerpts.

Notably, the GijsBERT model made very few mistakes with LOC entities. The only LOC mistake that was repeatedly made occurred with the token “Gelre” which was, as mentioned before frequently confused for the PER tag.

Chapter 5

Discussion

The purpose of this section is for discussing the results and the implications thereof. Additionally, we discuss what could be improved, along with discussing what we suggest for future directions in the field.

5.0.1 Interpretation of findings

The present work highlights the differences between several NER models on current and historical Dutch data. The evaluation included quantitative classification reports for out-of-the-box NER tools such as Stanza and Flair, as well as quantitative and qualitative assessments for three BERT-based models.

Based on the classification reports, it was observed that Stanza and the Baseline BERTje model performed the worst across all three data partitions. This finding was unexpected, particularly in the case of BERTje, considering that other fine-tuned variants of BERTje achieved significantly better scores according to the quantitative reports. During the experiment, we took extra care to ensure that the model was initialized correctly when we noticed this discrepancy. However, there was no difference in performance whether we initialized the model directly via the tools provided in the HuggingFace platform or by unpacking the subwords into tensors and making predictions from there. Notably, the results obtained from this model differ greatly from what was suggested in (?). It is possible that the referenced model was not identical to the one available in the HuggingFace repository, but there is no available documentation to confirm this. Another explanation could be that the baseline BERTje model, fine-tuned on CoNLL, does not generalize well to our data.

The remaining GijsBERT and BERTje models were assessed both quantitatively and qualitatively. When selecting the appropriate variants of these models, we considered the validation and training losses. The validation process revealed that all variations of BERTje and the GijsBERT model experienced an increase in validation loss after the second epoch. There could be several reasons for this observation. One possibility is that unique conditions present in the Tag de Tekst dataset, such as sections with lower OCR quality or worse annotation quality, contribute to this behavior. Since no explicit annotation guidelines were presented in the Tag de Tekst data source, in addition to only having a single set of gold labels for each partition, we can not measure interrater agreement.

Another problem might be that the noise present in OCR-based training data might limit the model’s ability to extract signal from the annotations, thereby hindering convergence. Alternatively, experimenting with different hyperparameter values, such as learning rate and max sequence length, could help address this issue. On a positive note, the fact that the GijsBERT and BERT models were exposed to the same extent of training facilitates their comparison.

When analyzing the classification reports, it was found that the performance of GijsBERT and BERTje was very similar. Surprisingly, the fine-tuned BERTje model did not necessarily perform worse on historical texts compared to GijsBERT. In fact, when assessing the weighted averages, BERTje even showed better performance on the NHA partition. Another unexpected finding was that the GijsBERT model outperformed BERTje on the Biography test partition, despite the fact that this partition largely contains modern data. It is important to note that the Biography test partition posed a greater challenge for both models, which is expected due to its different characteristics compared to the data used for model fine-tuning. Overall, it can be concluded that both models performed similarly, which aligns with the negligible differences reported in the original GijsBERT study (Manjavacas and Fonteyn, 2022) [p.130].

The qualitative evaluation provided further insights into the performance of BERTje and GijsBERT. It revealed that the models still struggled with some basic aspects of the task, such as the requirement for an I label to occur only if there is a preceding B label. By employing a regular expression to identify mistakes in the training data, it was discovered that the Tag de Tekst annotators also made several such errors. Additionally, a mistake was identified in the fine-tuned BERTje model, where it misclassified a PER entity when a given name followed a surname. Unfortunately, due to the limited size of our test partitions, we were unable to thoroughly investigate the causes of these mistakes.

5.0.2 Limitations and Future Research

There are several limitations of our current study. The most important of which is that our baseline system seems to perform poorly on all test partitions. Because this has been retrieved from an external party, we are also unable to assess the quality of the set. The name of this model however contains CoNLL-2002, suggesting that it was trained on approximately 16,000 sentences.

Another limitation lies in our training partition. The quality of the Tag de Tekst data could not have been verified by metrics such as annotator agreement. Additionally, performance could have likely improved if more effort was taken to further preprocess this data, as remnants of poor OCR quality were still found.

For future iterations we would also have liked to further expand the test partition for qualitative evaluation. This would have helped more consistency in the examples. Alternatively, future exploration in this area might find it sufficient to explore the development of challenge sets in order to better understand the behaviour of these models.

The findings of this thesis point additional investigation towards finetuning for domain adaptation. It seems that with BERTje performing similarly to GijsBERT on the historical domain, the finetuned more general BERTje model, is the more cost-effective

solution to this problem.

Chapter 6

Conclusion

This thesis explored the task of Named Entity Recognition (NER) by exploring the performance of different systems on current, and older dutch biographical data. We were interested in two research questions.

