Named Entity Recognition in Speech-to-Text Transcripts

Peter Røysland Aarnes

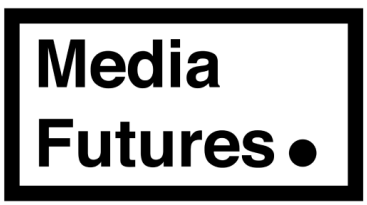
Supervisors: Samia Touileb (UiB), Lubos Steskal (TV2)

Department of Information Science and Media Studies
University of Bergen

May 31, 2023

Scientific environment

This study is carried out at the Department of Information Science and Media Studies, University of Bergen. The work is supported by MediaFutures: Research Centre for Responsible Media Technology & Innovation, and by TV2



Acknowledgements

I would like to give a special thanks to my two supervisors, Samia Touileb and Lubos Steskal. Thank you so much for your invaluable and thorough feedback on my project. Your discussions and deliberations in the research field of natural language processing have been especially inspiring and have played a large part in unlocking my special interest in this field of research.

I also want to bring attention to the research community that publishing their findings openly and making them easily accessible to the public. I believe your research contribute to the collective knowledge and progress in various fields, and allows for wider dissemination and promotes collaboration. Thank you.

Also a big thanks to family, friends and those whom I hold most dearly. You have provided me with endless support, and I am truly fortunate to have you in my life and am deeply grateful for your presence.

Peter Røysland Aarnes
Bergen, 24.05.23

Abstract

Traditionally, named entity recognition (NER) research use properly capitalized data for training and testing give little insight to how these models may perform in scenarios where proper capitalization is not in place. In this thesis, I explore the capabilities of five fine-tuning BERT based models for NER in all lowercase text. Furthermore, I aim to measure the performance for classifying named entity types correctly, as well as just simply detecting that a named entity is present, so that capitalization errors may be corrected. The performance is assessed using all lowercase data from the NorNE dataset, and the Norwegian Parliamentary Speech Corpus. Findings suggest that the fine-tuned BERT models are highly capable of detecting non-capitalized named entities, but do not perform as well as traditional NER models that are trained and tested on properly capitalized text.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 2
- 1.2 Research question . . . 2
- 1.3 Thesis Outline . . . 2

2 Background **5**
- 2.1 Natural language processing and sequence labeling . . . 5
- 2.2 Evaluation methods and analysis for NER . . . 6
  - 2.2.1 Confusion matrix . . . 7
  - 2.2.2 Accuracy, Precision, Recall and F1 . . . 7
- 2.3 Label encoding schemes . . . 9
- 2.4 English data . . . 9
- 2.5 Norwegian dataset – NorNE . . . 11
- 2.6 Traditional approaches to NER . . . 11
- 2.7 Neural approaches . . . 12
- 2.8 Challenges in NER . . . 16
  - 2.8.1 Out-of-vocabulary . . . 16
  - 2.8.2 Named entity span . . . 16
  - 2.8.3 Capitalization . . . 17
- 2.9 Loss, Optimizers and Regularization for Training . . . 17
  - 2.9.1 Feature Representation and Word Embeddings . . . 18
- 2.10 Large Pre-Trained Language Models . . . 20
  - 2.10.1 BERT Pre-Training and Fine-Tuning . . . 20
  - 2.10.2 Language specific BERT models . . . 21
  - 2.10.3 Large Pre-trained Language Model Biases . . . 22

3 Datasets **23**
- 3.1 NorNE . . . 23
- 3.2 Norwegian Parliamentary Speech Corpus . . . 24
- 3.3 Annotated NPSC sample . . . 26

4 Experimental Setup and Model Training **29**
- 4.1 Hardware and software environment . . . 29
- 4.2 Model Setup . . . 29

4.3 Pre-processing of datasets 30
4.4 Baseline models 31
4.4.1 SpaCy Norwegian Pipeline 31
4.4.2 NCRF++ 31
4.5 Main models 32
4.5.1 NorBERT 32
4.5.2 NorBERT2 33
4.5.3 mBERT 33
4.5.4 NB-BERT 33
4.5.5 Ner-Scandi 34
4.6 BERT training 34
4.7 Training metrics 36
4.7.1 Validation Recall and Learning Rate 37
4.7.2 Training loss and validation loss 40

5 Evaluation Methods 45
5.1 Methods 45
5.1.1 Named Entity Recognition 45
5.1.2 Strict Match 46
5.1.3 Named Entity Binary Classification 46
5.2 Evaluating Dataset Results 47
5.2.1 Evaluating NorNE 47
5.2.2 Evaluating the annotated NPSC sample 47
5.2.3 Evaluating the NPSC 48

6 Results and discussion 49
6.1 NorNE results 49
6.1.1 NorNE Named Entity Recognition 50
6.1.2 NorNE Strict Match 52
6.1.3 Binary Classification Evaluation 53
6.1.4 NorNE Results Analysis and Discussion 55
6.2 NPSC results 57
6.2.1 Annotated NPSC sample dataset results 58
6.2.2 NPSC Sample Named Entity Recognition 60
6.2.3 NPSC Sample Strict Match 60
6.2.4 Binary Classification 63
6.2.5 NPSC Results Analysis and Discussion 64
6.3 Comparing NorNE and NPSC results 66
6.4 Possible Limiting Factors for Model Performance 67
6.5 Results Summary 69

7 Conclusions and Future Work **71**
7.1 Contributions 71
7.2 Limitations 72
7.3 Future Work 72

A Extended Training Details **75**

B Extended Results **95**

Bibliography **107**

List of Figures

2.1 The simplest form a Confusion Matrix can take 7

4.1 NorBERT's recall rate for each training epoch. The colored lines represent different training runs on a given seed. With patience counter of 10, training is terminated if a model does not improve on that seed after 10 epochs. 38

4.2 NorBERT's learning rate for each training epoch. The colored lines represent different training runs on a given seed. The model had two attempts to improve its performance before its learning rate was decreased. 38

4.3 Ner-Scandi's recall rate for each training epoch. The colored lines represent different training runs on a given seed. Given the patience counter of 10, training is terminated if a model does not improve on that seed after 10 epochs. 39

4.4 Ner-Scandis's learning rate each training epoch. The colored lines represent different training runs on a given seed. The model had two attempts to improve its performance before its learning rate was decreased. 40

4.5 NorBERT's training dataset loss rate for each seed. Each colored line represent a model seed. All seeds follow the same trend, where the loss rate gets closer to zero for each training epoch. For full numeric details, refer to Appendix A 41

4.6 Ner-Scandi's training dataset loss rate for each seed. Each colored line represent a model seed. All seeds follow the same trend, where the loss rate gets closer to zero for each training epoch, except for epoch number 4 for seed 8, where there is a slight uptick in loss rate, before lowering the next epoch. For full numeric details, refer to Appendix A. 42

4.7 NorBERT's validation dataset loss rate for each seed. Each colored line represent a model seed. Loss rate steadily increase throughout the training. 43

4.8 Ner-Scandi's validation dataset loss rate for each seed. Each colored line represent a model seed. Loss rate steadily increase throughout the training. 43

6.1 Ner-Scandi Binary Classification results for the NorNE test dataset. The 0 class denotes the Not Capital Letter class, that is non-entity tokens. The 1 class denotes the Capital Letter class, that is named entities. 56
6.2 Ner-Scandi confusion matrix results for the NorNE dataset. The numbers are normalized averages that indicate percentage instead of actual count. To see non-normalized version, refer to Appendix B. 56
6.3 NB-BERT confusion matrix results for the NSPC sample dataset. The numbers are normalized averages that indicate percentage instead of actual count. To see non-normalized version, refer to Appendix B. 64
6.4 Ner-Scandi Binary Classification results for the full NPSC dataset. The 0 class denotes the Not Capital Letter class, that is, non-entity tokens. The 1 class denotes the Capital Letter class, that is, named entities. 65

A.1 Configuration file of the NCRF++ model 76

B.1 NorNE results for Ner-Scandi combined average confusion matrix at token level classification 104
B.2 NB-BERT confusion matrix token level classification NPSC sample 105

List of Tables

2.1 How a sentence would be annotated with IO, IOB, IOB2, and IOBES annotation schemes. 10
2.2 Micro-average F1 results from NER on NorNE Bokmål and Nynorsk test dataset (Kutuzov et al., 2021) 15

3.1 NorNE named entity distribution across the train, development, test splits and the average word span of a named entity. Avg. span notes to the average number of tokens a named entity is is composed of. 24
3.2 NorNE total word count, named entity count, and average sentence length. 25
3.3 NPSC total word count, average sentence length, and total capitalized words. 25
3.4 The annotated NPSC sample dataset's named entity count, distribution of type, and the average word span of a named entity. Avg. span notes to the average number of tokens a named entity is is composed of. 26
3.5 The annotated NPSC sample dataset's total count of non-named entities, named entities, combined count and average sentence length. 26
3.6 Distribution of the named entity types, comparing the NorNE test dataset and he annotated NPSC sample dataset. Prop. denotes proportion. 27

6.1 NorNE results for all models. BERT models use the average F1 across all seeds for a given model. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary classification denotes the task where only the detection if an entity is present, without specific type associated. *Ten sentences were dropped from SpaCy pipeline 50
6.2 Ner-Scandi NorNE test dataset results measured with precision, recall and F1 for NER without considering token span of a named entity. Support column denotes number of entries for each class. . 51

6.3 Scandi-ner NorNE test dataset results measured with precision, recall and F1 for NER with the strict evaluation criteria, where the named entity token boundary has to be correct as well as predicted label. Support column denotes number of entries for each class. . 53

6.4 NorNe Binary Classification precision, recall and F1 metrics for all models. The 0 class denotes the Not Capital Letter class. The 1 class denotes the Capital Letter class. The Support column refers to the number of entries for a given class. *10 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length. 54

6.5 The annotated NPSC sample dataset results for all models. BERT models use the average F1 across all seeds for a given model. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary Classification denotes the task where only the detection if an entity is present, without specific type associated. *3 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length. 58

6.6 NB-BERT results for the NPSC sample dataset, measured with precision, recall and F1 for NER without considering token span of a named entity. Support column denotes number of entries for each class. 59

6.7 NB-BERT results for the NPSC sample dataset measured with precision, recall and F1 for NER with the strict evaluation criteria, where the named entity token boundary has to be correct as well as predicted label. Support column denotes number of entries for each class. report 61

6.8 NPSC NorNe Binary Classification precision, recall and F1 metrics for all models. The 0 class denotes the Not Capital Letter class. The 1 class denotes the Capital Letter class. The Support collumn refers to the number of entries for a given class. *729 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length. 62

6.9 Ner-Scandi result comparisons between the NorNE and the NPSC dataset. The NPSC sample are results from NB-BERTS performance. The micro averages are reported for the No Boundary and Strict Match. Macro averages are reported for the Binary Classification. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary Classification denotes the task where only the detection if an entity is present, without specific type associated. 66

A.1 mBERT seed 1024 training metrics 77
A.2 mBERT seed 42 training metrics 77
A.3 mBERT seed 8 training metrics 78
A.4 mBERT seed 37 training metrics 79
A.5 mBERT seed 101 training metrics 80
A.6 NorBERT seed 101 training metrics 81
A.7 NorBERT seed 37 training metrics 82
A.8 NorBERT seed 42 training metrics 83
A.9 NorBERT seed 8 training metrics 84
A.10 NorBERT seed 1024 training metrics 85
A.11 NorBERT2 seed 42 training metrics 85
A.12 NorBERT2 seed 8 training metrics 86
A.13 NorBERT2 seed 101 training metrics 86
A.14 NorBERT2 seed 1024 training metrics 87
A.15 NorBERT2 seed 37 training metrics 88
A.16 NB-BERT seed 37 training metrics 89
A.17 NB-BERT seed 101 training metrics 89
A.18 NB-BERT seed 8 training metrics 90
A.19 NB-BERT seed 42 training metrics 91
A.20 NB-BERT seed 1024 training metrics 91
A.21 Scandi ner seed 1024 training metrics 92
A.22 Scandi ner seed 37 training metrics 92
A.23 Scandi ner seed 42 training metrics 93
A.24 Scandi ner seed 8 training metrics 93
A.25 Scandi ner seed 101 training metrics 94

B.1 mBERT NER results without token span evaluation for the NorNE test dataset 96
B.2 mBERT NER results without token span evaluation for the annotated NPSC sample dataset 96
B.3 mBERT Strict Match evaluation for the NorNE test dataset 97
B.4 mBERT Strict Match evaluation for the NPSC sample 97

B.5 NorBERT NER results without token span evaluation for the NorNE test dataset . . . 98
B.6 NorBERT NER results without token span evaluation for the annotated NPSC sample dataset . . . 98
B.7 NorBERT Strict Match evaluation for the NorNE test dataset . . . 99
B.8 NorBERT Strict Match the annotated NPSC sample dataset . . . 99
B.9 NorBERT2 NER results without token span evaluation for the NorNE dataset . . . 100
B.10 NorBERT2 NER results without token span evaluation for the annotated NPSC sample dataset . . . 100
B.11 NorBERT2 strict match NorNE . . . 101
B.12 NorBERT2 Strict Match for the annotated NPSC sample dataset . 101
B.13 NB-BERT NER results without token span evaluation for the annotated NPSC sample dataset . . . 102
B.14 NB-BERT Strict Match for the NorNE test dataset . . . 102
B.15 Ner-Scandi NER results without token span evaluation for the annotated NPSC sample dataset . . . 103
B.16 Ner-Scandi Strict Match for the annotated NPSC sample dataset . 103

Chapter 1

Introduction

Named Entity Recogntion (NER) is an important in the research field of Natural Language Processing (NLP). NER is used to efficiently identify and categorize real-world entities such as people, organizations, locations and more within text data. This process is foundational in automated information extraction and retrieval systems, where NER applications can used to extract useful and structured information, from unstructured text.

With the advent of Transformer based Large Language Models (Vaswani et al., 2017), and the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019), the NLP reasearch community has witnessed a paradigm shift in the state-of-the-art performance on a wide array of tasks, including NER.

Most NER research tend to focus on identifying and categorizing named entity types, rather simply detecting their presence. Proper capitalization of named entities could be a dead giveaway that a named entity is present, however, in real-world situations such as in the case of automated speech-to-text captions or user-generated text content, capitalization mistakes are not uncommon.

This Master's thesis takes a slightly diverged approach to NER research compared to traditional methods. Instead of using text data with normal capitalization, all the data we will use are lowercase. Furthermore, rather than focusing exclusively on precise classification of a named entity types, we will also focus on the detection of named entities regardless of type.

By researching the detection of named entities in texts with capitalization mistakes, our aim is to enhance our understanding of the challenges associated with NER in real-world situations where text data may contain capitalization errors. These real-world scenarios are particularly relevant to automatic speech-to-text transcripts, as this thesis is conducted in collaboration with TV2, Norway's second-largest broadcast producer. The proposed approach in this thesis aims to enhance capitalization accuracy in TV2's automatic speech-to-text captioning systems. However, automated speech-to-text caption and transcription systems are not only important for large media institutions to alleviate manual transcription labour, but they also play a beneficiary role for smaller businesses and individual

users creating transcripts and captions for audio and audio-visual media.

1.1 Problem Statement

TV2 applies speech-to-text technology to generate transcripts of TV programs. As a point of interest, they are exploring approaches to ensure acceptable quality of the automatically generated speech-to-text data. This project aims to explore methods of how to ensure that all named entities in Norwegian texts are properly capitalized, utilizing state-of-the-art neural network approaches for NER.

1.2 Research question

Aim: Create an effective method to detect named entities and categorize them into their respective type, utilizing state-of-the-art neural network models when text is lowercase.

The research questions are as follows:

(R1) To what extent can current neural models be used to identify non-capitalized named entities?

(R2) Which current neural models achieves best NER results in all lowercase texts?

1.3 Thesis Outline

This chapter included a brief statement about the task description and problem statement. The remainder of the thesis will be divided into six chapters, excluding the supplementary appendices.

Chapter 2 focus on providing background material necessary for the un-der-stand-ing of our work. This includes fundamental knowledge of NER research, such as evaluation methods, various artificial neural network model architectures, and previous research which used neural network models to tackle NER tasks.

Chapter 3 provide detailed information about the datasets that is used in the thesis' experiments.

Chapter 4 provides a detailed description of the experimental setup and model training techniques and preliminary results gathered from the validation dataset during training. The chapter also includes additional details for every model we utilize, and a thorough description of the experimental setup. Thorough documentation is essential in making this study easy to replicate and validate.

Chapter 5 describes three different methods that will benchmark the models' performance, named entity recognition without token span boundary, strict match classification, and binary classification. These task will serve as the evaluation methods used to assess the performance of our models.

Chapter 6 present the results gathered from the three different evaluation methods, described in Chapter 5. The results from our models are compared to one another and discussed, highlighting their strengths, weaknesses, and their effectiveness.

Chapter 7 provides answers to our research question and conclusions for our findings, in addition to the project's limiting factors, and possible future work.

The thesis include additional details in its appendices. Apendix A provides extensive detail regarding training metrics for each model. Apendix B provides test results for additional models that did not get highlighted in the Chapter 6.

Chapter 2

Background

2.1 Natural language processing and sequence labeling

Machine Learning (ML) relies on training data to recognize patterns and make predictions or take actions based on what has been learned. Natural Language Processing (NLP) focuses on language, as the name suggests. It involves creating computer programs that can process and interpret human language, be it spoken or written. In contemporary NLP applications, it is common to deploy machine learning in its development. Moreover, in text-based NLP, vast amounts of text data often serves as the foundation for the ML training process.

For task specific NLP applications, text corpora intended for training and testing will often be split into three different datasets: One dataset for the actual pattern training, one development dataset (may also be referred to as the validation dataset) for optimization of training, and a test dataset to access an applications ability to generalize to new, unseen data.

For example, text-based sentiment analysis classification, the ideal training corpus would include context-rich content that can effectively conveys sentiments. A suitable dataset would be corpora that contain various reviews, along with metadata for each review regarding its class of sentiment, such as negative, neutral or positive. With this metadata, a machine learning application could learn specific patterns associated with a given class, from which the application would later make its sentiment predictions on test data.

In other cases, the objective may be sequence labeling tasks, such as part-of-speech (POS) tagging. This task involves assigning a value to each token in a sequence, representing its grammatical role, such as an adjective, noun, or verb, among other descriptors. Token is a term that will be used throughout this thesis. A token can be an instance of a character, such as punctuation, and a group of characters, such as a word. Later on in the thesis, we will give descriptions of datasets. For example, if a dataset is said to contain 1 000 tokens, that will be a combined count of words and punctuation. In other instances of sequence labeling tasks, the focus may solely to identify tokens that are proper noun and categorize

them, a task known as Named Entity Recognition.

Named Entity Recognition (NER) involves classifying tokens with different named entity types, such one or more tokens in sequence being classified as a person, location, organization or non-named entity type, that is all tokens that are not proper nouns (Jurafsky and Martin, 2022, p. 160). These aforementioned classes are the most commonly used throughout NER research and ML model testing, however, sometimes more granular types which provide more specificity are also used. Among these are geopolitical entities such as nations and states, monetary values, time and dates, and miscellaneous entities such as products, events, and works of art.

In other instances, a NER task could revolve around having domain-specific named entities (DSNE). In such instances one would need highly specific corpora, tailored to the given task. For example in domain-specific NER within biology and health, a model could be trained to predict names of proteins, genes, enzymes, and illnesses in text (Smith et al., 2008).

NER is considered a subtask within information extraction, where the primary goal is to extract valuable information from text. Identifying and tagging named entities is a crucial component of this process. By deploying an accurate named entity labeling classifiers, a information extractor's performance can be improved in identifying relationships between entities (Jurafsky and Martin, 2022, p. 165). Take the following example: "Citing high fuel prices, United Airlines (ORG) said Friday (TIME) it has increased fares by $6 (MONEY)." It is relatively easy for humans to understand the sentence's content: an organization x announced on day y that the prices of something would increase by z amount. However, if the sentence was part of a larger document, extracting this information would be much more time-consuming for a person. In contrast, an information extraction model could efficiently gather and structure the data, and store it for later use.

2.2 Evaluation methods and analysis for NER

To evaluate a sequence labeling classifier's performance, its predictions have to be compared against the human-defined label, often referred to as the *gold standard* (Jurafsky and Martin, 2022, p. 68).

There are several ways to assess a NER model's performance assessment instead of just evaluating the predictive power for a single token at the time. For example a person's name could be composed of a first name, a middle name, and a surname. Then the named entity would then span a boundary of three tokens.

In the following section, we will present some of the most common way to analyse and assess performance for a NER system.

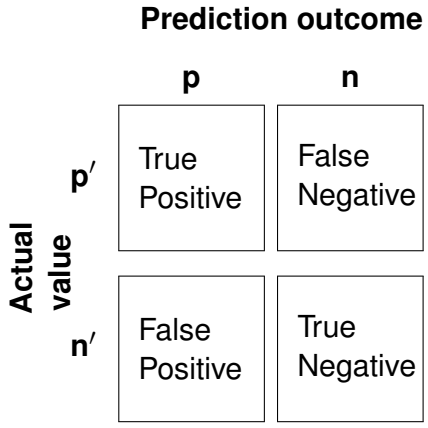


Figure 2.1: The simplest form a Confusion Matrix can take

2.2.1 Confusion matrix

A model's output can be compared to the the gold standard with the use of a *confusion matrix*. Figure 2.1 presents the simplest form of a confusion matrix, a binary classification confusion matrix. The matrix can be extended by as many dimensions as there are classes to predict, where each cell always refers to one possible outcome.

A prediction can be a *True Positive*, a *True Negative*, a *False Negative*, or a *False Positive* of a given class we want to predict. A common example that explains the differences between prediction types is classifying emails as "spam" or "not spam":

- *True Positive (TP)*: The prediction is positive for a spam, and it's correct. The email is spam.
- *True Negative (TN)*: The prediction is negative for spam, and it's correct. The email is not spam.
- *False Positive (FP)*: The prediction is positive for spam, the prediction is incorrect. The email is not spam.
- *False Negative (FN)*: The prediction is negative for spam, but it is incorrect. The email is spam.

2.2.2 Accuracy, Precision, Recall and F1

A common approach for assessing a classifier's performance, including NER classifiers, is to examine performance metrics, such as *accuracy*, *precision*, *recall* and *F-measures*. Each class, or named entity type for NER systems, can be assessed individually, and the performance overall can be measured. The aforementioned metrics are calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Accuracy is calculated by the total sum true positives and true negatives, divided by the total number of samples. For datasets with significantly class imbalance, accuracy is often not the preferred measurement to determine a model's overall predictive power. For these kinds of datasets precision and recall would be preferred (Jurafsky and Martin, 2022, p. 68).

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Precision measures the percentage of all the predictions that were given to a specific class and were actually that class. It measures how many true positives there were out of all the positive predictions for a specific class.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Recall measures, out of all the occurrences of a given class, what percentage was correctly identified by the model. It measures how many true positives there were out of all the actual positives for a specific class.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

F1 is a metric that combines precision and recall metrics, where a perfect F1 score of 100 equates that every prediction is correct. The most common variation of F1 weights recall and precision equally. However, for many classification tasks there are often more than two classes to predict. We can examine the F1 for each class, or the overall system performance using *macro-average* and *micro-average* F1. The macro-average will compute the average of all classes F1, weighing each class equally important (Jurafsky and Martin, 2022, p. 70). Micro-average calculates its average by aggregating the contributions of all classes. In cases where datasets have a significantly unbalanced number of samples for each class, micro-average may preferred, depending on if we favor the importance of some bigger classes. However, if all classes are equally important, macro-average is often the preferred measurement.