1. How does a Dutch BERTje model with an additional finetuned classifier from a generic dataset perform on the NER task in comparison to out of the box tools such as Stanza and Flair?
2. How well does BERTje finetuned on historical data perform against a BERTje that was pretrained on the historical domain?

In order to explore these questions, we prepared some out of the box tools such as Stanza and Flair, along with a BERTje model trained on CoNLL that was readily available through the HuggingFace platform. Additionally, all data partitions were analysed, and several extra partitions annotated on biographical data were used to explore the in domain performance quantitatively.

With regards to the first research question, our results imply that Flair had the best results on all of our quantitative evaluation data when compared to the generic BERTje model and Stanza. We interpreted the performance of both Stanza and the baseline BERTje model as poor. This result was unexpected as we initially hypothesized that the BERTje model would be better suited for this task due to its Word-Piece tokenization system. Extra care was taken to ensure that the BERTje model was initialized correctly, and our quantitative measurements show much worse performance from this model in comparison to the original paper from (?). One possible explanation highlighted in our discussion lies in the limited understanding that we have about how this particular BERTje model was finetuned.

For the second research question, we finetuned two models on historical Dutch text sourced from Tag de Tekst. These models were pretrained BERT based systems, called BERTje (de Vries et al., 2019), and GijsBERT (Manjavacas and Fonteyn, 2022). Notably, we found that both models had difficulty minimizing the loss function across epochs, and in both cases, the second epoch had the highest loss of the total eight. One benefit to this however, is that both models had been exposed to the same number of iterations across the training data, resulting in a more fair comparison.

For these two models the results imply that BERTje and GijsBERT perform similarly across all our test partitions and much better than the baseline model. This is also a surprising finding, as we initially expected that a model pre-trained on historical data will outperform a more general model like our finetuned BERTje. These negligible differences however, were also identified in the original GijsBERT paper (Manjavacas and Fonteyn, 2022)[p.130]. When evaluating their misclassifications, we found similar mistakes across these two systems. Our results seem to therefore imply that costly in-domain pretraining on BERT models does not necessarily yield higher performance in comparison to finetuning a more generic model on in-domain data. Another finding regarding the qualitative evaluation lies in the inability for these models to understand that B tags are associated with the onset of a tag.

To conclude, it appears that Flair is the best out-of-the-box tool available for the NER task when assessed on Dutch historical and biographical data. In addition to that, we found that a more generalized BERTje model finetuned on OCR level historical data, will perform similarly to a GysBERT model that was both pre-trained and finetuned on historical data. This result became apparent when assessed on both Dutch biographical, and historical data.

Bibliography

- A. Akbik, T. Bergmann, and R. Vollgraf. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, 2019.
- A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- I. Beltagy, K. Lo, and A. Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.
- H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.
- W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Dutch Language Institute. Ai-trainingset for ner (version 1.0). Data set, 2022. URL <http://hdl.handle.net/10032/tm-a2-v2>
- J. Ebrahimi, H. Yang, and W. Zhang. How does adversarial fine-tuning benefit bert? *arXiv preprint arXiv:2108.13602*, 2021.
- J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, pages 363–370, 2005.
- A. Fokkens, S. Ter Braake, N. Ockeloën, P. Vossen, S. Legêne, G. Schreiber, and V. de Boer. Biographynet: Extracting relations between people and events. *arXiv preprint arXiv:1801.07073*, 2018.

- Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015.
- Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.
- I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength natural language processing in Python. 2020. doi: 10.5281/zenodo.1212303.
- M. Hüning and U. Vogl. Middle dutch—a short introduction. *Of Reynaert the Fox*, pages 257–271.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- R. A. Kollewijn. *Onze lastige spelling: een voorstel tot vereenvoudiging*. Gerlings, 1891.
- K. Labusche, P. Kulturbesitz, C. Neudecker, and D. Zellhöfer. Bert for named entity recognition in contemporary and historical german. In *Proceedings of the 15th conference on natural language processing*, pages 9–11, 2019.
- X. Liu, P. He, W. Chen, and J. Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.
- E. Manjavacas and L. Fonteyn. Non-parametric word sense disambiguation for historical languages. In *Proceedings of the 2nd International Workshop on Natural Language Processing for Digital Humanities*, pages 123–134, 2022.
- P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*, 2020.
- D. H. Renders. Biografisch portaal - persoon id, 2023. URL <https://www.biografischportaal.nl/persoon/ID>. [Online; accessed 12-Feb-2023].
- H. Renders, H. Nielen, and L. Heerma van Voss. Biography portal of the netherlands, n.d. URL <http://www.biografischportaal.nl/en/about>. Accessed: April 26, 2023.
- S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- R. Schoonen, A. van Gelderen, K. de Glopper, J. Hulstijn, A. Simis, P. Snellings, and M. Stevenson. First language and second language writing: the role of linguistic knowledge, speed of processing and metacognitive knowledge.
- M. Siegenbeek. *Verhandeling over de Nederduitsche spelling, ter bevordering van eenparigheid in dezelve*. bij Johannes Allart, 1804.