Considering that named entities frequently span multiple tokens, assessing boundary matches can also offer valuable insights into the model's performance,

using the same metrics as discussed in the last paragraph. The token span boundary simply refers to how many tokens a named entity consists of. For example the name "James Paul McCartney"spans three tokens of the person PER type, therefore the token boundary is also three. Segura-Bedmar et al. (2013) define four different evaluation methods in terms of token boundary and match of named entity type:

- **Strict match**: Exact match for both named entity type and its token boundary.
- **Exact boundary:** A named entity is identified. Type could be wrong, but the boundary of the named entity is correct.
- **Partial boundary:** A named entity is identified, but the type might be wrong and boundary is partially correct.
- **Type match:** Named entity type is correct, but the predicted tokens does not match the named entity's token boundary.

2.3 Label encoding schemes

There are several ways of annotating tokens as named entities in text. Table 2.1 presents four different annotations schemes[1], IO, IOB, IOB2, and IOBES respectively. In practise, the differences are how the of the token boundary annotated a given named entity.

IO does only differentiate if a named entity is inside or outside a token span. IOB, and IOB2 adds another layer of specificity, where the beginning of a named entity will get "B-" for beginning label associated with it. IOB and IOB2 are quite similar, however, IOB2 will differentiate between separate named entities even when they are of the same type and appear consecutively in a sequence of tokens (Jørgensen et al., 2020).

If a named entity spans across multiple tokens, IOBES will specify the last word with an ending "E-" label and specify if there is a single named entity, with the "S-" label (Jurafsky and Martin, 2022, p. 166).

2.4 English data

There are several NER annotated datasets in English, such as CoNLL-2003 (later revised to CoNLL-2003++), WikiNeural and OntoNotes, to name a few popular ones. The number of named entity types in a dataset may vary, where one dataset may use more or fewer types than the other.

[1]The four annotations schemes are also commonly referred to as IO, BIO, BIO2, BIOES

Words	IO	IOB	IOB2	IOBES
Thai	I-ORG	B-ORG	B-ORG	B-ORG
Airways	I-ORG	I-ORG	I-ORG	I-ORG
International	I-ORG	I-ORG	I-ORG	E-ORG
flight	O	O	O	O
to	O	O	O	O
Bangkok	I-ORG	B-LOC	B-LOC	S-LOC
was	O	O	O	O
delayed	O	O	O	O
.	O	O	O	O

Table 2.1: How a sentence would be annotated with IO, IOB, IOB2, and IOBES annotation schemes.

The CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003a) use four entity types: location (annotated as `LOC`), organization (`ORG`), person (`PER`), and miscellaneous (`MISC`). `MISC` refers to any named entity that does not fit into category of any other aforementioned named entity types. The dataset consisting of about 300 000 tokens. The dataset is split in a typical 80/10/10 distribution, meaning that 80% of the data is for training, 10% is of the data is the development dataset, and the remaining 10% is the test dataset (Tjong Kim Sang and De Meulder, 2003b). In 2019 the dataset was revised by Wang et al. (2019) where 5.38% mislabelled test sentences were corrected to form the CoNLL++ dataset.

One of the largest publicly available datasets is the WikiNeural dataset, which is split into nine different language-specific sets, where each set has an average token size of over 2 million. The sets are produced with a silver-standard, meaning it is using an automatically annotated approach. The corpora are made up of Wikipedia articles, and use the same four named entity types as the CoNLL-2003 dataset (Tedeschi et al., 2021).

Compared to the previously mentioned datasets, the OntoNotes 5.0 operates with 18 named entity types for greater named entity classification specificity. Examples of these additional types includes names of human-made structures or facilities (annotated as `FAC`), monetary values (`MONEY`), and names of laws, legal codes, or regulations (`LAW`). The English version consists of roughly 1.5 million tokens which are acquired from various genres of text such as news, broadcasts, and conversational telephone speech (Pradhan et al., 2013).

2.5 Norwegian dataset – NorNE

In Norwegian there is one large publicly available named entity dataset, dubbed NorNE (Jørgensen et al., 2020). It was developed in collaboration by Schibsted Media Group, the Language Technology Group at the University of Oslo[2], and The Language Bank at the National Library of Norway[3].

NorNE use 7-9 named entity types depending on the version one want to use. Additionally, the dataset is split into both Norwegian Bokmål and Nynorsk. As this dataset will be the thesis' primary dataset for model training, more detail about NorNE will be given in Chapter 3.

2.6 Traditional approaches to NER

Provided it is not the first word of a sentence, capitalization is a strong indicator that a word is a named entity in English and many European languages. If a language orthography, the conventional spelling system of a language, does not use capitalisation, we have to be reliant on other cues, which is the case for several South Asian script languages. In these instances, suffixes could be good indicators of named entities, but some instances of names does not occur with any prefix or suffix (Riaz, 2010). Moreover, for languages with or without capitalization, lexical patterns and semantics are important indicators for named entity presence (Li et al., 2022).

Throughout the evolution of NER research, there have been techniques to used to detect and classify named entities. For example, rule-based approaches which rely on hand-crafted rules. Contrary to supervised ML methods, rule-based approaches do not need annotated data. The method would often utilize domain-specific gazetteers, and lexical patterns for its predictions (Etzioni et al., 2005; Sekine and Nobata, 2004). However, its not uncommon for a rule-based approach to perform poorly in domains that it was specifically designed for, in turn making it difficult for the these models to generalize across different datasets. (Goyal et al., 2018). Even though rule-based NER techniques have fallen out of fashion with the introduction of neural models and pre-trained large language models, rule-based can still be quite effective. Particularly for languages that lack features that would otherwise strongly indicate that a word is a named entity (Goyal et al., 2018; Riaz, 2010).

In the early to mid-2000s, unsupervised learning approaches gained popularity for similarity reasons as rule-based approaches. That is, both methods are not reliant on labeled training data (Nadeau and Sekine, 2007). Named entities would often be extracted out of unlabeled data by the utilization of distributional statistics coupled with context similarity of a clustering approach. Furthermore,

[2] https://www.mn.uio.no/ifi/english/research/groups/ltg/
[3] https://www.nb.no/sprakbanken/en/sprakbanken/

a association rule-based approach would detect association between items within datasets (Goyal et al., 2018).

Supervised machine learning became more popular as more high-quality training data became accessible, and is still relevant today. When training a supervised ML sequence labeling model, a common technique is to label each token according to its type. Supervised ML leverages labeled data to learn pattern based features and patterns for a given label type, and to distinguish it from other types. This is the goal when creating a model that can generalize, and classify correctly even when the test data is not similar to the training data (Goyal et al., 2018). Supervised machine learning can be effective, but it may have challenges, particularly regarding the availability of training data. The process of labeling large amounts of data can be labor-intensive and time-consuming, especially when creating large high-quality datasets.

Before transitioning to neural network methods, two highly influential models were used, the Hidden Markov Model (HMM) and Conditional Random Fields (CRF).

HMM is a variation of the Markov chain model, which suggests that to predict the future in a sequence, everything depends solely on the current state, with no influence from any other states. The HMM works by considering both observed events (such as a token in NLP) and hidden events (such as part-of-speech token label or named entity token label). These events are the causative factors for the probabilistic model (Jurafsky and Martin, 2022, p. 167-174).

CRFs are similar to HMM, however, while HMMs focus on maximizing the joint probability during training, CRFs are based on maximizing the conditional probability in the training dataset. Consequently, for sequence labeling tasks, CRFs makes its predictions by examining the relationships between each word and its surrounding context, relying on patterns learned from the training data (Jurafsky and Martin, 2022, p. 174-179).

2.7 Neural approaches

Throughout recent years, neural architectures in various combinations have been more commonly used to tackle NER tasks. The origins of artificial neural networks lie in the McCulloch-Pitts Neuron, which is a simplistic model that imitate the neural structure of the human brain (Jurafsky and Martin, 2022, p. 29).

Artificial neural networks architectures are typically described in terms of three types of layers. The first part is the input layer, which brings the initial data into the network for further processing. The input layer is connected to a hidden layer, moreover, the hidden layer is connected to the output layer. Calculations are made between the different layers as values are forward propagated in the network, influenced by weights values, bias and the activation function used in the different layers.

The simplest form of the artificial neural network is the *Feed forward neural network* (FFNN). This architecture is based on that all neural units from one layer are connected to the next layer, resulting in a fully connected network.

Even with FFNN's relatively simple architecture, compared to successive neural network architectures, the FFNN can be quite effective at various NLP tasks. A novel, but effective approach was presented by Collobert et al. (2011) using a FFNN, used for various NLP tasks with results being close to state-of-the-art at the time, including NER results. Owing to large unlabeled datasets, the modeled relied on discovering useful features for their various tasks, and used word-level log-likelihood, the sentence-level log-likelihood, and gazetteers. Their model achieved an impressive F1 of 89.59 on the CoNLL-2003 English dataset.

However, there are other types of neural networks that are specifically designed to handle sequential data, such as tokens in a sentence. Introducing the Recurrent Neural Networks (RNNs), where unlike Feed-Forward Networks (FFNNs), RNNs can capture dependencies and context within a sequence, rather than treating each input as independent (Jurafsky and Martin, 2022, p. 186). However, note is that not all information in a paragraph or an entire article carries equal importance. In fact, information is often quite localized, which the standard RNN struggles to capture. This has to do with how the network backpropogate to adjust its weights. Backpropagation involves repeated multiplications determined by the sequence's length. Issues arise with long sequences, as the gradient tends to approach zero during training when adjusting weights (Jurafsky and Martin, 2022, p. 198). Long Short-Term Memory (LSTM) networks are designed to address this long-term dependency issue caused by vanishing gradients. A key feature of the LSTM architecture is its ability to retain useful information within a sequence while discarding information that is no longer needed to maintain context, using specialized neural units (Jurafsky and Martin, 2022, p. 199-200).

An improved iteration of the LSTM was later introduced, called the bidirectional LSTM (Bi-LSTM). It works in the same way as a traditional LSTM, but there is one key difference. Instead of processing sequential data only in the forward direction, a Bi-LSTM will process a sequence in both the forward and the backwards direction. This allows the output units to compute dependencies in both the past and future, instead of just computing what is ahead in time. This key difference improved performance in NER, as demonstrated by Huang et al. (2015), they tested various LSTM and Bi-LSTM configurations on the CoNLL-2003 English dataset. Their LSTM model resulted in a F1 score of 83.74, whereas the Bi-LSTM model resulted in a F1 of 85.17. However, both of these models perform significantly better with a CRF inference layer, as the LSTM-CRF yielded a F1 of 88.36 and the Bi-LSTM-CRF a F1 of 90.10.

A network type not discussed thus far is the convolutional neural networks (CNN). This network type has been particularly successful in deep learning computer vision tasks, and NLP. A significant advantage of the CNN is that it

is considerably less computationally heavy than many other types of networks (Albawi et al., 2017; Goodfellow et al., 2016). This is, perhaps, a reason why CNN has been often used in combination with other types of networks as inference layer, or character and word level encoder.

Various Bi-LSTM and CNN models were tested by Chiu and Nichols (2016) on the CoNLL-2003 and OntoNotes 5.0 datasets. The model combinations varied in terms of the utilization word embeddings, character level embeddings, as well as some feature engineering included. They concluded that the best model was Bi-LISTM+CNN with embeddings and lexical features, resulting in a F1 score of 91.62 for the CoNLL-2003 English dataset, and 86.28 for the OntoNotes dataset.

Yang et al. (2018) had similar findings in terms of results on the CoNLL-2003 English dataset, in their paper "Design Challenges and Misconceptions in Neural Sequence Labeling". This comparative neural architecture study for sequence labeling tasks, including NER as one model performance benchmark. The architecture that exhibited the best results for NER was the CLSTM+WLSTM+ CRF and the CCNN+ WLSTM+CRF models[4], utilizing GloVe embeddings, which resulted in a F1 of 91.08 and 91.11 respectively.

The same year, Yang and Zhang (2018) published an easy customizable open-source framework for sequence labeling. The framework could be configured with different neural network types and encoders, whether that would be character, word level, or both combined. The framework also include the option to add custom embeddings or create its own from the input training data. With this framework, dubbed NCRF++, the researchers were again able to get a F1 score of 90-91, depending the framework configuration, on the CoNLL-2003 dataset.

Jørgensen et al. (2020) tested the NCRF++ framework on the NorNE dataset, and found that combining training data for both Bokmål and Nynorsk would result in the best performance. Tests on NorNE's Bokmål test dataset resulted in a F1 of 88.03 and for Nynorsk a F1 of 83.48. Jørgensen et al. (2020) argues that a joint model is a double win since it not only yields increased or comparable performance compared to a model trained on only one language, but also means that only one model would need to be maintained for Norwegian. Although the performance was not as high as observed in Yang and Zhang (2018)'s tests on the CoNLL-2003, the results which Jørgensen et al. (2020) observed are still impressive considering they use about twice as many named entity label types.

NCRF++ is still a powerful framework for sequence labeling tasks, however, transformer base architectures could be considered as current state-of-the-art technique to use. In Devlin et al. (2019) paper where BERT was initially introduced, NER was tested using a fine-tuned $\text{BERT}_{\text{BASE}}$ which resulted in an impressive F1 of 92.4 and $\text{BERT}_{\text{LARGE}}$ with a F1 of 92.8, on the CoNLL-2003 dataset. Since BERT is highly relevant for this thesis, we will revisit BERT with

[4]The first "C" in the LSTM/CRF denotes *character level encoding*. Similarly, "W" denoting *word level encoding*.

additional details in Section 2.10.1.

Kutuzov et al. (2021) compared NER results from their configuration of the NCRF++ framework, and a fine-tuned multilingual BERT (mBERT), NorBERT and NB-BERT on the NorNE Bokmåk and Nynorsk test dataset. In Table 2.2 we present Kutuzov's findings. NCRF++ and NorBERT yields comparable results, whereas mBERT performs the worst among the four tested models. However, the F1 observed using NB-BERT for both the Bokmål and Nynorsk is significantly higher than the other models tested, reaching a F1 of 90.2 for Bokmål and 88.6 for Nynorsk.

	Bokmål	Nynorsk
NCRF++	83.5	85.3
mBERT	78.8	81.7
NorBERT	85.5	82.8
NB-BERT	90.2	88.6

Table 2.2: Micro-average F1 results from NER on NorNE Bokmål and Nynorsk test dataset (Kutuzov et al., 2021)

It is important to note that the aforementioned studies are different from what will be presented in this thesis, since their results stem from NER evaluation where the models are subjected to test datasets with proper capitalization. Bodapati et al. (2019) claims that NER models are sensitive to capitalization changes. Their finding is that if a model was trained on standard capitalization, the model's performance suffered immensely if the model was tested on all lower case texts. For example, their ELMo model's performance went from a F1 of 92.0 on the normal cased CoNLL-2003, down to 34.8 when the data was all lowercase. An even sharper decline in performance was observed with their Lample et al. (2016) inspired Bi-LSTM mode, which went from a F1 of 90.8 down to an abysmal F1 of 0.4.

Bodapati et al. (2019) also trained and tested models with all lowercase data. The ELMo model's performance resulted in a F1 of 89.1, compared to 92.0 in normal casing training and testing. Whereas the Bi-LSTM's performance resulted in a F1 of 85.7, compared to the aforementioned 90.8 with normal casing conditions.

Bodapati's study is the only one of the highlighted studies in this section that tests models with text data with various text casing conditions. Their findings indicate that models can still have great performance for all lowercase text, if the model is trained on all lowercase data. The findings serves as a precursor to what to expect for similar studies when using all lowercase data.

2.8 Challenges in NER

For humans, identifying named entities may be a trivial task, however, for computers, it is not. As discussed in previous sections, there are various ways of automating named entity recognition, ranging from rule-based approaches and simple neural networks to more complex pre-trained large language models. Yet as we have examined in Section 2.7, an error rate is still expected to be about 1/10 for named entity. These results would indicate that named entity recognition is not without its challenges. Word ambiguity, token span, and grammatical rules are some of these challenges. In this section, we will discuss what challenges NER models faces in further detail.

2.8.1 Out-of-vocabulary

Out-of-vocabulary (OOV) words, word-tokens that were not present in the training data, but appear in test data, could be challenging for a NER model to identify correctly.

While gazetteers, large knowledge bases of known named entities, can be helpful when training data are limited, they cannot be relied upon solely to identify named entities. This is because the open nature of names, resulting in that new names are constantly being created, requiring knowledge bases to be updated frequently. Also Although without the use of gazetteers, ambiguity may arise whether a token or tokens are named entities. Especially for when names include what would normally been nouns, verbs or adjectives. Take for example“Burning Man” or “Apple”.

Brüggemann (2012) discuss, sentences like “Jobs said” and “Jobs are hard to find” have vastly different meanings. One technique to infer whether a word is a proper noun is to use text n-gram features to aggregate absence or presence of given word(s) before and/or after the ambiguous word (Ganti et al., 2008). An example of this could be “What is the probability that (said|Jobs) where “Jobs” is a named entity.

2.8.2 Named entity span

Another challenge for NER is to correctly identify the boundary of multi-word named entity. Sometimes named entities spanning multiple tokens, have single tokens where each could be a named entity on its own, however, in combination should only be one named entity. Take the example of the company “Sunnmøre og Romsdal Fiskesalslag” (`ORG`). Should the tokens be split into Sunnmøre (`LOC`), Romsdalen (`LOC`) and “Romsdalen Fiskesalslag” (`ORG`) or a conjunction of all tokens?

Identifying multi-word entities can be challenging since it requires determining the word boundaries, which can be especially difficult for entities that were

not present in the training dataset.

2.8.3 Capitalization

During the training of a machine learning model, recognizing capitalization patterns can be valuable for identifying named entities, however, depending solely on capitalization for named entity recognition is insufficient. For one, proper punctuation often indicate the beginning and the end of a sentence. This creates an obvious capitalization pattern, however, not every first word in a sentence is a named entity.

Secondly, not to rely solely on word capitalization for the recognition of named entities, is that for many languages, the first word of every sentence utilize capitalization (Riaz, 2010). Thirdly, for languages that utilize capitalization of words, grammatical rules for how and when to use them differ. For example German use capitalization for every noun, regardless if it is a proper noun or not (Pauly and Nottbusch, 2020).

Lastly, text data can be affected by human errors, such as incorrect capitalization or simple typos, which may disrupt a model's ability to recognize its learned patterns. Although, NER systems may continue to improve their accuracy over time, it remains uncertain whether these experimental results will translate to equal performance in real-world, non-curated datasets such as media posts or other user generated content. In NER research, rarely text riddled with grammatical errors tested for a NER model's performance, nevertheless. As for capitalization, Bodapati et al. (2019) findings suggest that NER models are very brittle to changes in capitalization if the model is not specifically trained for it.

2.9 Loss, Optimizers and Regularization for Training

In machine learning, the term "loss" refers to a value that represents how well a model's predictions match the true values. During training, it is important to calculate the loss. For classification tasks, it is common to use the *cross-entropy loss* function. The cross-entropy loss function measures the difference between the predicted value and the gold standard, where its output is a value between 0 and 1. A lower loss value indicates that the model is closer to its prediction target, which is what we ideally want to achieve during the training (Jurafsky and Martin, 2022, p. 90).

It is common for a model to go through several of training epochs, which refers to a single iteration of training. After each epoch, weights gets adjusted through backpropagation in an attempt to improve its predictions for the following epoch. An optimizer algorithm is used to adjust weights in an attempt to minimize the loss function. For example, an optimizer could adjust the learning rate at an dynamic or static rate. High learning rates will cause large updates to a network's

weights, whereas smaller learning rates give rise to smaller updates to the weights.

A common problem when training mode is called, *overfitting*. It refers to when some features seen in the training data gets problematically high weighted assigned to it. This could result in that when a model observes a set of features in the training data, it will overemphasize the importance of those features. Overfitting often results in bad predictions when the model is tested on data that was not seen in the training data.

Regularization helps to mitigate overfitting. Simply put, regularization is a way of regulate the different weight's importance. This can be done by implementing various regularization measures, such as using methods that introduce weight penalties of various sorts, such as L1 and L2 regularization. However, these L1 and L2 regularization can be computationally heavy for larger models compared to applying *dropout* (Srivastava et al., 2014).

Dropout is the technique when training a model, neural nits will temporarily be disabled in the network, for both incoming and outgoing connections. This helps to reduce overfitting by preventing certain nodes from relying too heavily on others. Srivastava et al. (2014) suggest a dropout probability of 0.5, which means there is a 50% chance a node will be disabled, is observed to be an optimal dropout value for many types of neural network architectures.

2.9.1 Feature Representation and Word Embeddings

To enable a model to interpret data, we can represent a sequence of text, such as words, characters, or a combination of both as numerical valued feature representations. These numerical values are formed in a vector or tensor space which forms word embeddings.

Word embeddings are highly relevant and often used in NLP tasks. Word embeddings tackle the task of comparing word similarity (Jurafsky and Martin, 2022, p. 107-108), derived from the distributional hypothesis (Harris, 1954), stating that words are similar if they appear in similar contexts. Take the following examples

> I took my *dog* to the veterinarian.
> I took my *cat* to the veterinarian.

According to the distributional hypothesis, cat and dog are similar words as they appear in similar contexts. Other words that are very much dissimilar, such as "thought" or "uboat". Hardly ever would it make sense to replace "thought" with "uboat", without making the sentence completely nonsensical.

Word embeddings are often pre-trained before it is used in NLP application, having dense vector space representation in dimensions ranging from 50 up to 600. The training of these embeddings are acquired using unsupervised machine learning on large amounts of data text data, where the distributional hypothesis

serves as a foundation. However, it is also possible to acquire word embeddings from the NLP task specific training data, initializing embedding vectors to random values and allowing the training algorithm to tune these values based on the training data (Jurafsky and Martin, 2022, p. 107-108).

Word2Vec

A model learns one fixed embedding for each word in a dataset, making word2vec a static embedding type. The word embedding method captures syntactic and semantic regularities by training a binary classifier on predicting is word w_1 likely to appear near w_2 (Jurafsky and Martin, 2022; Mikolov et al., 2013). Mikolov et al. (2013) propose two architectures; The Continuous-Bag-of-Words (CBOW) architecture seeks to predict w_1 given the c context of surrounding words, while the Skip-gram architecture attempts to predict the c context given w_1.

fastText

An extension of the skip-gram method was proposed by Bojanowski et al. (2017) in the fastText method. Rather than ignoring word morphology and assigning unique vectors to each word, fastText considers sub-word information by utilizing character n-grams. This approach results in morphologically similar words being assigned similar vectors

GloVe

Global Vectors (GloVe) is a widely used vector embedding model. GloVe's approach is to capture the ratios of probabilities from both word-word global statistics and local statistics of a corpus, by combining predictive methods such as word2vec and Positive Pointwise Mutual Information (PPMI) models (Jurafsky and Martin, 2022, p. 125).

ELMo

Peters et. al propose what they call *deep contextualized* word representations model, named ELMo (Embeddings from Language Models). The ELMo architecture tries to address previous word embedding architectures' shortcomings, such as complex characteristics of word use, and how word polysemy is problematic for static models (Peters et al., 2018).

BERT

Transformer-based language models are considered the state-of-the-art word embedding standard. One widely used model is the BERT (Bidirectional Encoder Representations from Transformers) model, which was introduced by Devlin et al. (2019). BERT further improves the understanding of the contextualized

and polysemy of words by introducing attention mechanisms in its architecture (Vaswani et al., 2017).

2.10 Large Pre-Trained Language Models

Large Pre-trained Language Models like various BERT models and the generation of different Generative Pre-trained Transformers (GPTs) (Brown et al., 2020) models can be used for a wide variety of NLP tasks. For BERT models, it is common practice that the language model is trained on a large text corpus, to then be fine-tuned to fit downstream NLP tasks such as NER in our case.

BERT models are neural networks-based, but they operate differently compared to their predecessors like Bi-LSTM. The foundation of these models is based on attention mechanisms, which were introduced in the seminal 2017 paper "Attention is all you need" (Vaswani et al., 2017), where researchers at Google used the concept of attention mechanisms to propose the Transformer architecture.

An attention-based mechanism allows a model to predict an output item based on parts of the input that the model deems most relevant. This prediction is based on the ability to compare the item to a collection of other items that reveals their relevance in the given context. When a set of these comparisons is used to evaluate the output for the current input, this process is called *self-attention*. Self-attention is the key design feature of the transformer (Jurafsky and Martin, 2022, p. 212).

Transformer models are composed of multiple transformer blocks stacked on top of each other. Each transformer block is a combination of a multi-head self-attention mechanism and simple linear layers feed-forward networks. Models such as the $\text{BERT}_{\text{BASE}}$ model types, which will be used in the experiments described later in this thesis, make use of 12 transformer blocks (Devlin et al., 2019).

2.10.1 BERT Pre-Training and Fine-Tuning

Bidirectional Encoder Representations from Transformers (BERT) uses a *masked language model* (MLM) architecture when it is pre-trained. The pre-train of a BERT model (the training that is involved before the model is used for downstream NLP tasks) is split into to parts. The first task of pre-training the BERT model involves randomly masking a certain percentage of tokens in the training vocabulary and predicting the correct word based on its context. This is achieved using a deep bidirectional transformer that analyzes the masked item's sequential context from both left-to-right and right-to-left directions. The bidirectional nature of the transformer allows the model to take into account both past and future context when making predictions. This is similar to the bidirectional processing in a Bi-LSTM, where the hidden state at each time step

is a function of both past and future context (Devlin et al., 2019).

The second task involves predicting the next sentence, which is especially important for tasks such as question answering systems, and natural language inference that rely on understanding the relationships between sentences. These pre-training steps enable the model to better comprehend contextual relationships and improve its performance on language tasks (Devlin et al., 2019).

The pre-training of a BERT model involves unsupervised learning over a large existing corpus of text. In the original BERT model proposed by Devlin et al. (2019) in their seminal paper a corpus of 3.3 billion words was used with 75.8% of the words extracted from the English Wikipedia and the remainder from the BooksCorpus (Zhu et al., 2015).

To find relationships between words within a sentence, a typical BERT base model uses 110 million[5] parameters, meaning the model has 110 million tunable weights. Thanks to the large numbers of parameters, the models are able to capture relationships between different tokens, as well as rich, context-aware representations. These parameters are includes layers and attention heads, which enables the model the to form the hierarchical representation of a given input. The number of layers determines the extent of the hierarchical representations, while the attention heads is responsible for capturing various relationships between the input tokens. Increasing the amount attention heads is associated with having a model that is able to form more types of relationships simultaneously in an input.

A BERT model's training parameters are highly customizable and can be reconfigured as needed. Depending on their configuration, the larger BERT models can often have multiple times the amount of weights in the base model, and more attention heads, more layers, larger hidden layer sizes and larger sizes of its vocabulary. After a model is pre-trained, it has to be fine-tuned to be able to carry out task specific NLP challenges, such as sequence tagging. Fine-tuning could involve feeding task-specific inputs into the pre-trained BERT and fine-tuning its weights by adding a classification layer to the model. For NER fine-tuning, one would input tokens and include annotations for each and every token for the model to able to learn what to predict.

2.10.2 Language specific BERT models

Since the introduction of BERT and its open source framework, numerous language specific BERT based models have emerged. Kutuzov et al. (2021) introduced NorBERT, and in parallel efforts the AI Lab of the National Library of Norway released NB-BERT (Kummervold et al., 2021).

Norway has two official standards for written Norwegian, *Bokmål*, and *Nynorsk*. NorBERT was trained for both standards, with training data consiting

[5]More info specifications about various BERTs' parameters: https://huggingface.co/transformers/v3.3.1/pretrained_models.html

of roughly 1.9 billion tokens. The majority of tokens was derived from news text, sourced from the Norsk Aviskorpus, but text from the Norwegian Wikipedia was also used.

Likewise NB-BERT (Kummervold et al., 2021), was trained on both Bokmål and Nynorsk data, though with a much larger corpus, consisting of roughly 18 billion tokens. As well as Norwegian text data, the training data also consisted of about 4% English text and 1% mix of Sami, Danish, Swedish, and a few traces from other languages.

2.10.3 Large Pre-trained Language Model Biases

The large generative language models such as GPT-2[6] and GPT-3[7], use large amounts of training data based on the "Common Crawl"[8] dataset (Brown et al., 2020; Radford et al., 2019).

The "Common Crawl" dataset is composed of content that has been scraped from the internet over a period of more than ten years. The dataset is known to have problematic content due to its aggregation of data from various publicly available sources, including Twitter, Reddit, and private blogs. This content includes hate speech, mentions of violent acts towards ethnic groups, sexually explicit material, sexual violence, and other forms of hate speech (Luccioni and Viviano, 2021).

Several biases have also been identified in Wikipedia (Hube, 2017), from which many BERT models derive its training data from. Though Hube's research revolve around languages other than Norwegian, we can not conclude that these biases are absent in the Norwegian Wikipedia data.

According to Bender et al. (2021), simply having more data does not necessarily mean that the data is diverse or free from unintentional biases. Instead, the concern is that large uncurated datasets containing dominant views may lead to the marginalization of underrepresented populations. Training models on trained on loosely filtered data could result in models with encoded stereotypical, derogatory associations along gender, race, ethnicity, gender occupation, and disability status (Bender et al., 2021; Touileb et al., 2022).

It is important to be aware of possible biases when working with Large Language models, as it is possible that the pre-training can affect the performance in unexpected ways in downstream tasks.

[6] https://huggingface.co/gpt2
[7] https://platform.openai.com/docs/models/gpt-3
[8] https://commoncrawl.org/

Chapter 3

Datasets

3.1 NorNE

As previously stated in the Chapter 2, NorNE[1] (Jørgensen et al., 2020) is the only publicly available named entity dataset for Norwegian. The dataset consists of a split between Norwegian Bokmål (nob) and Nynorsk (nno), whereas the Bokmål version is just slightly larger. The text is gathered from various domains, mostly news text, but also blogs, parliamentary proceedings and reports.

NorNE uses the IOB2 named entity annotations scheme which was added on top of the existing Norwegian Dependency Treebank corpus Solberg et al. (2014).

NorNE follows a 80/10/10 distribution between training, validation, and test datasets. The combined token count of both the Bokmål and Nynorsk splits adds up to approximately 600 000. There are nine types of annotated entities which are the following:

- `PER`: A persons name. Could be full name, partial or nicknames.
- `ORG`: Organizational names such as firms, institutions, political parties etc.
- `GPE_LOC`: Geo-political-entity location in a locative meaning. For example "A person lives in *Norway*"
- `GPE_ORG`: Geo-political-entity location in a organizational meaning. For example "Norway declined further *discussions with* Sweden". Both Norway and Sweden would be categorised as `GPE_ORG`.
- `PROD`: Named entities that are artificially produced. This is a broad category, and may include abstract entities, such as speeches, TV-shows, laws, ideas etc.
- `LOC`: Geographical places, buildings and facilities.
- `EVT`: Events or happenings. Can be festivals, cultural events, wars, etc.

[1] https://github.com/ltgoslo/norne

Type	Train	Dev	Test	Total	Proportion (%)	Avg. span
PER	8320	1092	961	10373	36,52	1.58
ORG	5601	688	521	6810	23,97	1.38
GPE_LOC	4222	454	428	5104	17,97	1.08
PROD	1404	248	131	1783	6,28	2.01
LOC	1511	194	185	1890	6,65	1.30
GPE_ORG	756	121	61	938	3,30	1.09
DRV	969	129	78	1176	4,14	1.19
EVT	274	16	14	304	1,07	1.55
MISC	14	0	14	28	0,10	1.25

Table 3.1: NorNE named entity distribution across the train, development, test splits and the average word span of a named entity. Avg. span notes to the average number of tokens a named entity is is composed of.

- `DRV`: Words or phrases derived from a name, but not considered names themselves. These terms usually include a full name and are capitalized, yet they are not classified as proper nouns. “Examples (fictive) are "Leeds-treneren" ("the Leeds coach") or "Bergen-mannen" ("the man from Bergen").”

- `MISC`: Named entities that do no belong in the aforementioned categories.

As discussed in Section 2.1, regarding named entity label encoding schemes, for IOB2 which will be used in this thesis, each entity type has a *beginning* and *inside* tag modifier. For example, for the named entity type `PER` (person), the first name would be annotated as `B-PER` and surname `I-PER`.

Table 3.1 displays the respective count for each entity type for each of the different datasets, the total amount of entities of each type, and the average token span of each named entity.

Furthermore, Table 3.2 refers to the total number of tokens, both non-named entity tokens and named entity tokens, regardless of named entity span. Meaning that if a named entity has for example a `B-PER` and `I-PER`, these would count as two named entity tokens.

3.2 Norwegian Parliamentary Speech Corpus

Norwegian Parliamentary Speech Corpus[2] (NPSC) is developed and distributed by the Norwegian Language bank at the National Library of Norway. The corpus made up of about 140 hours of recorded speech from the Norwegian

[2] https://www.nb.no/sprakbanken/en/resource-catalogue/oai-nb-no-sbr-58/

	Train	Dev	Test	Total
Non_NE_token	456 592	63 491	51 271	571 354
Named Entities	23 071	2942	2393	28 406
All tokens	479 663	66 433	53 664	599 760
Avg. sentence length	21.55	20.3	20.63	21.33

Table 3.2: NorNE total word count, named entity count, and average sentence length.

parliament (*Stortinget*), transcribed using Google Cloud Speech-to-Text (Solberg and Ortiz, 2022). After the automated transcriptions, the output of the Google Cloud Speech-to-Text was reviewed and corrected by transcribers who would listen to the audio, and compare it to the Speech-to-Text output (Solberg and Ortiz, 2022). The NPSC was predominantly developed to be an additional dataset for Norwegian automatic speech recognition.

The NSPC has both of the Norwegian official written forms included, Bokmål and Nynorsk. The most frequently used written form, Bokmål has a 87.2% distribution, whilst Nynorsk make up 12.8% of the corpus.

The NSPC transcriptions come in different formats, including normalized and non-normalized versions. In non-normalized transcriptions, numbers, dates, and years are spelled out using letters, and abbreviations are avoided. On the other hand, normalized transcriptions utilize common abbreviations and present numbers, dates, and years using digits in a standardized format.

The non-normalized and normalized transcriptions can be further categorized into sentence-segmented, word-level, and normalized sentence-segmented machine-translated versions. In the word-level transcriptions, additional metadata is organized into distinct fields for each individual token.

In the experiments discussed later in this thesis, the normalized word-token transcriptions will be utilized. This choice is made because it closely mirrors the training and testing process employed for our BERT models on the NorNE corpus. Sentences in this format consist of predefined token splits, eliminating the need for a BERT-tokenizer or alternative tokenization methods to separate tokens within non-tokenized sentences.

An additional advantage of using the word-tokenized version of NSPC is the inclusion of metadata known as "special_status" for each token. This metadata indicates the presence of non-word tokens within a sentence. Special statuses

Total word count	Avg. sentence length	Total capitalized words
1 056 038	18.02	41 247

Table 3.3: NPSC total word count, average sentence length, and total capitalized words.

Type	Total	Proportion (%)	Avg. span
PER	63	29.17	1.47
ORG	88	40.74	1.14
GPE_LOC	34	15.74	1.21
PROD	5	2.31	1.0
LOC	2	0.93	1.0
GPE_ORG	20	9.26	1.0
DRV	3	1.39	1.0
EVT	0	0	0
MISC	1	0.46	1.0

Table 3.4: The annotated NPSC sample dataset's named entity count, distribution of type, and the average word span of a named entity. Avg. span notes to the average number of tokens a named entity is is composed of.

may denote hesitations, represented by tokens like "<eee>". In other instances, tokens might have a special status for interruptions, where a speaker is interrupted mid-word, resulting in a partially transcribed word.

These tokens with special statuses will be excluded in the thesis' experiments, as well as in the statistics in Table 3.3.

3.3 Annotated NPSC sample

Total Non_NE_token	5222
Total Named Entities	216
Combined total tokens	5438
Avg. sentence length	18.13

Table 3.5: The annotated NPSC sample dataset's total count of non-named entities, named entities, combined count and average sentence length.

Since the NPSC does not have any gold standard in terms of named entity types, an effort was made to manually annotate some of the data to have NER results for NorNE and the NPSC that could be compared to one another, using the IOB2 tag scheme. Having annotated named entity data from the NPSC will provide insight on whether the models are good at generalizing across multiple datasets.

The dataset contains 300 pseudo randomly selected sentences, selected byletting a random algorithm generate 300 unique numbers between zero and the number of sentences contained in the combined sentence count of the full NPSC.

Type	NorNE prop. (%)	NSPC prop. (%)
PER	39.40	29.17
ORG	22.05	40.74
GPE_LOC	18.11	15.74
PROD	5.54	2.31
LOC	7.83	0.93
GPE_ORG	2.58	9.26
DRV	3.30	1.39
EVT	0.59	0
MISC	0.59	0.46

Table 3.6: Distribution of the named entity types, comparing the NorNE test dataset and he annotated NPSC sample dataset. Prop. denotes proportion.

These number would be the indices of which sentences that were selected from the NPSC, thereafter manually annotated. If a sentence was hard to interpret, which would occasionally happen stemming from the lack of context, a new random number would be generated and subsequently a new sentence is picked.

This manually annotated NPSC sample dataset, is about 10% of the total size compared to the NorNE test dataset. Due to a large variance in size, some differences must be highlighted:

First, there is significant proportinal variance between class label count, between the NPSC and NorNE. For example if we compare the named entity type count in Table 3.6, we can see the greatest variance between the `PER` and `ORG` proportion. Furthermore, for the sample dataset, the `LOC` class has six times less entries in proportion to the rest of the classes, compared to NorNE, and the `EVT` class does not appear at all in the annotated NPSC dataset.

Secondly, having such as small dataset, other data statistics that should be highlighted is that, the most infrequent classes that appear in the annotated NPSC sample has an average token span of only single token. Having a token span 1 does not necessarily mean that a prediction is would be easier, or harder for a model. This would depending heavily on what kind of data a model was trained on. Therefore, it is plausible that the NorNE test dataset results will be better than the annotated NPSC, as the NorNE test dataset is more similar to the NorNE training dataset than the NPSC.

Chapter 4

Experimental Setup and Model Training

In this chapter, we present a comprehensive overview of the experimental setup, including hardware and software environments, model specifications, model hyperparameters, and performance metrics gathered during the training process. To reiterate, our task is to train a models that will be able to identify, and predict named entity types where all text data is lowercase. By providing a detailed account of the models and frameworks deployment, we aim to facilitate the reproducibility of our findings and to strengthen the validity of our results.

Section 4.1-4.6 of this chapter lists the details of the experimental setup details, and describe the models that are used in our experiments.

Section 4.7 presents and discusses metrics that were gathered during the model training process for the NorBERT and the Ner-Scandi models.

4.1 Hardware and software environment

The experiments were performed on Python 3.9.13 version, PyTorch 1.12.1+cu113, on a Windows 11 desktop computer. The hardware specifications include a 5800X AMD Ryzen CPU, 32GB RAM, a M.2 SSD, and a NVIDIA GeForce RTX 3090 GPU.

4.2 Model Setup

Our models were trained using supervised machine learning methods. The NorNE dataset (Jørgensen et al., 2020) was used as the training material since the dataset includes IOB2 annotations for every token. For supervised machine learning, having annotated data is crucial for learning.

Two baseline models were chosen, the NCRF++ framework (Yang and Zhang, 2018), and SpaCy's NER Norwegian Pipeline[1]. Because of SpaCy's pipeline implementation (using a pre-existing model), this model does not require further

[1] https://spacy.io/models/nb#nb_core_news_sm

training. On the otherhand, NCRF++ does need training to function as we intend to use it.

Five different BERT models were chosen as our main models, NorBERT (Kutuzov et al., 2021), NorBERT2[2], mBERT (Pires et al., 2019), NB-BERT (Kummervold et al., 2021), and lastly nbailab-base-ner-scandi[3], which we will refer to as Ner-Scandi for the rest of this thesis. Ner-Scandi is a pre-existing fine-tuned version of NB-BERT model specifically intended for NER.

These five BERT models were further fine-tuned for our task in lowercase NER. Ner-Scandi is already fine-tuned for NER, but only for four different named entity types. Consequently, Ner-Scandi required additional fine-tuning to be able to capture more named entity types, but also fine-tuning for lowercase text data.

4.3 Pre-processing of datasets

For supervised machine learning, when training a model for NER, it is common to use datasets that are designed for NER, which contain prerequisite properties such as labels for each token. In our case, we want the models to be able to identify named entities when a word is not properly capitalized, therefore we had to do some light pre-processing. The pre-processing step involved converting all words, including proper nouns, into lowercase in all of the datasets.

Unlike our training dataset, the Norwegian Parliamentary Speech Corpus (NPSC) is not richly annotated with lexical features or named entity labels for each token. Upon manual inspection of the NPSC, it was apparent that the corpus does not capitalize the first word in a sentence unless it is a named entity. Although the entire corpus has not been inspected, it is reasonable to assume that the lowercase consistency holds true, unless there are transcription errors.

The decision was made to automatically annotate all tokens with either a 0 or 1, using a simple Python script. 0 denoting that a token does not have a capital letter word associated with it, and 1 denoting that a token was capitalized. These labels would serve as our gold standard when testing. After the gold standard was acquired, the corpus was made lowercase which would serve as the models' test dataset.

Having all text lowercase is of course not a perfect scenario for predicting named entities in text, since conventionally most written western language, every sentence starts with a capitalized word. However, as mentioned the NPSC has every first tokens in a sentence lowercase, unless it is a named entity.

An additional pre-processing step was done to the NPSC, removing certain tokens. Since the NPSC has been developed for use in speech recognition, some additional special tokens is included there. Tokens such as <ee>, <qq> and

[2] https://huggingface.co/ltg/norbert2

[3] Huggingface nbailab-base-ner-scandi: https://huggingface.co/saattrupdan/nbailab-base-ner-scandi

similar, indicate different special statuses of the current position in a sentence. These tokens indicate if a speaker was interrupted, had vocalic hesitations, or whether a word as inaudible or overlapping with someone else speaking. As these tokens do not serve any purpose for our experiments, to avoid the possibility that the models would be confused by their presence, all of these special tokens were removed before testing the models.

4.4 Baseline models

As previously stated, we make use of two baseline models, the NCRF++ framework and a SpaCy Norwegian pipeline. Under normal capitalization test datasets, both models have been reported to perform pretty good, reaching a F1 of about 85 for SpaCy[4] and about 88 for NCRF++ (Jørgensen et al., 2020).

4.4.1 SpaCy Norwegian Pipeline

We chose a CPU based SpaCy Norwegian pipeline for the initial benchmarks, since it is easy to implement and test, and because SpaCy's own reported results for Norwegian NER are quite good. For the model we use, SpaCy reported a precision of 85.0 and recall value of 84.0, using their largest and best performing Norwegian model "nb_core_news_lg".

The pipeline makes use of several pre-trained SpaCy active components for its predictions: tok2vec[5], morphologizer[6], Dependency Parser [7], lemmatizer[8], Attribute Ruler[9], Entity Recognizer [10].

Due to how the SpaCy tokenizer functions, words may occasionally be divided into several tokens. This can cause discrepancies in sequence length between the prediction output and the gold standard, hindering a one-to-one comparison. If a predicted sequence differs in length from the gold standard, the sequences is dropped from the evaluation.

4.4.2 NCRF++

The NCRF++ was chosen as the second baseline, as it is relatively easy to setup. Compared to the SpaCy pipeline, it is, however, more time and resource consuming, with additional configuration, code tinkering and training.

Some pre-processing of the datasets was also needed before initiating training and testing, utilizing the NCRF++ framework. All data had to be converted into a

[4]SpaCy models and metrics: https://spacy.io/models/nb/#nb_core_news_sm-accuracy
[5]SpaCy Tok2Vec: https://spacy.io/api/tok2vec
[6]SpaCy Morphologizer: https://spacy.io/api/morphologizer
[7]SpaCy Dependency Parser: https://spacy.io/api/dependencyparser
[8]SpaCy Lemmatizer: https://spacy.io/api/lemmatizer
[9]SpaCy Attribute Ruler: https://spacy.io/api/attributeruler
[10]SpaCy Entity Recognizer: https://spacy.io/api/entityrecognizer

`.bmes` format, where every token and named entity label would be on a separate line. To indicate the beginning of a new sentence, there would be a line of white space.

Large word embedding dimensions may be better to capture nuances in language, therefore the NCRF++ model was configured with a pre-trained fastText Skipgram, Norwegian embedding[11], with word embedding dimension size of 600.

The NCRF++ model was configured as a 200 hidden dimension Bi-LSTM word sequence layer, 50 dimension character feature CNN layer, with a CRF inference layer. The model was trained for 50 epochs, and with a stochastic gradient descent optimiser. For all hyperparameters details, see Appendix A.

4.5 Main models

As stated in Section 4.2, five different pre-trained BERT models were chosen for the main experiments. Four of these models were trained on mostly Norwegian text material.

All the models utilize 12 layers, 12 attention heads, 768 hidden size, 0.1 dropout probability for both attention heads and layers, and a position embeddings size of 512 for its pre-training. The vocabulary size of each model differs. These are listed in the respective model detail subsections below.

Utilizing multiple models offers a wider basis for comparison when analyzing the results. Comparing the results against one another gives us an indication of which model that could be best for real life non-curated datasets, and automatically generated transcriptions.

Note that one of the models that are tested, the multilingual BERT (mBERT), is not pre-trained primarily on Norwegian data. Nevertheless, it has its relevancy: First, because its multilingual properties could potentially highlight pros and cons of using a non-language specific BERT model. Secondly, because one of our other primary models, NB-BERT is based on the same structure and inherits its initial training weights from mBERT. Comparing these two models could highlight some of the possible advantages of having a language specific pre-BERT, which has certain properties derived from mBERT.