- E. F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. URL <https://aclanthology.org/W02-2024>.
- J.-M. Trouille. 6. changes in the spelling of dutch. *Simplified Spelling Society, Journal* 5. 1987/2, 5:13, 1987.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- R. Vosters. Taalgebruik, taalvariatie en taalpolitiek in vlaamse assisendossiers ten tijde van het verenigd koninkrijk der nederlanden (1815-1830). In *Histoiredu Droit et de la Justice. Actes du Colloque Belgo-Néerlandais d’Histoire du Droit*, 2009.
- S. Wu and M. Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1077. URL <https://aclanthology.org/D19-1077>.

Chapter 7

Appendix

7.1 Word Clouds

During the data analysis phase, several Word Clouds were produced, some of which have been omitted from the data overview. The wordcloud in [7.1](#) shows the most popular named entities in the biography test partition when identified through Stanza. We applied this same procedure to the other partitions during the data exploration phase. Figure [7.2](#) shows the popular entities according to Stanza for the NHA partition. Figure [7.3](#) shows the popular entities according to Stanza in the discarded RHC partition. Figure [7.4](#) shows the popular entities according to Stanza in the SA partition.

7.2 BiographyNet plots

When exploring the available data for this project, we also made some circle plots containing the list of sources in the full biography net set and approximately how many articles are present in each. This can be seen in [7.5](#).



Figure 7.1: Word cloud showing the most popular entities in the biography test partition



Figure 7.2: Word cloud showing the most popular entities in the NHA partition

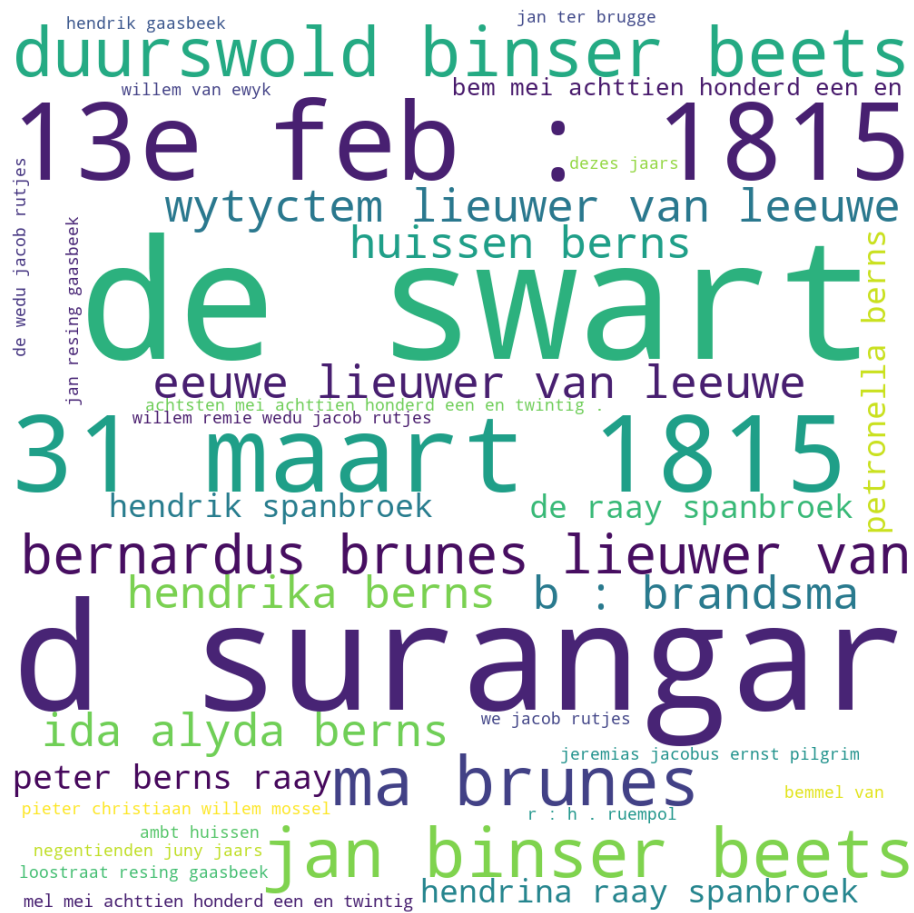


Figure 7.3: Word cloud showing the most popular entities in the discarded RHC partition