4.5.1 NorBERT

NorBERT was trained from scratch for Norwegian by the Language Technology Group[12] at the University of Oslo. The model's vocabulary size is 30 000, predominantly consisting of Norwegian words. About 90% of the training data

[11] Embedings ID 132 http://vectors.nlpl.eu/repository/#

[12] https://www.mn.uio.no/ifi/english/research/groups/ltg/

is composed of the Norsk Aviskorpus (NAK)[13], a collection of both Bokmål and Nynorsk news text. The rest of the dataset is text extracted from Bokmål and Nynorsk Wikipedia articles. In total, NorBERT was trained on combined token count of about 1900 million (Kutuzov et al., 2021).

For our fine-tuning and the experiments, the v1.1 (February 13, 2021) NorBERT base model was used.

4.5.2 NorBERT2

NorBERT Base 2.0 (NorBERT2)[14] is a continuation of NorBERT, with a significantly larger corpus, and a larger vocabulary of 50 000. The total token count is about 15 billion, composed by the C4[15] and the NCC[16] corpus. Futhermore, NorBERT2 use Whole Word Masking techniques during training, meaning that all subwords corresponding to a word are all be masked at the same time.

For our fine-tuning and experiments the Cased Norwegian BERT Base 2.0 (NorBERT2, February 7. 2022) was used.

4.5.3 mBERT

Multilingual-BERT, also known as mBERT[17] is a pre-trained model that was trained on 104 languages, where the training data was derived from Wikipedia text data from the different languages used. The proportion of its vocabulary from each language vary, and the total size of the vocabulary is 110 000.

While a fine-tuned mBERT is not necessarily the best performer in various NLP tasks compared to language specific BERT models, mBERT can still produce respectable results. As reported in Section 2.7, Kummervold et al. (2021) achieved a F1 score of 78.8 in Bokmål, and 81.7 in Nynorsk, on the NorNE test dataaset using a fine-tuned mBERT model.

For our fine-tuning and experiments the lastest version BERT Base Multilingual Cased version was used (June 18, 2019).

4.5.4 NB-BERT

Trained by a Norwegian governmental entity, The National Library of Norway (NLN), NB-BERT, has a total token count of 18,4 billion (Kummervold et al., 2021).

[13] https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/
[14] https://huggingface.co/ltgoslo/norbert2
[15] https://www.tensorflow.org/datasets/catalog/c4
[16] https://huggingface.co/datasets/NbAiLab/NCC
[17] https://huggingface.co/bert-base-multilingual-cased

NB-BERT differentiates itself from NorBERT and NorBERT2 largely by the size of the training corpus, using the Colossal Norwegian Corpus[18], and by the large vocabulary size of 119,547 words. Nearly 12 billion of the 18.4 billion total tokens are from books, from year 1814 up until 2020, which results in a rich variety of Norwegian written grammatical and linguistic text structure throughout this period.

Also, NB-BERT initiated model weights are derived from mBERT's weights. Kummervold et al. (2021) assumed that this increased NB-BERT's performance compared to starting with random initiated weights. Kummervold et al. (2021) also theorize that because of initiated weights derived from mBERT, NB-BERT might be better prepared when dealing with previously unseen words, and texts that incorporate bits of different languages.

For our fine-tuning and the experiments the NB-BERT base 1.1 release (March 11, 2021) was used.

4.5.5 Ner-Scandi

The Ner-Scandi is a version of NB-BERT$_{\text{BASE}}$, fine-tuned to do NER in Norwegian (Bokmål and Nynorsk), Swedish, Danish, Icelandic, and Faroese. It is fine-tuned to predict four named entity types, `PER`, `LOC`, `ORG`, and `MISC`, but for our purpose, we fine-tune Ner-Scandi further so it include all the entity types that NorNE[19] contain.

The training coropora used for the pre-existing fine-tuned Ner-Scandi are soured from multiple datasets. This includes NorNE (Jørgensen et al., 2020), DaNE (Hvingelby et al., 2020), SUC[20] 3.0 and the Icelandic and Faroese parts of the WikiANN[21] dataset.

For our fine-tuning and the experiments we used the latest Ner-Scandi model, released September 25. 2021.

4.6 BERT training

In this section we aim to document all relevant details that could potentially impact our BERT models' performance to ensure that the training can be replicated and that our results can be reproduced.

The documentation includes the dataset used for training, fixed models seeds, the activation function, loss function, regularization, optimiser, number of epochs and scheduler. All of which could have a performance impact on the results when training a given model.

[18] https://huggingface.co/datasets/NbAiLab/NCC
[19] NorNE entity types: `ORG`, `LOC`, `GPE_LOC`, `GPE_ORG`, `PER`, `PROD`, `EVT`, `DRV`, `MISC`
[20] https://spraakbanken.gu.se/en/resources/suc3
[21] https://huggingface.co/datasets/wikiann

Note that to have entirely reproducible results is difficult when working with machine learning models. There are several factors that could have an impact, most are related to that many experiment's details are not properly documented (Schelter et al., 2015). Even software library versions could have an impact on a model's performance (Beam et al., 2020; Li and Talwalkar, 2020).

Training datasets

For the model training the NorNE dataset was used. Both Nynorsk and Bokmål training dataset splits were combined into one big training corpus. At the beginning of each training epoch, the combined corpus got shuffled so there would be a good mix of both Nynorsk and Bokmål for each training batch, instead of large chunks of only one or the other language version.

Fixed seeds

We used five manually picked seeds for training, which results in five different trained models for each of the five different BERT variants. The seeds 8, 37, 42, 101, 1024 were selected. The seeds were used to initialize deterministic CUDA behaviour[22] to ensure that the code would produce similar results even across different graphics processing units (GPUs). For details how this was implemented, the reader is referred to the project's GitHub repository[23].

Activation function and Pooling

In the experiments, a pooler was used to obtain an output from the BERT model. A pooler technique involves that the last hidden layer state is a tensor, which is made up of hidden states for each input token's vector representation. Subsequently, these vectors are inputs for a linear layer that classifies the input tokens using a linear activation function.

Loss function

The Cross Entropy Loss function from the PyTorch library was used in all BERT model for each training batch, the loss value was used to calculate the back-propagation.

Optimiser

The gradient-based Adam algorithm was used as the model optimiser, with an initial learning rate of $2e-5$.

[22] Reproducibility for PyTorch: https://pytorch.org/docs/stable/notes/randomness.html
[23] https://github.com/sfimediafutures/MA_Peter-R-ysland-Aarnes

Number of epochs and patience

50 epochs was set as the maximum number of training iterations for a given seed. To reduce time spent training a model, a patience counter of 10 was set. A patience counter in this context is how many training epochs a model was allowed to do *without* improving its recall value on the validation set, without the training is terminated before the maximum number of epochs is reached. The assessment of performance during training is measured by a model's recall value.

In practise this could result in that a model stops training after 15 epochs if the model did not improve after epoch number 5.

Scheduler

Instead of having a fixed learning rate throughout the training epochs, the learning rate scheduler ReduceLROnPlateau (Reduce learning rate on plateau) on used in combination with the Adam optimiser (Kingma and Ba, 2014). The benefit of using a scheduler, is that if a model's learning stagnates after iteratively training, the learning rate can adjust dynamically. The ReduceLROnPlateau[24] will reduce its learning rate if the model does not decrease its development loss value in the given epochs, compared to earlier epochs with the same learning rate.

Similar to the patience counter early training termination, the ReduceLR-OnPlateau also uses a patience counter for when to reduce the learning rate. This patience counter is set to 1. In practise this means that the models will have two attempts at improving its loss rate before the learning rate will be reduced. The new learning rate will be a product of $current\_learn\_rate \times factor$. After some preliminary testing we decided to use the a factor of 0.93. This number was chosen since the initial learning rate of $2e-5$ is already quite low, and therefore having a factor of 0.93 would not shift the learning rate too much. This could of course lead to little to no change in the subsequent epoch, however, in combination of a low patience threshold and the small adjustments in learning rate, the learning rate could drop quite fast.

Because we use a total of five different BERT models, and each model having five seeds, we have a total of 25 trained models. The particular parameters discussed here were chosen to maximize the training effectiveness, that is, balance the amount of time spent with the training yield.

4.7 Training metrics

This section present metrics gathered during the training of two BERT models, NorBERT and Ner-Scandi. These metrics include recall values from the validation

[24] Scheduler ReduceLROnPlateau: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

dataset, adaptive learning rate, training loss, and validation loss for each epoch during training and for each of the different seeds used.

NorBERT and Ner-Scandi are highlighted in the following subsections, since they vary significantly in the number of epochs they were permitted to train until the patience counter of 10 stopped, resulting in a early terminating model training on a particular seed.

Even though the number of epochs for each model and seeds vary greatly, the two models' metrics follow similar trends. These trends also apply for the three remaining models. For a tabular view of all the model training metrics, see Appendix A.

4.7.1 Validation Recall and Learning Rate

For the NorNE validation dataset, the total number of named entities 2 942 as shown in Table 3.2. However, total count of named entity tokens, not concatenating the token span of a named entity, is 4 128.

In our training of the different BERT models, the validation recall rate is defined as the number of correctly labeled named entity tokens (disregarding token boundary) divided by the total number of labeled entities. Words with the "O", the non-named entity class, is excluded from the validation recall rate calculation.

Using recall is beneficial in this scenario, where one of our aims is to detect named entities regardless of class or token span. By focusing on recall, the model's performance is optimised towards detecting as many named entities as possible, maximizing the number of correct named entity identifications without emphasis on minimizing false positives or considering label type.

Focusing on F1 score and precision is less suitable in our experiments, as it does not prioritize the detection of named entities. F1 score balances precision and recall, while precision measures the accuracy of positive predictions. Using precision could encourage overly conservative predictions, which may leading to sub-optimal performance in entity detection. Therefore, recall aligns better with the objective of detecting named entities regardless of class.

NorBERT

After the first epoch for all NorBERT model seeds, there are dramatic spikes in the improved recall by 8-10% before the recall fluctuate in the remaining epochs, as shown in Figure 4.1.

Figure 4.2 shows that the model would typically settle for a new learning rate about every other epoch. As the learning rate value decreases in an approximate linear fashion, the validation recall sometimes get a small increase, thus resetting the patience counter.

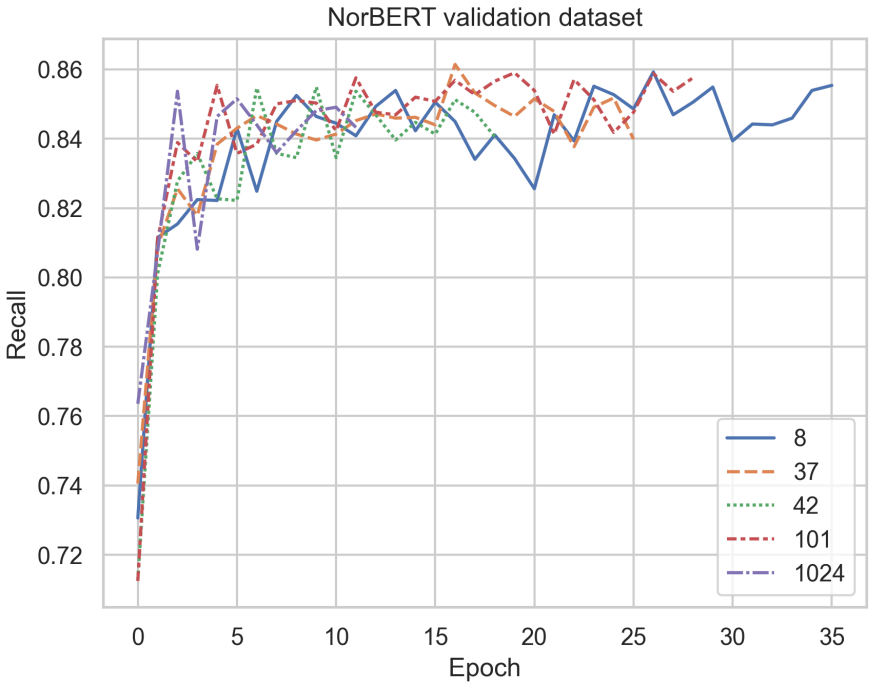


Figure 4.1: NorBERT's recall rate for each training epoch. The colored lines represent different training runs on a given seed. With patience counter of 10, training is terminated if a model does not improve on that seed after 10 epochs.

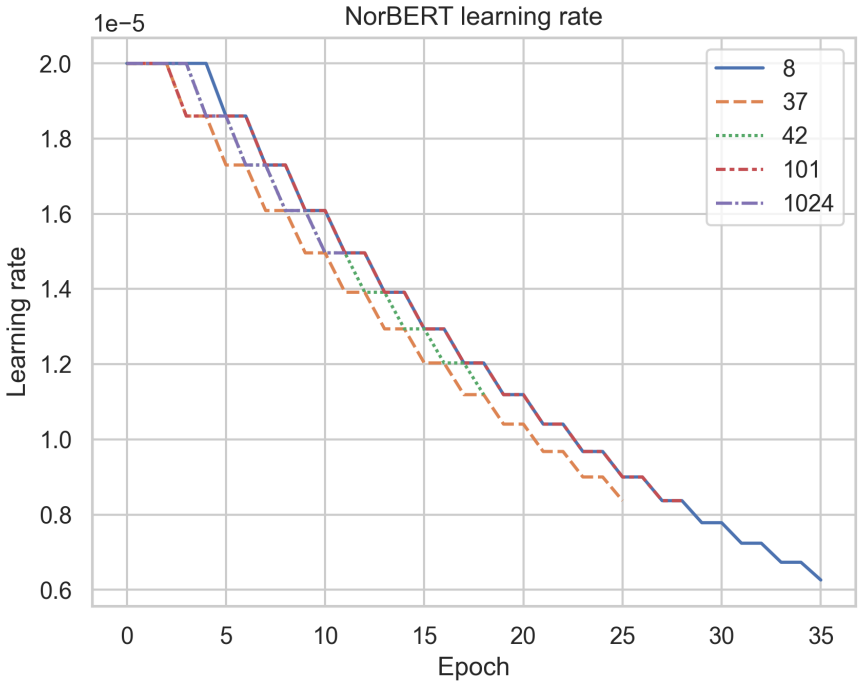


Figure 4.2: NorBERT's learning rate for each training epoch. The colored lines represent different training runs on a given seed. The model had two attempts to improve its performance before its learning rate was decreased.

On average NorBERT had the highest total number of training epochs during training compared to the other BERT models, yet for each seed, the numbers of epochs varied greatly. This was particularly evident when comparing seed 8, which stopped at epoch 35, to seed 1024, which stopped at epoch 11. Despite the significant differences in the number of training epochs, the highest validation recall for both seeds was almost identical. For the exact recall values of each seed, please refer to the Appendix A.

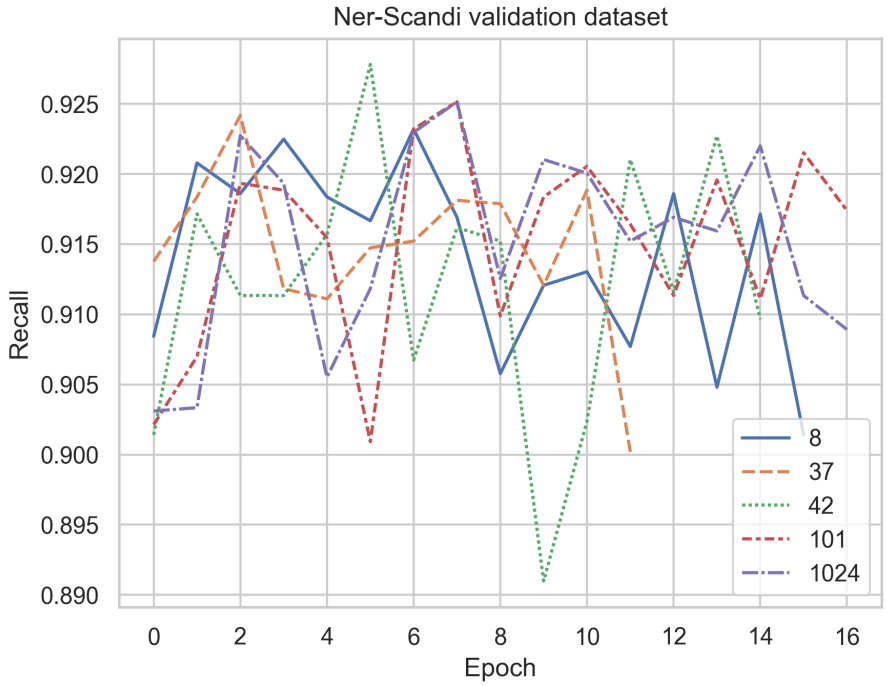


Figure 4.3: Ner-Scandi's recall rate for each training epoch. The colored lines represent different training runs on a given seed. Given the patience counter of 10, training is terminated if a model does not improve on that seed after 10 epochs.

Ner-Scandi

Ner-Scandi does not only differ from NorBERT in the improved validation set recall, but also in the average epochs spent on a given seed. NorBERT spent on average 24.4 epochs training across the five seeds, whereas Ner-Scandi spent on average 15.4.

It is possible that Ner-Scandi needed less time training to reach its optimal performance since this model is a pre-exsisting fine-tuned NER model. However, the Ner-Scandi is derived from NB-BERT, which spent 18.4 epochs training, only 3 epochs more than Ner-Scandi. This suggests that NB-BERT requires less fine-tuning on average, at least for our experiments. It is inconclusive if this is the case for other NLP tasks.

As early as after one training iteration, Ner-Scandi achieved high recall values. For the remaining epochs before stopping, it would fluctuate about 2%, similar to

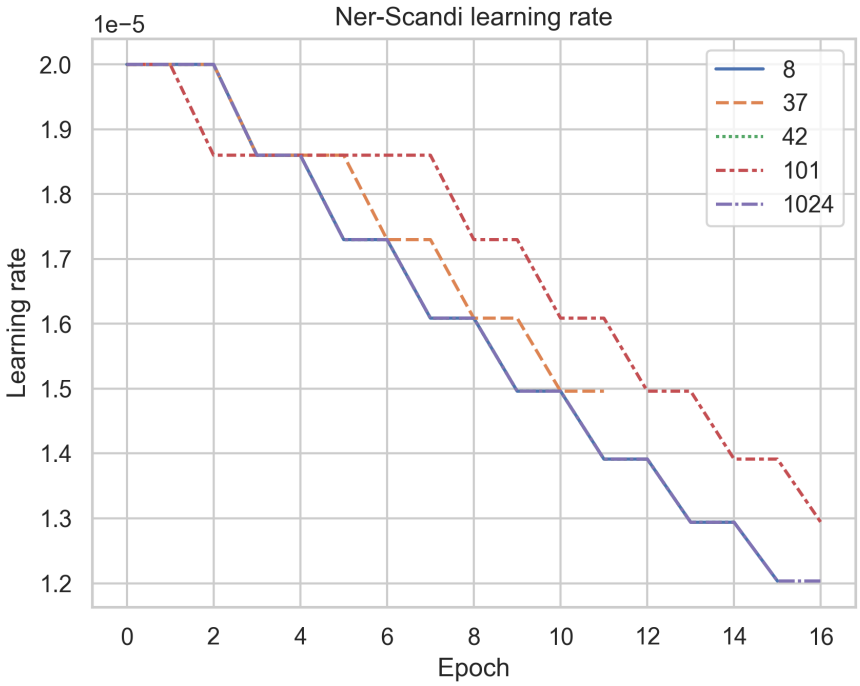


Figure 4.4: Ner-Scandis's learning rate each training epoch. The colored lines represent different training runs on a given seed. The model had two attempts to improve its performance before its learning rate was decreased.

the other models. The adaptive learning rate also decreases, very much like the other models during training, though ending at a higher learning rate value than most other models because of early training termination.

4.7.2 Training loss and validation loss

Ideally we would want the training dataset loss and the validation dataset loss to be as close in value to each other as possible. In a combined figure, illustrating training and validation loss rate, we would want to see the graphs converge.

As long as the training data stays the same for each epoch, typically the loss rate decreases exponentially over the training duration as the model performance increase. Similarly, we want the validation loss to decrease as well. If there are large discrepancies between the training and validation loss, it could indicate that the models have a hard time applying what they have learned from the training data onto the validation dataset. If the validation loss goes far beyond the training loss value, it would indicate that the modesl are overfitting to the training data which will affect its ability to generalize across different datasets.

From figure 4.5-4.8 and from our validation recall data, we can discern how well the training advanced.

In our use, if the graphs were combined, training and validation loss would not converge, since the validation loss was typically a bit higher than the training

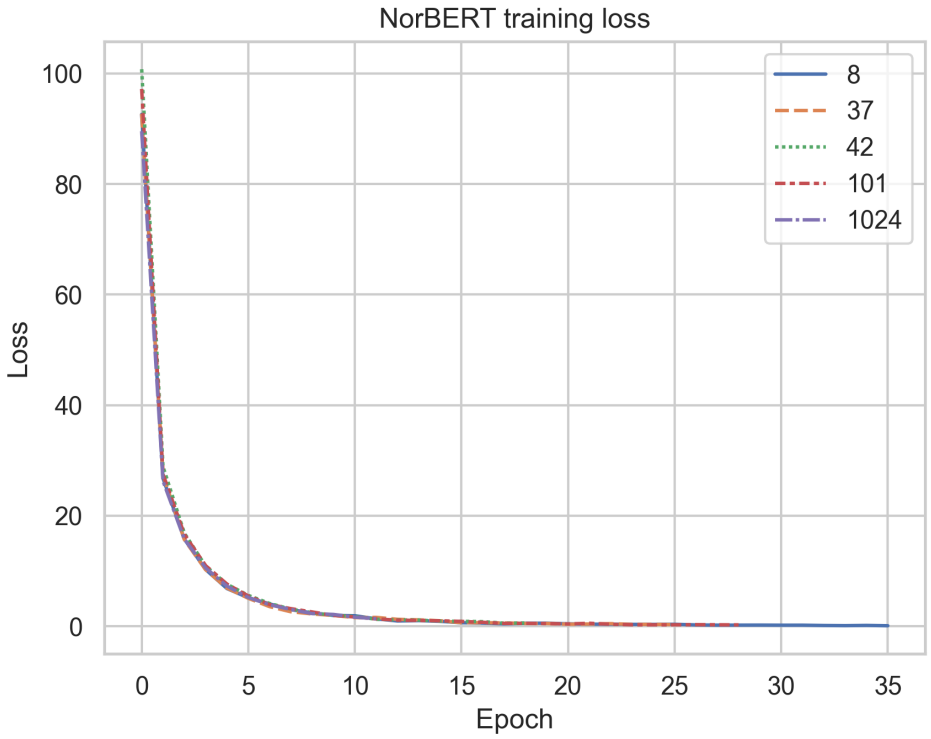


Figure 4.5: NorBERT's training dataset loss rate for each seed. Each colored line represent a model seed. All seeds follow the same trend, where the loss rate gets closer to zero for each training epoch. For full numeric details, refer to Appendix A

loss. Frequently, after epoch 2-5, the validation loss would incrementally increase by a small amount, yet, the validation dataset recall value would sometimes also increase. The observations of validation loss increase, and validation set recall increases may seem counterintuitive. This can, however, be partially understood by how Cross Entropy Loss operates. In simple terms, the Cross Entropy Loss measures the confidence of a prediction. The models may be "less confident" in its prediction using the Cross Entropy Loss calculation, thereby increasing the validation loss. The recall may increase even though model is less confident of whether the prediction is correct.

Note that some of the labels are quite similar in how they appear in the text. For example `GPE_LOC` and `GPE_ORG` get used almost interchangeably, and a simple suffix may be the only clue to differentiate how a word was labeled. This could be one of several causes to the models' less than confident predictions, although this cannot be concluded with certainly. To get a better understanding of why the loss value increase, one would need to analyse and compare individual vectors values throughout the training iterations, which is beyond the scope of this thesis.

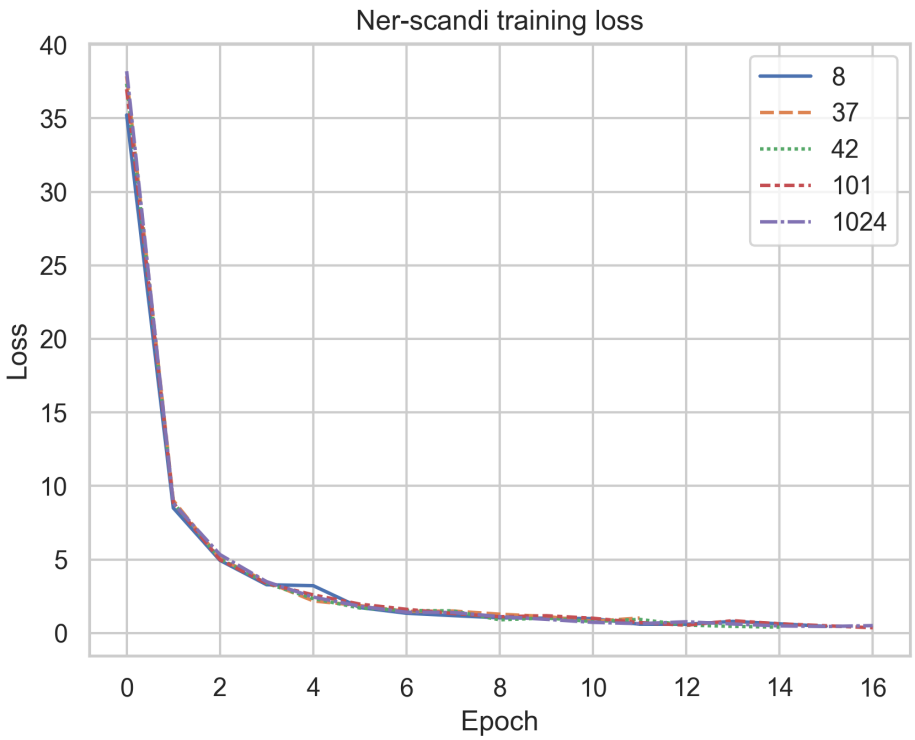


Figure 4.6: Ner-Scandi's training dataset loss rate for each seed. Each colored line represent a model seed. All seeds follow the same trend, where the loss rate gets closer to zero for each training epoch, except for epoch number 4 for seed 8, where there is a slight uptick in loss rate, before lowering the next epoch. For full numeric details, refer to Appendix A.

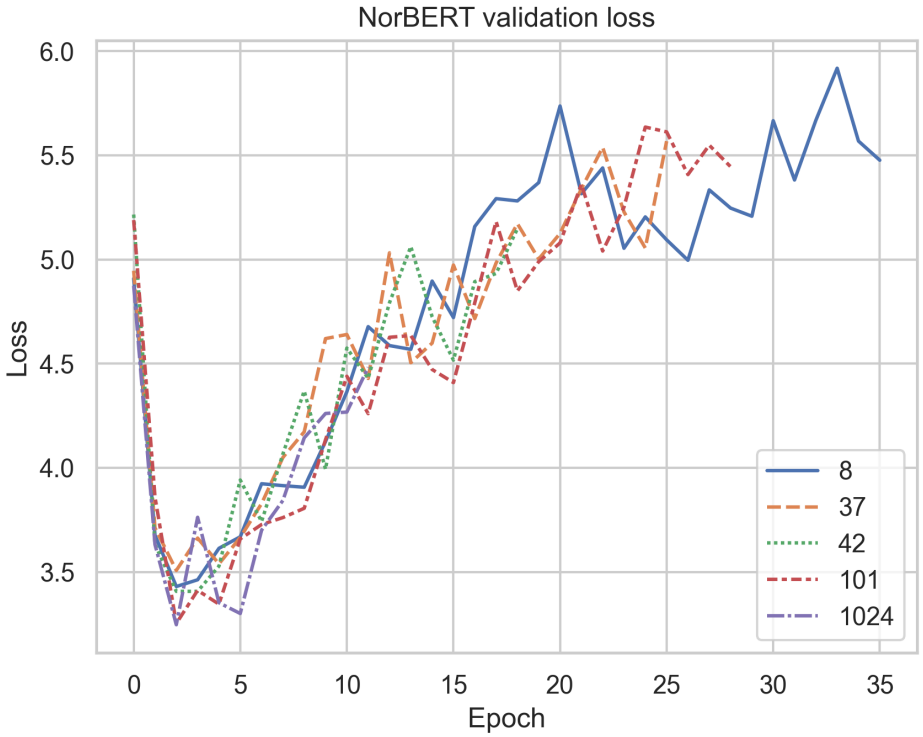


Figure 4.7: NorBERT's validation dataset loss rate for each seed. Each colored line represent a model seed. Loss rate steadily increase throughout the training.

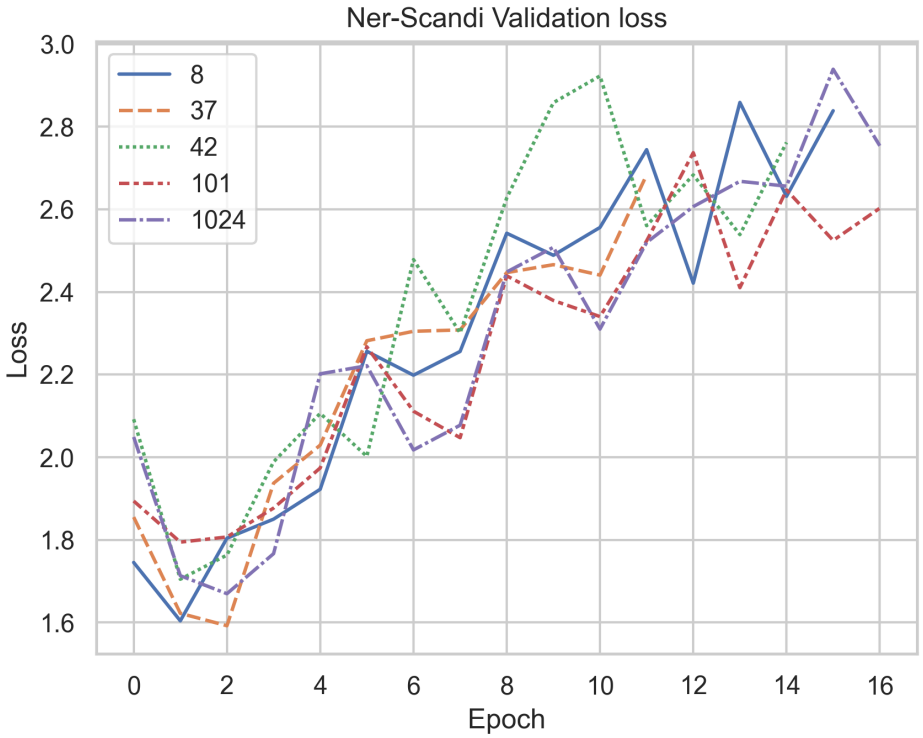


Figure 4.8: Ner-Scandi's validation dataset loss rate for each seed. Each colored line represent a model seed. Loss rate steadily increase throughout the training.

Chapter 5

Evaluation Methods

This chapter details the different evaluation tasks which serve as the methods to assess a models performance. The first section provides the specific details about the different methods, and what metrics they include. The second section specify which evaluation methods that are used for a given dataset. In the next chapter we present the results from our experiments, using the methods and datasets described in this chapter.

5.1 Methods

The different models are subject to three specific methods of evaluations, accompanied with precision, recall and F1 as metrics serving as quantitative performance measurements. These methods are a "Named Entity Recognition", "Strict Match", and "Named Entity Binary Classification ". Detail about the methods are given in the following subsections.

Additionally, confusion matrices are used for further analysis of named entity misclassifications. Since the BERT model results consists of multiple seeds per model, the results are presented as performance averages calculated for all seeds with the inclusion of the standard deviation, instead of assessing one particular seed's performance.

The assessed performance for the models have three parts because of the three datasets. First, an evaluation is done using the NorNE test dataset. The NorNE test dataset is a combination of both the Bokmål and Nynorsk test dataset. Secondly, an evaluation on of the IOB2 annotated NPSC sample dataset. Lastly, an evaluation is done on the full NPSC.

5.1.1 Named Entity Recognition

The first method we describe is the named entity recognition evaluation *without* token boundary considerations.

NER without token boundary considerations means that if a named entity has

a *beginning* and a *inside* tagged token, these will count as two separate tokens when calculating performance metrics. By definition, this would entail that for example that the named entity "Steve Jobs" would be split into two. "Steve" `B-PER`, and "Jobs" `I-PER` will be two separate instances of named entities and are also two separate types.

The metric used for model performance comparisons is the F1 micro-average. The reason for using a micro-average F1 is that NorNE having 18 different named entity classes when *beginning* and *inside* tags are included, leading to some classes being much more frequent than others. Micro-average will favor the importance of bigger classes, contrary to the macro-average that weighs all classes with equal importance, results that may be misleading. If all classes were weighted as equally important, results in classes such as `B-MISC`, which only has 14 entries, (less than 0.5% of all named entity tokens test data) would have a great impact on the F1.

After the different micro-averages for each model are presented, the best model is highlighted in further detail, including an overview of precision, recall and F1 for each class. Misclassifications analysis are presented and discussed as well, utilizing a confusion matrix.

5.1.2 Strict Match

The second evaluation metric for model comparisons use the micro-average F1 scores for how a model performs NER with a strict match evaluation criteria. As defined in Section 2.2.2, a strict match evaluation requires that the model predicts the correct named entity type and its token boundary. In practice, this means that a model needs to predict both `B-PER` and `I-PER` to have a correct prediction for a `PER` (person) named entity, if a full name spans two tokens. For example, for a correct prediction for the name "Steve Jobs", a `B-PER` must be followed by an `I-PER` prediction. If the name is only partially recognized, such as just having `B-PER` without being followed by the `I-PER`, the prediction will be evaluated as incorrect.

F1 micro-averages are used for the same reason here as for the named entity recognition evaluation without the strict match criteria. That is, due to the substantial imbalance in the number of instances for each class of named entity.

After assessing the best performing model is highlighted, with an overview of precision, recall, and F1 with for all named entity classes.

5.1.3 Named Entity Binary Classification

The third evaluation metric for model comparisons includes precision, recall, and F1 scores for all models at a binary classification level. This metric will assess how effectively a model can determine whether a token is a named entity or not. For this method, predicted named entity class and whether the named entity spans

multiple tokens are excluded for the evaluation. The focus is solely on recognizing whether a token could be a potential named entity. The significance of this binary classification metric is emphasised, as it acts as the primary evaluation measure to identifying improperly capitalized named entities.

These binary classification predictions are derived from the models NER prediction. If prediction outputs of a token is *any* named entity class, the prediction is set to "1", denoting a positive prediction for the named entity. Likewise, if the prediction output is a non-named entity, the output is set to "0".

Instead of using micro-average F1, the binary classification results are reported as macro-averages. There are vast differences between the number of non-named entities and those that are named entity tokens in the corpus. Since our goal is to recognize improperly capitalized named entities, however, it is equally important to consider false positives alongside true positives. If fine-tuned models such as these are used to correct capitalization mistakes in a production environment, it is important that they do not capitalize non-named entities.

In addition to the aforementioned metrics, further analysis is provided using a confusion matrix for the model that achieves the highest macro-average F1 score.

5.2 Evaluating Dataset Results

The following subsections provide brief descriptions of what model evaluation methods that are used on a given dataset. There is some variation in the methods used for each dataset, stemming from the fact that the full NPSC does not have a IOB2 gold standard.

5.2.1 Evaluating NorNE

For the NorNE dataset, the models are assessed using all model evaluation methods described in the previous section. To reiterate, this will be the named entity recognition, the strict match named entity recognition, and lastly the binary classification.

5.2.2 Evaluating the annotated NPSC sample

As described in Chapter 3, a sample of 300 random sentences from the NPSC was selected and annotated in the IOB2 format. This was done to gain insight into how the different BERT models perform in named entity recognition on the NPSC. Although the sample size is limited, encompassing only a fraction of the number of sentences of the full NPSC, evaluating models on this sample is important since it provide us with some understanding of how the models are able to generalize across various datasets.

The evaluation methods for the NPSC sample involves NER evaluating without the token span boundary criteria, and the strict match NER classification. For this sample dataset, binary classification are not highlighted in detail.

5.2.3 Evaluating the NPSC

For the full NPSC dataset, the models are only subjected to the named entity binary classification evaluation. There is no official gold standard for the NPSC to determine whether a word is a named entity or not, besides solely looking for capitalized words. Back in Section 4.3 we gave details on how we annotated the NPSC to create a gold standard intended for binary classification evaluation.

Chapter 6

Results and discussion

This chapter presents our main findings, and is divided into four sections. The first section presents the results from the NorNE test dataset. The second presents results from the annotated NPSC sample dataset and the full NPSC dataset. The third section compares the model performance on the NorNE and NPSC. Each of these sections are divided into subsections where we present and discuss the outcomes of the different evaluation methods. In the fourth section we discuss some possible factors that could limit the models' performances.

The findings from named entity recognition without token boundary is highlighted in more detail, compared to the strict match evaluation. We regard NER without token boundaries as more important than strict match in our experiments, primarily because the binary classification results are derived from this NER evaluation. For the binary classification evaluation, token span and boundary is not important the key aspect is recognizing that a token is any type of named entity.

6.1 NorNE results

Table 6.1 presents an overview of all models' F1 score metrics, corresponding to the three respective subtasks: NER classification without token boundary, NER with the strict match criteria, and the named entity binary classification.

Our baseline model, the trained NCRF++ performs the poorest among all the evaluations. The SpaCy pipeline model performs significantly better than the NCRF++ even though the SpaCy model was not fine-tuned to detect or predict lowercase named entities. In contrast, all of the fine-tuned BERT models outperform the baseline models in all model evaluations by a large margin.

For all the evaluation methods, the Ner-Scandi model achieved the greatest F1 scores. NB-BERT achieved comparable results to Ner-Scandi, most notably at the binary classification evaluation. The standard deviation[1] for these two models

[1]We will sometimes present the standard deviation for results in the text when it is notably high. The standard deviation will be denoted as SD.

Model	No Boundary	Strict Match	Binary
SpaCy*	50.87	45.89	80.17
NCRF++	42.33	32.20	73.66
mBERT	82.15 (±0.50)	79.97 (±0.41)	95.51 (±0.56)
NorBERT	83.12 (±1.10)	82.19 (±1.26)	96.32 (±0.10)
NorBERT2	85.45 (±0.23)	84.38 (±0.37)	95.32 (±0.08)
NB-BERT	87.00 (±0.30)	86.50 (±0.34)	97.06 (±0.09)
Ner-Scandi	**87.06** (±0.56)	**86.58 (±0.45)**	**97.07 (±0.13**)

*Table 6.1: NorNE results for all models. BERT models use the average F1 across all seeds for a given model. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary classification denotes the task where only the detection if an entity is present, without specific type associated. *Ten sentences were dropped from SpaCy pipeline*

indicate that the performance differences between them are insignificant.

The larger vocabulary language specific BERT models typically achieved greater F1 metrics. The exception is for the binary classification evaluation, where surprisingly, NorBERT2 produces worse results than the smaller model NorBERT, and the larger, but non-language specific, multilingual model mBERT.

6.1.1 NorNE Named Entity Recognition

This subsection, highlight the precision, recall and F1 for specific class results by Ner-Scandi, the model that performed the greatest for the NER without token boundary and span evaluation. The discussion highlights some of the possible strengths and weaknesses of the Ner-Scandi's predictions.

Table 6.2 present the the combined average result across all Ner-Scandi model seeds. These metrics include named entity classes performance prediction for both labels with the "`B-`" (beginning) and "`I-`" (inside) IOB2 labels.

The model achieved an overall micro-average precision value of 86.35, and with a slightly greater recall value of 87.79, resulting in a F1 of 87.06. Inspecting 6.2 in a descending order, we observe that the model performed best for tokens that are the most abundant. Most notably, `B-PER` and `I-PER` have excellent precision and recall values, both above 95%. Precision being particularly great, suggesting that the model rarely produce predictions that are false positives. `B-GPE_LOC` and `I-GPE_LOC` are also among the classes with the greatest scores, where the F1 exceeds 91.

Subpar performance is observed for the `B-MISC` predictions, exibiting a precision rate of 40.0, and `I-MISC` 20.0. The recall rate is considerably better, reaching 80.0 and 60.0 respectively, however, the standard deviation is high (`B-`

NE Type	Precision	Recall	F1	Support
B-PER	96.90 (±0.95)	95.11 (±0.68)	95.99 (±0.29)	961
I-PER	98.31 (±0.36)	95.37 (±0.79)	96.82 (±0.39)	510
B-ORG	82.26 (±0.65)	87.08 (±0.20)	84.60 (±0.42)	521
I-ORG	72.26 (±2.23)	92.05 (±1.53)	80.92 (±1.16)	248
B-GPE_LOC	93.46 (±0.57)	89.06 (±0.99)	91.20 (±0.37)	428
I-GPE_LOC	93.42 (±1.24)	90.49 (±2.41)	91.92 (±1.61)	79
B-LOC	78.92 (±3.28)	75.79 (±2.30)	77.28 (±1.99)	185
I-LOC	74.35 (±5.23)	86.39 (±2.92)	79.72 (±1.88)	85
B-PROD	60.61 (±3.33)	56.48 (±4.38)	58.31 (±2.65)	131
I-PROD	48.25 (±6.87)	63.32 (±4.58)	54.57 (±5.53)	126
B-DRV	79.49 (±2.69)	76.54 (±2.93)	77.90 (±1.27)	78
I-DRV	41.54 (±3.77)	45.42 (±8.55)	43.13 (±5.34)	13
B-GPE_ORG	64.26 (±1.91)	75.91 (±3.44)	69.51 (±0.89)	61
I-GPE_ORG	100.00 (±0.00)	71.12 (±14.91)	82.32 (±9.26)	7
B-EVT	68.57 (±14.00)	55.69 (±7.94)	60.89 (±8.13)	14
I-EVT	40.00 (±25.50)	25.90 (±18.22)	30.62 (±20.01)	4
B-MISC	40.00 (±25.79)	80.00 (±40.00)	51.73 (±29.77)	14
I-MISC	20.00 (±16.33)	60.00 (±48.99)	30.00 (±24.49)	3
Micro avg.	86.35 (±0.64)	87.79 (±0.49)	87.06 (±0.56)	3468

Table 6.2: Ner-Scandi NorNE test dataset results measured with precision, recall and F1 for NER without considering token span of a named entity. Support column denotes number of entries for each class.

MISC: $SD = 40.0$, I-MISC: $SD = 48.99$), suggesting that the results vary greatly across the different model seeds. The precision results suggest that the model tends to predict more MISC classes than necessary, resulting in an increase in false positives. The false positives indicate that the MISC classes is the hardest to predict correctly for the Ner-Scandi model, as well as the class providing least reliable performance due to the significant varying results across different seeds. Similarly, the EVT classes also suffer from low precision and an even lower recall. However, it is important to note that these classes are among the least frequent classes in the test dataset compared to other classes. Consequently, the metrics will change significantly based on whether a prediction is correct or incorrect.

By examining the NorNE dataset named entity distribution of the training data in Table 3.1, it is evident that the MISC class is by far the least frequent class with only 14 entries. The MISC class constitutes less than 0.5% of all named entities in the NorNE dataset. Considering the scarcity of data for this class, it could be that the model lacks sufficient training data to effectively learn and make accurate predictions for this particular class.

Again, upon inspection of the NorNE data statistics in Table 3.1, it is evident that the EVT class is more frequent in the training data than the MISC class. One might expect the model to produce better predictions for the EVT classes due to the larger amount of available training data. Nevertheless, the EVT classes, both B-EVT and I-EVT, only exhibit slightly better predictions than the MISC classes. The lower standard deviation is, however, lower which indicates better performance consistency for the EVT classes, across the multiple seeds.

6.1.2 NorNE Strict Match

Table 6.3 provides an overview of how well Ner-Scandi perform for the strict match evaluation, an evaluation where the token boundary applies. Logically, if the model performed good for a class with both the B-tag and I-tag in the previous evaluation, the model is expected to have similar performance for the strict match evaluation.

Among all Ner-Scandi model's seeds, the micro-average precision is 86.90, and a micro-average recall rate of 86.26. The average F1 therefore results in a score of 86.58.

As the results in Table 6.3 indicate, the classes that are more frequent generally achieves higher values for both precision and recall, in line with the argument above. Conversely, the less frequent classes are harder to predict, most notably the MISC class and EVT, with a F1 score of 46.66 ($SD = 25.07$) and 57.12 ($SD = 6.40$) respectively.

NE Type	Precision	Recall	F1	Support
PER	96.75 (±0.86)	94.97 (±0.76)	95.85 (±0.26)	961
ORG	80.15 (±0.89)	84.84 (±0.48)	82.43 (±0.69)	521
GPE_LOC	93.32 (±0.67)	88.93 (±0.95)	91.06 (±0.40)	428
LOC	78.38 (±3.15)	75.26 (±1.85)	76.74 (±1.64)	185
PROD	56.95 (±2.49)	53.12 (±4.38)	54.81 (±2.37)	131
DRV	76.92 (±2.29)	74.08 (±2.96)	75.39 (±1.05)	78
GPE_ORG	64.26 (±1.91)	75.91 (±3.44)	69.51 (±0.89)	61
EVT	64.29 (±11.95)	52.25 (±6.51)	57.12 (±6.40)	14
MISC	35.71 (±21.19)	73.78 (±37.64)	46.66 (±25.07)	14
Micro avg.	86.90 (±0.55)	86.26 (±0.47)	86.58 (±0.45)	2393

Table 6.3: Scandi-ner NorNE test dataset results measured with precision, recall and F1 for NER with the strict evaluation criteria, where the named entity token boundary has to be correct as well as predicted label. Support column denotes number of entries for each class.

6.1.3 Binary Classification Evaluation

Table 6.4 presents precision, recall and F1 metrics for all seven models, in the binary classification task using the NorNE dataset. All of the fine-tuned BERT models exceed the performance in all metrics compared to the baseline models.

Overall, Ner-Scandi's macro-average F1 of 97.07 is the greatest, whereas the poorest is observed in mBERT's results with a macro-average F1 at 95.51.

Baselines

Comparing the baseline models, both yield similar F1 macro-averages, NCRF++ with 78.3 and SpaCy with 80.17, for the binary classification. Both models exhibit great precision and recall for the Not Capital Letter class, the class which encapsulates all words that are not named entities.

For the Capital Letter class, the class that refers to tokens that are named entities, and consequently should be capitalized, the baseline models exhibit a poor recall metrics. SpaCy's and NCRF++'s recall of 48.4 and 45.36, respectively, indicating that these models would fail to detect capitalization errors more than half the time. Precision however, is much greater than the recall, with at NCRF++ at 83.27 precision and SpaCy at 87.38. The recall for the baselines suggests that there is a lot of false negatives for the Capital Letter class. However, the precision suggest that at least when they predict a class to be of the Capital Letter class, the majority of the time the prediction is a true positive.

Model	Class	Precision	Recall	F1	Support
SpaCy*	0	96.62	99.53	98.05	51047
	1	87.38	48.40	62.30	3446
	Macro avg.	92.00	73.97	80.17	54493
NCRF++	0	96.41	99.38	97.88	51271
	1	83.27	45.36	58.73	3468
	Macro avg.	89.84	72.37	78.30	54739
mBERT	0	99.57 (±0.08)	99.31 (±0.07)	99.44 (±0.01)	51271
	1	89.78 (±1.10)	93.46 (±1.06)	91.57 (±0.19)	3468
	Macro avg.	94.68 (±0.51)	96.38 (±0.49)	95.51 (±0.10)	54739
NorBERT	0	99.58 (±0.09)	99.49 (±0.07)	99.54 (±0.02)	51271
	1	92.51 (±0.99)	93.73 (±1.18)	93.10 (±0.19)	3468
	Macro avg.	96.04 (±0.45)	96.61 (±0.56)	96.32 (±0.10)	54739
NorBERT2	0	99.60 (±0.03)	99.51 (±0.02)	99.56 (±0.01)	51271
	1	92.81 (±0.37)	94.00 (±0.44)	93.40 (±0.14)	3468
	Macro avg.	96.21 (±0.17)	96.76 (±0.21)	96.48 (±0.08)	54739
NB-BERT	0	99.64 (±0.04)	**99.62 (±0.05)**	99.63 (±0.01)	51271
	1	**94.31 (±0.71)**	94.70 (±0.58)	94.50 (±0.16)	3468
	Macro avg.	**96.98 (±0.34)**	97.16 (±0.27)	97.06 (±0.09)	54739
Ner-Scandi	0	**99.69 (±0.02)**	99.58 (±0.02)	**99.63 (±0.02)**	51271
	1	93.73 (±0.31)	**95.30 (±0.24)**	**94.51 (±0.24)**	3468
	Macro avg.	96.71 (±0.16)	**97.44 (±0.12)**	**97.07 (±0.13)**	54739

*Table 6.4: NorNe Binary Classification precision, recall and F1 metrics for all models. The 0 class denotes the Not Capital Letter class. The 1 class denotes the Capital Letter class. The Support column refers to the number of entries for a given class. *10 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length.*

BERT Models' Not Capital Letter Classification

For the Not Capital Letter classifications, the variance between the precision, recall, and F1 scores of the BERT models are minimal. The difference between the best precision (99.69, achieved by Ner-Scandi) and the worst (99.57, achieved by mBERT) is only 0.12%. Furthermore, the best recall is at 99.62 (NB-BERT), and worst recall at 99.31 (mBERT), a difference of 0.31%.

Having such excellent scores for precision and recall, results in high F1 scores across all BERT models for the Not Capital Letter class. mBERT performed the poorest, yet with a F1 at 99.44, and Ner-Scandi at 99.63 being the best, makes the difference best/poorest them just 0.21%. All BERT models' F1 scores for this class is subject to an negligible variance between the different seeds, resulting in a low standard deviation (highest $SD = 0.02$).

BERT Models' Capital Letter Classification

For the Capital Letter classification, the variance between their precision, recall, and F1 scores is significantly higher compared to the Not Capital Letter class. NB-BERT 94.31 precision rate is the best, whereas mBERT 89.78 is the worst among the BERT models.

Recall values also vary more for the different BERT models for this class, compared to the Not Capital Letter class. The recall value difference between the best performing and poorest performing model is 1.84%, Ner-Scandi achieving the best (95.30) and mBERT once again perform the worst (93.46).

The resulting difference in F1 for this class comparing the best model against the worst is 2.94%. Ner-Scandi performed the best (F1 94.51), and mBERT the worst (91.57).

6.1.4 NorNE Results Analysis and Discussion

NER without Token Boundary and Strict Match

As presented in subsection 6.1.1, we see that less abundant classes more frequently get misclassified, such as the `MISC` and `EVT` classes. This is the case for the both the NER without token boundary and strict match evaluation.

By considering the results in Figure 6.2 corresponding to the Ner-Scandi predictions, we see that there are instances where the model tries to predict a `B-MISC` token and fails the classification. Instead the model often predicted the token to be either `B-PER` or `B-PROD`.

It does make sense that the miscellaneous class would be hard to predict, not only because it has very few instances of `MISC` in the training data. To reiterate, the `MISC` class encapsulates all named entities that does not fit the other named entity classifications.

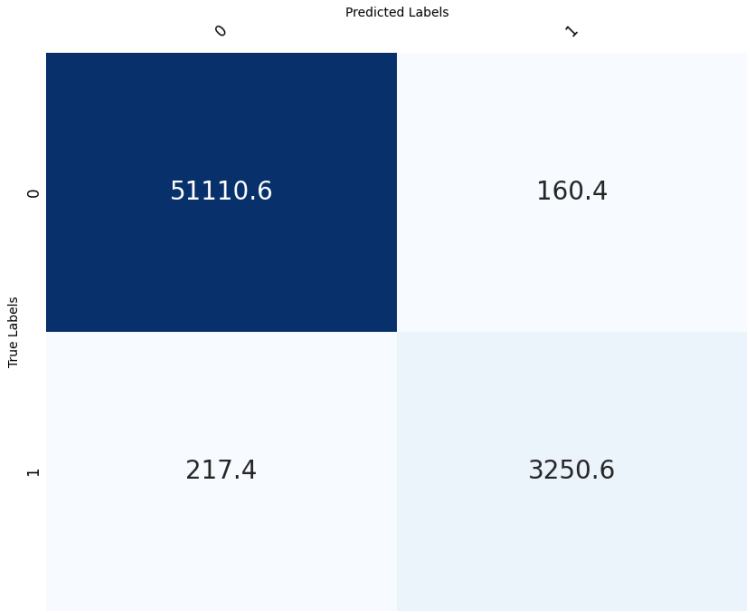


Figure 6.1: Ner-Scandi Binary Classification results for the NorNE test dataset. The 0 class denotes the Not Capital Letter class, that is non-entity tokens. The 1 class denotes the Capital Letter class, that is named entities.

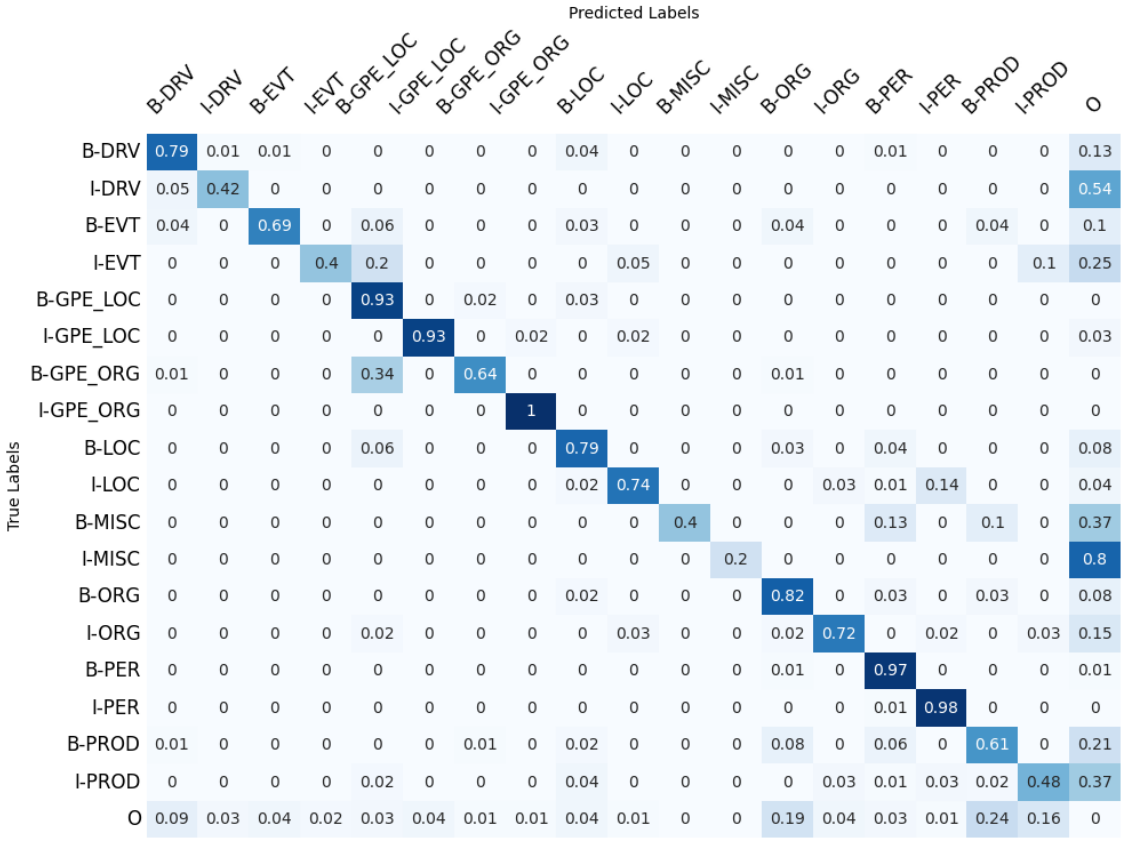


Figure 6.2: Ner-Scandi confusion matrix results for the NorNE dataset. The numbers are normalized averages that indicate percentage instead of actual count. To see non-normalized version, refer to Appendix B.

For more frequent entries, such as the LOC labeled tokens, it would seem that Ner-Scandi most frequently misclassify the tokens as either GPE_LOC, ORG, or PER labeled tokens. It is perhaps not surprising that the model might mix up LOC for GPE_LOC, as they are almost interchangeable. Likewise, GPE_LOC often gets misclassified as LOC. That a model that could mix named locations and a persons name also has some rationale, since many surnames are named after locations. The opposite may also apply, as many places are named after people.

Notably, for the majority of classes, the model misclassifies named entities as O (outside token), the non-named entity type class. This misclassification is especially common among B-PROD, I-PROD, I-DRV, B-MISC and I-MISC. Perhaps one of the reasons that PROD often gets misclassified as a non-named entity is because products can be named after about anything. Be it nouns, verbs, or concepts, it may make the PROD class indistinct in sentence when it is not properly capitalized.

DRV and MISC are less frequent in the training dataset, compared to most other classes. This may have resulted in worse performance for these classes, if the models did not have sufficient data to properly capture patterns of associated with them.

Binary Classification

For the binary classification the BERT models excel compared to the baseline models for the NorNE test dataset. The biggest difference in performance for the BERT models is the Capital Letter class precision.

The performance difference between NB-BERT and Ner-Scandi are marginal. This makes sense since they are essentially the same model, the only difference being that Ner-Scandi had its weights fine-tuned for NER in several Scandinavian languages before being fine-tuned once again by us. We cannot conclude whether the fact that Ner-Scandi was fine-tuned before undergoing additional fine-tuning would provide the model with a performance advantage over NB-BERT, but it could certainly be the case.

For the binary classification, Ner-Scandi has a combined misclassification rate of 1.57%. By analysing the confusion matrix, we can calculate the false positive rate and the false negative error rate. For the scenario where the model might predict a token to be capitalized when it should not, the false positive rate is at 0.31%. For the opposite case, where the model may predict that a token should not be capitalized, when in reality it should, the false negative error rate is 6.27%.

6.2 NPSC results

In the previous section we discussed the NorNE results. A similar section structure will follow as we present the results for the NPSC. First, we examine the NER results for the NPSC annotated sample dataset, focusing on both NER

Model	No Boundary	Strict Match	Binary Classification
SpaCy*	52.42	54.24	83.36
NCRF++	35.69	33.07	75.42
mBERT	81.88 (±0.67)	80.27 (±0.87)	95.17 (±0.52)
NorBERT	81.71 (±1.74)	79.00 (±1.74)	96.19 (±0.49)
NorBERT2	80.77 (±0.81)	77.59 (±0.73)	96.52 (±0.19)
NB-BERT	**84.44** (±0.55)	**82.30** (±0.84)	**96.62** (±0.22)
Ner-Scandi	83.72 (±1.00)	81.08 (±1.33)	96.26 (±0.37)

*Table 6.5: The annotated NPSC sample dataset results for all models. BERT models use the average F1 across all seeds for a given model. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary Classification denotes the task where only the detection if an entity is present, without specific type associated. *3 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length.*

and NER with strict match criteria. Second, we analyse and discuss the binary classification results for the full NPSC. Finally, we discuss the results from these three evaluations and delve deeper into an analysis of their possible implications.

6.2.1 Annotated NPSC sample dataset results

This subsection begins by highlighting test results for the annotated NPSC sample dataset. Table 6.5 presents an overview for all model's F1 score metrics, corresponding to the respective evaluation: the NER without boundary, strict match classification, and binary classification.

As for the NorNE dataset, the NCRF++ model performs the poorest for all F1 measures for the NPSC sample dataset. The SpaCy pipeline model performs significantly better than the NCRF++, but is, nevertheless, still subpar compared to the BERT models.

Contrary to the NorNE results, Ner-Scandi does not perform the best. Rather, NB-BERT achieved the greatest F1 for all evaluations here.

If we compare the results table for the NPSC sample, Table 6.5, with the NorNE classification results, Table 6.1, it is apparent that both the NER evaluation without boundary and strict match evaluation, the F1 scores have decreased significantly.

The most signficant decrease in performance is observed for NorBERT2. For the NER without token span boundary, the decrease in F1 is 4.68, and the strict match evaluation F1 decrease is 6.79. Interestingly, NorBERT performs better than NorBERT2 on these two evaluations, which is not the case for the NorNE

NE Type	Precision	Recall	F1	Support
B-PER	93.02 (±2.08)	93.05 (±2.43)	93.03 (±2.05)	43
I-PER	97.00 (±4.00)	100.00 (±0.00)	98.43 (±2.10)	20
B-ORG	89.35 (±3.22)	80.28 (±1.80)	84.51 (±0.95)	77
I-ORG	100.00 (±0.00)	65.17 (±9.80)	78.52 (±6.65)	11
B-GPE_LOC	88.57 (±2.67)	85.12 (±3.61)	86.73 (±1.65)	28
I-GPE_LOC	66.67 (±0.00)	92.00 (±9.80)	77.09 (±3.56)	6
B-LOC	100.00 (±0.00)	37.33 (±3.27)	54.29 (±3.50)	2
I-LOC	0	0	0	0
B-PROD	88.00 (±9.80)	54.76 (±7.45)	66.93 (±5.47)	5
I-PROD	0	0	0	0
B-DRV	33.33 (±0.00)	80.00 (±24.49)	46.00 (±4.90)	3
I-DRV	0	0	0	0
B-GPE_ORG	73.00 (±2.45)	89.18 (±4.16)	80.23 (±2.52)	20
I-GPE_ORG	0	0	0	0
B-EVT	0	0	0	0
I-EVT	0	0	0	0
B-MISC	0	0	0	1
I-MISC	0	0	0	0
Micro avg.	87.96 (±0.77)	81.20 (±0.60)	84.44 (±0.55)	216

Table 6.6: NB-BERT results for the NPSC sample dataset, measured with precision, recall and F1 for NER without considering token span of a named entity. Support column denotes number of entries for each class.

results.

The model that shows the least variance in results between the NorNE dataset and the NPSC sample dataset is mBERT. Additionally, mBERT outperforms NorBERT and NorBERT2 in all evaluation tasks except for the binary classification evaluation. Note that for the following subsections, we will not go into more details about the binary classification for annotated NPSC sample dataset. The binary classification discussions are saved for when the results for the whole NPSC dataset is presented in Section 6.2.4.

6.2.2 NPSC Sample Named Entity Recognition

As mentioned in the previous section, NB-BERT performs the best when tested on the annotated NPSC sample dataset. Table 6.6 presents the combined average results across all model seeds, with precision, recall and F1 values. The model achieves a micro-average precision score of 87.77, a recall score of 81.20, and F1 of 84.44. For this dataset it is important to stress that there are far less tokens in total, resulting in less named entities to classify.

The `B-PER` and `I-PER` achieve the greatest F1 among the different named entity classes. The `I-PER` class resulted in a perfect recall score of 100, including an excellent precision of 97.00. The most frequent class, `B-ORG` exhibit a satisfactory precision rate of 89.35, however, the recall is only 80.28. The `I-ORG` class recall of 65.17 is subpar, although the precision is a perfect 100.

6.2.3 NPSC Sample Strict Match

Table 6.7 provides an overview of how well NB-BERT performed NER with the strict match classification criteria, that is when named entity token boundary is taken into account.

For all NB-BERT model seeds, the micro-average precision is 84.69 and the recall rate is 80.05, resulting in a micro-average F1 of 82.30.

The performance observed for the `PER` class is good, with F1 of 92.11. As we observed in the previous subsection, NB-BERT especially good at classifying `I-PER` correctly, with F1 of 98.43, however, only reaching F1 of 93.03 for the `B-PER` class. This suggests that the `B-PER` predictions are mostly to blame for the `PER` strict match classification not getting a higher F1 than 92.11.

`ORG`, the most frequent class, has an average precision rate of 85.97, but unfortunately suffers from a significantly lower average recall of 77.19, resulting in an F1 of 81.29. `GPE_LOC`, and `GPE_ORG` are relatively frequent in the dataset, resulting in F1 values of 83.23 and 80.23 respectively.

NE Type	Precision	Recall	F1	Support
PER	92.09 (±2.37)	92.14 (±3.33)	92.11 (±2.73)	43
LOC	100.00 (±0.00)	37.33 (±3.27)	54.29 (±3.50)	2
ORG	85.97 (±4.14)	77.19 (±1.19)	81.29 (±1.69)	77
GPE_LOC	85.00 (±1.43)	81.73 (±3.77)	83.25 (±1.32)	28
GPE_ORG	73.00 (±2.45)	89.18 (±4.16)	80.23 (±2.52)	20
PROD	88.00 (±9.80)	54.76 (±7.45)	66.93 (±5.47)	5
EVT	0	0	0	0
DRV	33.33 (±0.00)	80.00 (±24.49)	46.00 (±4.90)	3
MISC	0	0	0	1
Micro avg.	84.69 (±1.15)	80.05 (±1.01)	82.30 (±0.84)	179

Table 6.7: NB-BERT results for the NPSC sample dataset measured with precision, recall and F1 for NER with the strict evaluation criteria, where the named entity token boundary has to be correct as well as predicted label. Support column denotes number of entries for each class. report

Model	Class	Precision	Recall	F1	Support
SpaCy*	0	99.76	98.28	99.02	1013745
	1	57.27	90.77	70.23	25723
	Macro avg.	78.51	94.52	84.62	1039468
NCRF++	0	97.38	99.77	98.56	1014791
	1	85.85	34.04	48.75	41247
	Macro avg.	91.61	66.91	73.66	1056038
mBERT	0	99.41 (±0.06)	99.64 (±0.03)	99.52 (±0.02)	1014791
	1	91.09 (±0.70)	86.18 (±1.19)	88.56 (±0.39	41247
	Macro avg.	95.25 (±0.32)	92.91 (±0.58)	94.04 (±0.20)	1039468
NorBERT	0	99.41 (±0.11)	99.74 (±0.04)	99.58 (±0.04)	1014791
	1	93.69 (±1.04)	86.64 (±1.97)	90.00 (±0.66)	41247
	Macro avg.	96.55 (±0.47)	93.19 (±0.97)	94.79 (±0.35)	1039468
NorBERT2	0	99.39 (±0.05)	99.79 (±0.02)	99.59 (±0.02)	1014791
	1	94.81 (±0.47)	86.42 (±0.89)	90.41 (±0.36)	41247
	Macro avg.	97.10 (±0.22)	93.10 (±0.44)	95.00 (±0.19)	1039468
NB-BERT	0	99.35 (±0.04)	**99.85 (±0.02)**	99.59 (±0.02)	1014791
	1	**96.22 (±0.47)**	85.66 (±0.78)	90.63 (±0.43)	41247
	Macro avg.	**97.78 (±0.23)**	92.76 (±0.39)	95.11 (±0.23)	1039468
Ner-Scandi	0	**99.42 (±0.05)**	99.82 (±0.02)	**99.62 (±0.02)**	1014791
	1	95.71 (±0.39)	**87.03 (±0.97)**	**91.16 (±0.43)**	41247
	Macro avg.	97.56 (±0.18)	**93.43 (±0.48)**	**95.39 (±0.23)**	1039468

*Table 6.8: NPSC NorNe Binary Classification precision, recall and F1 metrics for all models. The 0 class denotes the Not Capital Letter class. The 1 class denotes the Capital Letter class. The Support collumn refers to the number of entries for a given class. *729 sentences were dropped from SpaCy pipeline because of difference prediction sequence length and gold standard sequence length.*

6.2.4 Binary Classification

Table 6.8 presents precision, recall and F1 for all seven models, for the binary classification evaluation using the full NPSC dataset. For this evaluation once again all of the fine-tuned BERT models exhibit far greater performance in all metrics compared to the baseline models.

The greatest macro-average F1 scores of 95.39 is observed in the results from Ner-Scandi. The worst macro-average F1 of 94.04 is observed in the results from mBERT.

Baselines

Both the baseline models yield similar F1 macro-averages, NCRF++ with 73.66 and SpaCy with 84.62. As observed in the NorNE results, both models exhibited excellent precision and recall for the Not Capital Letter class. This is also the case for the NPSC results.

Similar to the results gathered from the NorNE dataset, NCRF++ exhibits poor recall for the NPSC. For the Capital Letter class, NCRF++'s recall is only 34.04, however, precision is much higher, averaging at 85.85.

For the SpaCy pipeline, we observe the opposite of what NCRF++ performs. For the Capital Letter class the recall is 90.77, but, the precision is only 57.27. This indicates that SpaCy's predictions produce more false positives than NCRF++ for the Capital Letter class.

BERT Models' Not Capital Letter Classification

For the Not Capital Letter class in BERT models' performance, the variance between the precision, recall, and F1 scores are minimal. Ner-Scandi precision rate of 99.42, is the best, and NB-BERT's precision rate of 99.35 is the worst, among the BERT models.

Similar performance applies for the recall results among the BERT models. NB-BERT achieved the highest recall rate (99.85), and mBERT the worst (99.64). With great precision and recall for all BERT models, F1 scores are correspondingly great. mBERT has the lowest F1 of 99.52, and Ner-Scandi the highest with an F1 of 99.62.

BERT Models' Capital Letter Classification

Similar trends as observed in the NorNE results, are again observed in the NPSC results where the Capital Letter class metrics are significantly lower than its class counterpart. NB-BERT's 96.22 precision rate is the best, whereas mBERT's 91.09 is the worst precision for the BERT models.

Ner-Scandi exhibits the greatest recall rate at 87.03, whereas NB-BERT surprisingly performs the poorest of the BERT models with a recall rate of 85.66.

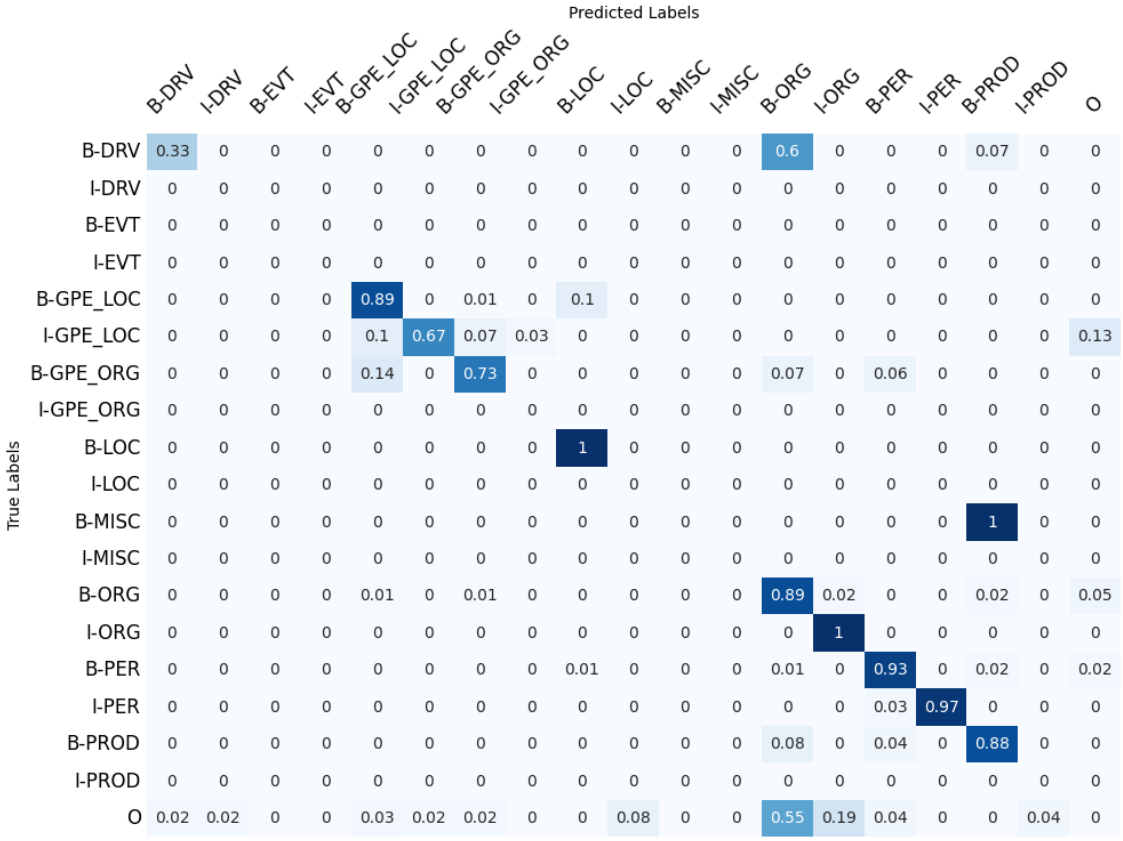


Figure 6.3: NB-BERT confusion matrix results for the NSPC sample dataset. The numbers are normalized averages that indicate percentage instead of actual count. To see non-normalized version, refer to Appendix B.

Comparing mBERT, NorBERT and NorBERT2, it is surprising that NB-BERT has the lowest recall since it has generally been performing better than these models at the other evaluations. Nevertheless, NB-BERT's 90.63 F1 score for this class is still second best, the best being Ner-Scandi's 91.16 F1.

6.2.5 NPSC Results Analysis and Discussion

NER without Token Boundary and Strict Match

Analysing the confusion matrix corresponding to NB-BERT's predictions for the annotated NPSC sample results (Figure 6.3), it is apparent that the model misclassified the `GPE` classes quite often. For example incorrectly predicting `B-GPE_LOC` as `B-GPE_ORG`.

There are other instances where the model classified a non-named entity class `O` to be a named entity. For instance, 55% of the time these incorrect classifications are predicted as `B-ORG`, and 19% of the time `I-ORG`

There are classes such as `B-DRV`, and `B-MISC`, that have greater misclassification rates than any other, as observed in the confusion matrix of Figure 6.3. However, the dataset contains only three instances of `B-DRV` and one instance of

B-PROD. As a result, any misclassification of these infrequent classes significantly impact the results in the negative direction. More instances of these classes would be preferable to infer any conclusive performance.

Binary classification

NB-BERT achieved the greatest average-macro precision of 97.78. The second best, Ner-Scandi's average-macro precision, is 97.56. For macro-average recall and macro F1, Ner-Scandi outperformed all other models. Ner-Scandi produced the highest F1 score (95.39), while mBERT the lowest (94.04). That makes the greatest F1 difference between all BERT models only 1.35% for the binary classification evaluation.

Analysing the confusion matrix we can calculate the total error rate, false positive rate, and the false negative error rate. For the binary classification, Ner-Scandi has a combined error rate of 1.57%.

The false positive rate, the scenario where the Ner-Scandi would predict a token to be capitalized when it should not be classified as so, is on average 4.29%. The scenario where the model may predict that a token should not be capitalized, when in reality it should, yield the false negative error rate of 4.29%.

All the models, baselines included, have few issues with classifying the No Capital Letter class. The Capital Letter class is significantly more difficult for the models to predict correctly.

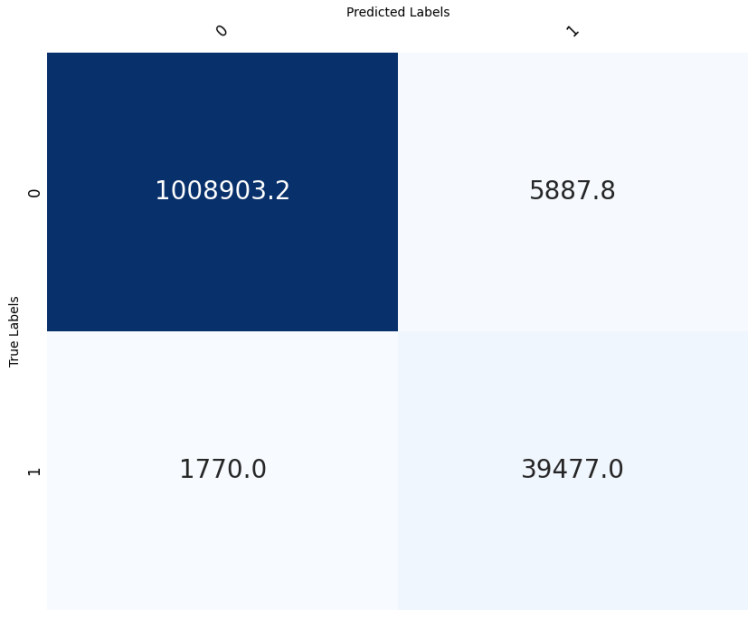


Figure 6.4: Ner-Scandi Binary Classification results for the full NPSC dataset. The 0 class denotes the Not Capital Letter class, that is, non-entity tokens. The 1 class denotes the Capital Letter class, that is, named entities.

No Boundary	Precision	Recall	F1	Support
NorNE	86.35 (±0.64)	87.79 (±0.49)	87.06 (±0.56)	3468
NPSC sample	87.96 (±0.77)	81.20 (±0.60)	84.44 (±0.55)	216
Strict Match				
NorNE	86.90 (±0.55)	86.26 (±0.47)	86.58 (±0.45)	2393
NPSC sample	84.69 (±1.15)	80.05 (±1.01)	82.30 (±0.84)	179
Binary Classification				
NorNE	96.71 (±0.16)	97.44 (±0.12)	97.07 (±0.13)	54739
NPSC	97.56 (±0.18)	93.43 (±0.48)	95.39 (±0.23)	1039468

Table 6.9: Ner-Scandi result comparisons between the NorNE and the NPSC dataset. The NPSC sample are results from NB-BERTS performance. The micro averages are reported for the No Boundary and Strict Match. Macro averages are reported for the Binary Classification. No Boundary denotes NER without named entity token span taken into account. Strict Match denotes NER with named entity token span taken into the evaluation. Binary Classification denotes the task where only the detection if an entity is present, without specific type associated.

6.3 Comparing NorNE and NPSC results

Table 6.9 presents the results for for each evaluation according to its respective dataset. For the annotated NPSC sample results, NB-BERT's results are reported, since this model performed the greatest for this dataset. For the NorNE test dataset and the full NPSC, Ner-Scandi's results are reported.

Comparing results for the NER evaluation without token boundary, NB-BERT has about 1% higher precision for the annotated NPSC sample dataset results, compared to Ner-Scandi's NorNE results. For NER evaluation without token boundary, F1 scores from tests on NorNE is about 2.6% higher than the results from the annotated NPSC results. This indicates that our models have achieved some capabilities of generalizing across different data. Though it is important to note that the annotated NPSC has far less class diversity and class count than the NorNE dataset. This may skew the results in favour of the annotated NPSC results if one class is especially frequent and classified correctly. Therefore, while the NB-BERT's performance on the NPSC sample is promising, further evaluation on a more diverse range of datasets is necessary to fully assess its ability to generalize.

The previous statement also applies to the strict match evaluation, where the NB-BERT model obtained significantly lower scores for the NPSC sample compared to Ner-Scandi's performance for the NorNE dataset. NorNE dataset Strict Match classification resulted at a micro-average of 86.58, compared to 82.30 for the NPSC sample.

For the binary classification, the Ner-Scandi results show a slightly lower

macro-average F1 for the NPSC (95.39), compared to its F1 for the NorNE dataset (97.07). The differences in F1 is largely due to about 4% lower macro-average recall from the NPSC. As discussed in Section 6.2.4, the Capital Letter class suffered the lowest recall rate (87.03), whereas the Not Capital Letter predictions are near perfect recall rate (99.82) and precision (99.42), as presented in Table 6.8.

Performance for the Not Capital Letter class is very similar in both evaluation of the NorNE and NPSC datasets. However, for the Capital Letter class, the models' recall rate the are significantly higher for NorNE dataset test results, with a macro-average of 95.30 (see Table 6.4), compared to the NPSC results of 87.03, a difference of approximately 8%.

6.4 Possible Limiting Factors for Model Performance

There are several factors that contributed to limit the performance of the BERT models for both the NorNE dataset and the NPSC. The following list discuss some plausible limitations, and why the NorNE results were generally better than the NPSC results.

1. *Overfitting*: When a model was trained on the NorNE dataset, the model may have learned patterns that are specific to their training data. The optimal selection of hyperparameters such as dropout and regularization could help to alleviate overfitting issues. However, if the training data does not represent data in the real world, the model will most likely have a hard time generalizing on new data.

   Even if the validation and test datasets come from the same source and follow a similar data pattern, it is common to observe some variance in the validation and the test dataset results. As Jørgensen et al. (2020) observed, there was up to 7% F1 variance between the validation and test set in their testing. Nevertheless, for our binary classification evaluation, it is curious how the NPSC recall for the Capital Letter class is about 8% lower than the NorNE test dataset. This could be a sign of overfitting.

   A portion of NorNE contain parliamentary transcripts, but news text are far more dominant (Jørgensen et al., 2020). Since the training data are mostly news text, it may be the case that the models have learned pattern to detect named entities in text articles. Perhaps the models perform better at predicting named entities in written texts as opposed to transcript derived from oral communication. Oral communication may differ in sentence structure, additionally the transcribed speech in the NPSC dataset is formal in nature, which further complicates the sentence structure. It can be argued that the NPSC dataset may not accurately represent the way people in the

general public communicate with one other. Perhaps in a different speech-to-text transcribed dataset the models' performance would differ from the NPSC.

2. *Label noise*: The labeled data for either training or test data might be incorrect. NorNE is professionally annotated and reviewed by multiple annotators, which makes the dataset less prone to noisy data.

   However, the NPSC sample data was annotated by the author of this thesis, who is not a professional annotator. The annotations have not been reviewed by other annotators either, so the sample dataset may contain incorrect labels. The annotation process used a best practise approach, where if there was any doubt if an annotation was correct, the NorNE dataset would serve as reference work if the same word was present in NorNE in a similar context. Even though such an approach was used, errors may occur. If a label is incorrectly annotated, this could lead to misleading model performance.

3. *Quality of training data*: Both the quality and quantity of data are crucial factors when training a model that is intended to generalize across multiple data sources. If a model lacks sufficient task-specific training material to learn the patterns of the data, it is unlikely that it will give good predictions.

   As mentioned, the NorNE dataset consists mostly of news texts. As a result, our models may be susceptible to overfitting, which could be due to the disproportionate distribution of content types within the dataset training data, which may lead to poor generalization performance. Perhaps, if we had more data similar in text structure of the NSPC dataset, or just more diverse training data in general, the models would perform better.

4. *Inadequate training method*: There might be better ways to train a model to detect named entities regardless of capitalization.

   To reiterate, our models were trained on the NorNE dataset where every token was forced lowercase. Perhaps a mix between capitalization and lowercase would produce better results.

   A related aspect is the number of named entity classes the model was trained on. Instead of using NorNE-full, which entails using all 8 labels, perhaps, NorNE-7 where geo-political labels (`GPE`) are conflated into a single type would yield better results. An even further reduction in the number of classes that the model trained on could perhaps have been better for the binary classification.

5. *Pre-trained language model type*: Other pre-trained language model may have produced better results when trained. this includes larger BERT

models, such as NB-BERT$_{\text{LARGE}}$[2] and MegatronBERT[3], which both have twice the number of hidden layers, and a higher attention head count compared to the models used in this thesis.

It is also possible that the generative pre-trained transformer (GPT) models would produce similar or better results if the prompt requests were properly fine-tuned and tested, compared to the specialized fine-tuned BERT approach.

6. *Hyperparameter combination*: A grid search for optimal hyperparameters could be beneficial, but would require larger computing power and longer project time than the restrictions of the current research allowed. A grid search involves that one automates the search to find the best combination of hyperparamters by trial and error. In practise, it could be that one has a set of different optimiser types, a set of loss function type, schedulers, number of epochs and so on. All these options within a set of hyperparameters are tried with every combination of the other set of hyperparameters to find the best tuning.

   A downside of a grid search that cannot be understated, is the amount of time a model would spend training just to find the optimal hyperparameters. In practise, with every new option in a grid search the training time could increase exponentially if we were to exhaustively search all possible combinations in the different sets of parameters. Say we had $3^4 = 81$ different combinations, and we added another hyperparameter with 3 possible values. This would lead to a total of $3^5 = 243$ combinations.

   Even if a training epoch lasts only about 3 minutes, which was the case for our BERT training, having a model train for 50 epochs will results in 150 minutes training for one combination of hyperparameters. Taking those 150 minutes and multiplying them with 243 combinations, yields an approximate calculation of how long the computer for our experiments would run full power, non-stop just to get optimal hyperparameters: about 25 days.

6.5 Results Summary

For the evaluation results for the NorNE test dataset and the NPSC, we observed that Ner-Scandi performed best among the models tested. NB-BERT achieves similar results to Ner-Scandi, where the differences is hardly statistically significant . From a technical standpoint, the similar performance of NB-BERT and Ner-Scandi is not surprising, as they are essentially the same model. The only difference is that Ner-Scandi was fine-tuned on additional Scandinavian data before further fine-tuning. Notably though, NB-BERT is the best performer for

[2] https://huggingface.co/NbAiLab/nb-bert-large
[3] https://huggingface.co/docs/transformers/model_doc/megatron-bert

the annotated NPSC sample dataset, but not by a large margin compared to the second-best model, Ner-Scandi.

Comparing the results from the NorNE test dataset and the NPSC, the results from NorNE are significantly better than the results from the NPSC. This may indicate an overfitting issue, however, further testing on different datasets is needed to draw this conclusion.

Ner-Scandi's results on the NorNE dataset yield a micro-average of F1 of 87.06 for NER evaluation without token boundary, and a F1 micro-average of 86.58 for the strict match evaluation. Lastly, Ner-Scandi achieved a macro-average of 97.07 when evaluating the binary classification performance.

For the IOB2 annotated NPSC sample dataset results, NB-BERT performed the best, but not by a huge margin compared to the second best model, Ner-Scandi. For NER evaluation without token boundary, NB-BERT's micro-average resulted in a F1 of 84.44, and 82.30 micro-average for the strict match evaluation.

For the full NPSC dataset binary classification task, Ner-Scandi achieved highest macro-average, 95.39, only 0.28% higher than NB-BERT's macro-average.

Chapter 7

Conclusions and Future Work

To answer our first research question (R1) “To what extent can current neural models be used to identify non-capitalized named entities?”, our findings strongly indicate that BERT-models are highly capable of identifying non-capitalized named entities. This includes both the traditional approach of NER, where named entities given a specific named entity class, and the binary classification approach, which involves detecting named entities without labeling them for a specific class.

For our second research question (R2) ”Which current neural models achieves best NER results in all lowercase texts?”, we conclude that fine-tuned BERT models outmatch our baselines models (SpaCy and NCRF++) by a significant margin. Specifically, the Ner-Scandi and NB-BERT performed best in our testing.

7.1 Contributions

Our research focus is on NER in all lowercase text, an area of research that is largely unexplored. In this regard, our hope is that this thesis may provide valuable insight into how to improve the efficiency of NER applications for text where named entities include capitalization errors. These models could in turn be used for post processing of speech-to-text system to improve proper capitalization.

We have provided a thorough description of the experimental setup, assessment methods and evaluation metrics with the aim of making replication and validation of this study easy for researchers interested to undertake similar research.

The research presented in this thesis has the potential to benefit various industries, such as the mass media and the technology industry. The media industry could deploy similar approaches to improve automatically generated speech-to-text transcriptions for capitalization errors. Our findings include suggestions of techniques for creating robust NER models that can detect named entities even when they appear in lowercase in text, which may increase the efficiency of downsteam tasks such as information extraction systems.

As an additional contribution to NER research, a small annotated dataset was

created using sentences from the Norwegian Parliamentary Speech Corpus. This dataset will be made publicly available and can be used to further improve NER models for the Norwegian language.

7.2 Limitations

Section 6.4 covers some potential limiting factors that could have impacted the performance of our models and possible approaches to addressee these shortcomings. In this section, we will briefly mention some of the limitations of this study.

This study only presents results from datasets that were forced to be lowercase. However, in a real-world scenarios there may be a mix of capitalization errors and properly capitalized words. It is important to keep this in mind because if our models were tested on mixed capitalization, rather than just lowercase text, their behavior would most likely differ from the reported results.

Another limiting factor of this study is the limited amount of annotated data, besides the NorNE test set. To confidently conclude if the models have achieved the ability to generalizing its performance across data, we would ideally want more test datasets, with more annotated named entities than the annotated NPSC sample dataset.

As discussed in Section 6.4, the annotated NPSC sample dataset, besides from being limited in size, might contains mislabeled named entity types. It is important to acknowledge these limitation and recommend reader to consider its potential impact on the research outcome.

7.3 Future Work

While the findings of this study provide valuable insights on NER, there are several opportunities for future research to build upon and expand the current of understanding how to use transformer-based language models to tackle NER. The following sections outline potential directions for further investigation.

Firstly, in regards to the limitations highlighted above, we encourage researchers to explore new approaches to conduct similar studies, e.g. by exploring alternative training methods, using different training data, and utilizing hyperparameter tuning.

Further research can also be conducted to see the efficiency of our approach to NER and possible capitalization correction in industry setting. Specifically it would be great to get insight into how these types of NER models would fit as a component of a larger system perhaps as a one of multiple layers to detect and correct capitalization errors.

We also encourage researchers to explore the use of larger and different types of language models for similar tasks as the those explored in thesis thesis. A

special area of interest would be to compare performance for the same tasks, using powerful auto-regressive generative pre-trained transformer models, such as ChatGPT[1], GPT-4[2] or similar.

[1] https://chat.openai.com/chat
[2] https://openai.com/product/gpt-4

Appendix A

Extended Training Details

This appendix provides specific hyperparameters for NCRF++, and detailed training metrics for each BERT model and every seed, presented in tabular form. The metrics include the epoch number, along with its corresponding validation dataset recall rate, training loss, validation loss, and learning rate for that particular epoch.

```
### I/O ###
train_dir=NorNE_data/train_split.bmes
dev_dir=NorNE_data/dev_split.bmes
test_dir=NorNE_data/test_split.bmes
model_dir=model_name
word_emb_dir=Embeddings\132\model.txt

norm_word_emb=False
norm_char_emb=False
number_normalized=True
seg=True
word_emb_dim=600
char_emb_dim=30

###NetworkConfiguration###
use_crf=True
use_char=True
word_seq_feature=LSTM
char_seq_feature=CNN

###TrainingSetting###
status=train
optimizer=SGD
iteration=50
batch_size=64
ave_batch_loss=False

###Hyperparameters###
cnn_layer=4
char_hidden_dim=50
hidden_dim=200
dropout=0.5
lstm_layer=1
bilstm=True
learning_rate=0.015
lr_decay=0.05
momentum=0
l2=1e-8
gpu
```

Figure A.1: Configuration file of the NCRF++ model

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7914	68.5052	3.9147	2.00e-05
2	0.8234	22.3259	3.5576	2.00e-05
3	0.8239	13.9207	3.6928	2.00e-05
4	0.8350	9.2733	3.6146	2.00e-05
5	0.8539	6.8918	3.5897	2.00e-05
6	0.8401	5.3763	4.0479	1.86e-05
7	0.8433	3.9361	4.1711	1.86e-05
8	0.8440	3.2099	4.6394	1.73e-05
9	0.8389	2.9028	4.6286	1.73e-05
10	0.8530	2.0669	4.4494	1.61e-05
11	0.8375	1.8255	4.9256	1.61e-05
12	0.8442	1.5254	5.1655	1.50e-05
13	0.8387	1.4220	5.1197	1.50e-05
14	0.8479	1.4381	5.1185	1.39e-05

Table A.1: mBERT seed 1024 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8052	72.8943	3.8747	2.00e-05
2	0.8263	22.7135	3.4376	2.00e-05
3	0.8261	14.0016	3.4130	2.00e-05
4	0.8358	9.3994	3.6396	2.00e-05
5	0.8362	6.9198	3.9657	1.86e-05
6	0.8411	4.8009	4.3146	1.86e-05
7	0.8435	3.8334	4.3902	1.73e-05
8	0.8217	2.9609	4.8945	1.73e-05
9	0.8454	2.3790	4.6530	1.61e-05
10	0.8520	2.4006	4.2368	1.61e-05
11	0.8515	1.9010	4.5653	1.50e-05
12	0.8433	1.6242	5.0868	1.50e-05
13	0.8365	1.3760	5.3600	1.39e-05
14	0.8377	1.1662	5.1492	1.39e-05
15	0.8345	0.9457	5.4017	1.29e-05
16	0.8430	1.0117	5.1804	1.29e-05
17	0.8498	0.8583	5.5515	1.20e-05
18	0.8445	0.7061	5.5779	1.20e-05
19	0.8302	0.7815	5.9122	1.12e-05

Table A.2: mBERT seed 42 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7684	68.0075	4.2706	2.00e-05
2	0.8249	22.3122	3.5049	2.00e-05
3	0.8239	13.5103	3.6727	2.00e-05
4	0.8462	9.4838	3.5525	2.00e-05
5	0.8510	6.5424	3.9455	1.86e-05
6	0.8253	5.0036	4.5083	1.86e-05
7	0.8433	3.8627	4.2149	1.73e-05
8	0.8251	3.2255	4.7884	1.73e-05
9	0.8348	2.5549	4.7548	1.61e-05
10	0.8467	2.0466	4.9655	1.61e-05
11	0.8394	1.8670	4.6874	1.50e-05
12	0.8375	1.5565	5.2846	1.50e-05
13	0.8234	1.2027	5.8072	1.39e-05
14	0.8343	1.3031	5.8198	1.39e-05

Table A.3: mBERT seed 8 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7989	72.2627	3.9953	2.00e-05
2	0.7984	22.8389	3.5976	2.00e-05
3	0.8372	13.7829	3.3550	2.00e-05
4	0.8396	9.2690	3.6457	2.00e-05
5	0.8333	6.9039	3.9777	1.86e-05
6	0.8401	4.9983	4.0292	1.86e-05
7	0.8438	4.0289	4.2630	1.73e-05
8	0.8447	3.2377	4.5652	1.73e-05
9	0.8416	2.7318	4.8095	1.61e-05
10	0.8367	2.4836	4.9105	1.61e-05
11	0.8372	1.9560	5.0740	1.50e-05
12	0.8370	1.5267	5.0022	1.50e-05
13	0.8418	1.1764	5.3892	1.39e-05
14	0.8345	1.0659	5.3329	1.39e-05
15	0.8469	0.9984	5.5240	1.29e-05
16	0.8341	0.8448	5.9528	1.29e-05
17	0.8367	1.0515	5.7047	1.20e-05
18	0.8338	0.7948	5.8308	1.20e-05
19	0.8450	0.6735	5.7182	1.12e-05
20	0.8418	0.5532	5.8720	1.12e-05
21	0.8479	0.6422	6.0070	1.04e-05
22	0.8421	0.6116	5.6259	1.04e-05
23	0.8343	0.4988	5.8740	9.68e-06
24	0.8454	0.5097	5.9765	9.68e-06
25	0.8484	0.4689	6.1552	9.00e-06
26	0.8435	0.8369	6.2201	9.00e-06
27	0.8377	0.4423	6.2814	8.37e-06
28	0.8309	0.3489	6.5396	8.37e-06
29	0.8406	0.2836	6.2307	7.79e-06
30	0.8375	0.2672	6.4734	7.79e-06
31	0.8474	0.2943	6.2924	7.24e-06
32	0.8387	0.2428	6.5679	7.24e-06
33	0.8464	0.2464	6.4747	6.73e-06
34	0.8408	0.1814	6.4641	6.73e-06

Table A.4: mBERT seed 37 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7740	70.3983	4.2179	2.00e-05
2	0.8084	22.9184	3.6122	2.00e-05
3	0.8205	13.8833	3.5907	2.00e-05
4	0.8358	9.6895	3.5682	2.00e-05
5	0.8421	6.6674	4.0060	1.86e-05
6	0.8336	5.2350	4.1901	1.86e-05
7	0.8408	3.9585	4.4679	1.73e-05
8	0.8375	3.2135	4.5344	1.73e-05
9	0.8498	2.8077	4.4353	1.61e-05
10	0.8375	1.9541	4.8751	1.61e-05
11	0.8365	2.1582	4.9123	1.50e-05
12	0.8442	1.6182	4.9229	1.50e-05
13	0.8454	1.2320	5.1833	1.39e-05
14	0.8462	1.1042	5.3992	1.39e-05
15	0.8401	1.1980	5.4005	1.29e-05
16	0.8486	0.9297	5.2077	1.29e-05
17	0.8304	0.7452	5.7505	1.20e-05
18	0.8375	0.8193	6.1152	1.20e-05

Table A.5: mBERT seed 101 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7125	97.1865	5.1887	2.00e-05
2	0.8110	27.5390	3.8597	2.00e-05
3	0.8389	16.6439	3.2529	2.00e-05
4	0.8333	10.9792	3.4119	1.86e-05
5	0.8554	7.6425	3.3456	1.86e-05
6	0.8358	5.4948	3.6588	1.86e-05
7	0.8384	3.9299	3.7279	1.86e-05
8	0.8500	3.1927	3.7611	1.73e-05
9	0.8510	2.5959	3.8057	1.73e-05
10	0.8503	1.9686	4.1348	1.61e-05
11	0.8423	1.6832	4.4392	1.61e-05
12	0.8576	1.3368	4.2600	1.50e-05
13	0.8476	1.1130	4.6264	1.50e-05
14	0.8469	1.0656	4.6346	1.39e-05
15	0.8520	0.9986	4.4718	1.39e-05
16	0.8508	0.8883	4.4089	1.29e-05
17	0.8568	0.7255	4.7918	1.29e-05
18	0.8527	0.5299	5.1819	1.20e-05
19	0.8566	0.5347	4.8490	1.20e-05
20	0.8590	0.5068	4.9891	1.12e-05
21	0.8539	0.4271	5.0791	1.12e-05
22	0.8413	0.5532	5.3603	1.04e-05
23	0.8571	0.3779	5.0402	1.04e-05
24	0.8513	0.2764	5.2507	9.68e-06
25	0.8418	0.2740	5.6351	9.68e-06
26	0.8476	0.2740	5.6121	9.00e-06
27	0.8588	0.2990	5.4070	9.00e-06
28	0.8537	0.2783	5.5482	8.37e-06
29	0.8576	0.2628	5.4449	8.37e-06

Table A.6: NorBERT seed 101 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7406	92.6809	4.9456	2.00e-05
2	0.8089	27.4191	3.7120	2.00e-05
3	0.8256	15.6934	3.5074	2.00e-05
4	0.8178	10.2390	3.6628	1.86e-05
5	0.8384	6.9185	3.5390	1.86e-05
6	0.8430	5.0726	3.6656	1.73e-05
7	0.8467	3.5598	3.8283	1.73e-05
8	0.8442	2.6312	4.0496	1.61e-05
9	0.8413	2.2403	4.1726	1.61e-05
10	0.8396	1.9767	4.6201	1.50e-05
11	0.8413	1.5807	4.6397	1.50e-05
12	0.8452	1.6099	4.4286	1.39e-05
13	0.8471	1.2611	5.0316	1.39e-05
14	0.8459	1.1650	4.5043	1.29e-05
15	0.8462	1.0067	4.5983	1.29e-05
16	0.8440	0.6461	4.9729	1.20e-05
17	0.8614	0.6611	4.7159	1.20e-05
18	0.8532	0.5982	4.9827	1.12e-05
19	0.8496	0.5197	5.1715	1.12e-05
20	0.8464	0.5145	5.0018	1.04e-05
21	0.8515	0.3811	5.1245	1.04e-05
22	0.8479	0.4136	5.3384	9.68e-06
23	0.8377	0.4803	5.5377	9.68e-06
24	0.8491	0.3480	5.2279	9.00e-06
25	0.8517	0.3495	5.0548	9.00e-06
26	0.8399	0.3106	5.5673	8.37e-06

Table A.7: NorBERT seed 37 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7127	100.7491	5.2153	2.00e-05
2	0.8011	28.8864	3.6324	2.00e-05
3	0.8278	16.8526	3.4071	2.00e-05
4	0.8358	10.8933	3.4072	2.00e-05
5	0.8227	7.6063	3.5280	1.86e-05
6	0.8222	5.5975	3.9414	1.86e-05
7	0.8547	4.1314	3.7420	1.73e-05
8	0.8358	3.1115	4.0664	1.73e-05
9	0.8345	2.4497	4.3688	1.61e-05
10	0.8549	1.9965	3.9931	1.61e-05
11	0.8345	1.7865	4.5745	1.50e-05
12	0.8537	1.2779	4.4293	1.50e-05
13	0.8469	1.1582	4.7921	1.39e-05
14	0.8396	1.1292	5.0618	1.39e-05
15	0.8447	0.9246	4.7251	1.29e-05
16	0.8413	0.9059	4.5169	1.29e-05
17	0.8513	0.8402	4.8933	1.20e-05
18	0.8476	0.5657	4.9325	1.20e-05
19	0.8406	0.5977	5.1469	1.12e-05

Table A.8: NorBERT seed 42 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7306	92.5764	4.9081	2.00e-05
2	0.8115	26.7694	3.6770	2.00e-05
3	0.8154	15.7897	3.4303	2.00e-05
4	0.8224	10.2722	3.4626	2.00e-05
5	0.8222	6.8309	3.6136	2.00e-05
6	0.8430	5.1689	3.6690	1.86e-05
7	0.8249	4.0613	3.9235	1.86e-05
8	0.8450	3.0128	3.9144	1.73e-05
9	0.8525	2.4284	3.9068	1.73e-05
10	0.8464	1.9653	4.1265	1.61e-05
11	0.8445	1.9132	4.3648	1.61e-05
12	0.8408	1.3520	4.6774	1.50e-05
13	0.8493	0.9689	4.5865	1.50e-05
14	0.8539	1.0314	4.5684	1.39e-05
15	0.8423	0.9082	4.8960	1.39e-05
16	0.8505	0.6494	4.7209	1.29e-05
17	0.8450	0.5765	5.1580	1.29e-05
18	0.8341	0.4491	5.2917	1.20e-05
19	0.8411	0.5536	5.2807	1.20e-05
20	0.8343	0.5653	5.3686	1.12e-05
21	0.8256	0.4481	5.7361	1.12e-05
22	0.8469	0.3929	5.3151	1.04e-05
23	0.8394	0.3685	5.4390	1.04e-05
24	0.8551	0.3509	5.0535	9.68e-06
25	0.8527	0.3437	5.2040	9.68e-06
26	0.8486	0.3655	5.0941	9.00e-06
27	0.8593	0.2528	4.9960	9.00e-06
28	0.8469	0.2115	5.3336	8.37e-06
29	0.8505	0.2130	5.2463	8.37e-06
30	0.8549	0.2201	5.2072	7.79e-06
31	0.8394	0.2021	5.6657	7.79e-06
32	0.8442	0.2047	5.3811	7.24e-06
33	0.8440	0.1502	5.6674	7.24e-06
34	0.8459	0.1241	5.9175	6.73e-06
35	0.8539	0.1652	5.5680	6.73e-06
36	0.8554	0.1061	5.4760	6.26e-06

Table A.9: NorBERT seed 8 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.7636	89.5387	4.8734	2.00e-05
2	0.8074	26.2759	3.6314	2.00e-05
3	0.8537	15.7820	3.2470	2.00e-05
4	0.8081	10.5570	3.7625	2.00e-05
5	0.8464	7.2826	3.3526	1.86e-05
6	0.8515	5.2202	3.3007	1.86e-05
7	0.8440	3.9002	3.7005	1.73e-05
8	0.8360	3.0315	3.8433	1.73e-05
9	0.8423	2.2982	4.1437	1.61e-05
10	0.8481	2.1501	4.2607	1.61e-05
11	0.8491	1.6538	4.2675	1.50e-05
12	0.8433	1.3722	4.4822	1.50e-05

Table A.10: NorBERT seed 1024 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8377	52.2142	2.8407	2.00e-05
2	0.8622	12.3361	2.4389	2.00e-05
3	0.8631	6.2733	2.7640	2.00e-05
4	0.8714	3.4814	2.9224	2.00e-05
5	0.8651	2.2019	3.4047	1.86e-05
6	0.8663	1.5534	3.4323	1.86e-05
7	0.8651	1.0808	3.7757	1.73e-05
8	0.8762	0.8991	3.6880	1.73e-05
9	0.8653	0.8292	3.8373	1.61e-05
10	0.8772	0.7939	3.7952	1.61e-05
11	0.8728	0.5443	3.8610	1.50e-05
12	0.8590	0.4789	4.1652	1.50e-05
13	0.8672	0.3228	4.3434	1.39e-05
14	0.8709	0.2740	4.4650	1.39e-05
15	0.8716	0.2703	4.3906	1.29e-05
16	0.8568	0.3783	4.5439	1.29e-05
17	0.8554	0.2802	5.0246	1.20e-05
18	0.8590	0.2335	4.9764	1.20e-05
19	0.8609	0.3832	4.5612	1.12e-05

Table A.11: NorBERT2 seed 42 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8520	52.2087	2.6768	2.00e-05
2	0.8721	12.8177	2.3641	2.00e-05
3	0.8711	6.2395	2.5484	2.00e-05
4	0.8656	3.7737	3.0555	2.00e-05
5	0.8675	2.3021	3.3562	1.86e-05
6	0.8697	1.4955	3.4261	1.86e-05
7	0.8668	1.0214	3.8558	1.73e-05
8	0.8699	0.9928	3.8122	1.73e-05
9	0.8643	0.7999	4.1243	1.61e-05
10	0.8716	0.6768	4.1476	1.61e-05
11	0.8847	0.5901	3.8993	1.50e-05
12	0.8735	0.4748	4.0456	1.50e-05
13	0.8682	0.3153	4.3943	1.39e-05
14	0.8789	0.3179	4.5130	1.39e-05
15	0.8779	0.5663	4.1369	1.29e-05
16	0.8622	0.3134	4.3180	1.29e-05
17	0.8648	0.2249	4.8040	1.20e-05
18	0.8806	0.2848	4.4037	1.20e-05
19	0.8665	0.3333	4.3716	1.12e-05
20	0.8641	0.2501	4.7525	1.12e-05

Table A.12: NorBERT2 seed 8 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8319	50.3822	2.9576	2.00e-05
2	0.8677	12.2517	2.4116	2.00e-05
3	0.8607	6.0945	2.7742	2.00e-05
4	0.8750	3.4882	2.8794	1.86e-05
5	0.8680	2.0177	3.3829	1.86e-05
6	0.8665	1.3927	3.7446	1.73e-05
7	0.8665	1.0645	4.0039	1.73e-05
8	0.8726	0.9379	3.7415	1.61e-05
9	0.8801	0.7072	3.8398	1.61e-05
10	0.8614	0.6811	4.4174	1.50e-05
11	0.8675	0.5919	4.1718	1.50e-05
12	0.8605	0.4542	4.3496	1.39e-05
13	0.8634	0.3844	4.6252	1.39e-05
14	0.8617	0.4277	4.6908	1.29e-05
15	0.8602	0.3118	4.7844	1.29e-05
16	0.8672	0.3079	4.8100	1.20e-05
17	0.8607	0.3554	4.7319	1.20e-05
18	0.8709	0.2512	4.5213	1.12e-05

Table A.13: NorBERT2 seed 101 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8447	52.6635	2.7712	2.00e-05
2	0.8542	12.3776	2.6412	2.00e-05
3	0.8781	6.3086	2.6085	1.86e-05
4	0.8639	3.4005	3.1123	1.86e-05
5	0.8801	1.9722	3.0757	1.73e-05
6	0.8631	1.6063	3.5565	1.73e-05
7	0.8566	1.1563	4.0525	1.61e-05
8	0.8738	0.8527	3.9074	1.61e-05
9	0.8806	0.8713	3.7743	1.50e-05
10	0.8757	0.5881	4.0760	1.50e-05
11	0.8624	0.7394	4.4666	1.39e-05
12	0.8697	0.5583	4.3511	1.39e-05
13	0.8530	0.4381	4.6990	1.29e-05
14	0.8772	0.4068	4.2894	1.29e-05
15	0.8721	0.2895	4.6798	1.20e-05
16	0.8689	0.3396	4.6886	1.20e-05
17	0.8883	0.2581	4.4104	1.12e-05
18	0.8786	0.2127	4.8362	1.12e-05
19	0.8706	0.3138	4.5362	1.04e-05
20	0.8656	0.2165	4.9504	1.04e-05
21	0.8714	0.1650	4.9988	9.68e-06
22	0.8719	0.1040	4.9943	9.68e-06
23	0.8689	0.0793	5.2409	9.00e-06
24	0.8760	0.1522	4.8077	9.00e-06
25	0.8723	0.1962	4.9911	8.37e-06
26	0.8752	0.2069	4.6940	8.37e-06

Table A.14: NorBERT2 seed 1024 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8493	50.0674	2.8220	2.00e-05
2	0.8670	12.4641	2.4962	2.00e-05
3	0.8612	6.1594	2.7439	2.00e-05
4	0.8714	3.6004	2.9489	1.86e-05
5	0.8636	2.2588	3.3997	1.86e-05
6	0.8602	1.3531	3.6782	1.73e-05
7	0.8622	1.2065	3.7929	1.73e-05
8	0.8471	0.7857	4.5201	1.61e-05
9	0.8740	0.7741	4.0984	1.61e-05
10	0.8561	0.7548	4.4241	1.50e-05
11	0.8653	0.4965	4.3692	1.50e-05
12	0.8663	0.4109	4.2945	1.39e-05
13	0.8515	0.4611	4.5490	1.39e-05
14	0.8709	0.4617	4.3248	1.29e-05
15	0.8823	0.5046	4.0945	1.29e-05
16	0.8670	0.4823	4.3887	1.20e-05
17	0.8769	0.2523	4.1500	1.20e-05
18	0.8687	0.1715	4.6427	1.12e-05
19	0.8704	0.2317	4.6500	1.12e-05
20	0.8832	0.1865	4.2543	1.04e-05
21	0.8765	0.1585	4.6759	1.04e-05
22	0.8798	0.1408	4.7155	9.68e-06
23	0.8740	0.1230	4.7775	9.68e-06
24	0.8714	0.1652	4.8895	9.00e-06
25	0.8622	0.2276	5.0816	9.00e-06
26	0.8675	0.1574	5.2099	8.37e-06
27	0.8689	0.1374	5.0820	8.37e-06
28	0.8697	0.1169	5.0177	7.79e-06
29	0.8784	0.1387	4.7382	7.79e-06

Table A.15: NorBERT2 seed 37 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8464	52.0772	2.6354	2.00e-05
2	0.8682	13.0837	2.3323	2.00e-05
3	0.8857	7.2475	2.3209	2.00e-05
4	0.8781	4.8451	2.6476	2.00e-05
5	0.8898	3.4069	2.6461	1.86e-05
6	0.8997	2.4228	2.6254	1.86e-05
7	0.8883	1.7802	3.0084	1.73e-05
8	0.8912	1.5040	3.0689	1.73e-05
9	0.8883	1.2387	3.2360	1.61e-05
10	0.8910	0.9481	3.0965	1.61e-05
11	0.8874	0.7708	3.4764	1.61e-05
12	0.8852	0.8753	3.3880	1.50e-05
13	0.8706	0.6192	3.9092	1.50e-05
14	0.8939	0.7180	3.4948	1.39e-05
15	0.8750	0.5822	3.9152	1.39e-05

Table A.16: NB-BERT seed 37 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8343	53.9982	2.7768	2.00e-05
2	0.8811	13.2062	2.2109	2.00e-05
3	0.8941	7.3976	2.1899	2.00e-05
4	0.8854	4.5101	2.5016	2.00e-05
5	0.8983	3.2253	2.6415	1.86e-05
6	0.8844	2.3198	2.7323	1.86e-05
7	0.8895	1.7363	3.1111	1.73e-05
8	0.8794	1.4158	3.3251	1.73e-05
9	0.8774	1.2233	3.5711	1.61e-05
10	0.8941	0.9818	3.3455	1.61e-05
11	0.8801	0.9547	3.4682	1.50e-05
12	0.8781	0.7722	3.8062	1.50e-05
13	0.8733	0.8162	3.6838	1.39e-05
14	0.8905	0.7012	3.2753	1.39e-05

Table A.17: NB-BERT seed 101 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8408	50.9818	2.7669	2.00e-05
2	0.8769	12.6998	2.2829	2.00e-05
3	0.8857	7.3607	2.3898	2.00e-05
4	0.8903	4.7718	2.4702	2.00e-05
5	0.8794	3.5760	2.7959	2.00e-05
6	0.8898	2.3370	2.9843	1.86e-05
7	0.8854	1.9192	2.9241	1.86e-05
8	0.8876	1.4987	3.2120	1.73e-05
9	0.8857	1.1741	3.2532	1.73e-05
10	0.8961	1.0172	3.3765	1.61e-05
11	0.8900	0.8381	3.5328	1.61e-05
12	0.8927	0.8658	3.3764	1.50e-05
13	0.8973	0.7424	3.3678	1.50e-05
14	0.8956	0.6069	3.4169	1.39e-05
15	0.9029	0.4315	3.5408	1.39e-05
16	0.8905	0.5421	3.6905	1.29e-05
17	0.8917	0.4682	3.6455	1.29e-05
18	0.8903	0.4652	3.8390	1.20e-05
19	0.8968	0.2920	3.7444	1.20e-05
20	0.8765	0.3375	4.0788	1.12e-05
21	0.8932	0.3406	4.0165	1.12e-05
22	0.8910	0.3301	4.0579	1.04e-05
23	0.8951	0.2519	4.0468	1.04e-05
24	0.8895	0.2588	4.2287	9.68e-06

Table A.18: NB-BERT seed 8 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8563	50.7012	2.5224	2.00e-05
2	0.8685	12.8018	2.3840	2.00e-05
3	0.8689	7.2362	2.5616	2.00e-05
4	0.8818	4.6250	2.4641	2.00e-05
5	0.8932	3.2888	2.5852	2.00e-05
6	0.8903	2.3128	2.6598	1.86e-05
7	0.8949	1.7152	2.9561	1.86e-05
8	0.8735	1.5033	3.5905	1.73e-05
9	0.8966	0.9791	3.1546	1.73e-05
10	0.8903	1.0002	3.2425	1.61e-05
11	0.8886	0.7639	3.3270	1.61e-05
12	0.9062	0.8803	2.9820	1.50e-05
13	0.8774	0.7128	3.7163	1.50e-05
14	0.8953	0.7542	3.3716	1.39e-05
15	0.8990	0.5989	3.3048	1.39e-05
16	0.8951	0.3917	3.4243	1.39e-05
17	0.9004	0.6041	3.4581	1.39e-05
18	0.8939	0.5382	3.3386	1.39e-05
19	0.8847	0.5838	3.7502	1.39e-05
20	0.8876	0.4865	3.6501	1.29e-05
21	0.8815	0.2173	4.1599	1.29e-05

Table A.19: NB-BERT seed 42 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.8469	55.3110	2.8197	2.00e-05
2	0.8835	12.8683	2.3170	2.00e-05
3	0.8866	7.4436	2.2853	2.00e-05
4	0.8871	4.6853	2.6367	2.00e-05
5	0.8781	3.2783	2.9055	1.86e-05
6	0.8956	2.1381	2.7540	1.86e-05
7	0.8927	1.9683	2.8259	1.73e-05
8	0.8958	1.3189	3.1300	1.73e-05
9	0.8815	1.1802	3.4506	1.61e-05
10	0.8980	1.1192	3.2311	1.61e-05
11	0.8895	0.7781	3.2925	1.50e-05
12	0.8915	0.7062	3.4681	1.50e-05
13	0.8755	0.6731	3.8036	1.39e-05
14	0.8920	0.5437	3.4443	1.39e-05
15	0.8934	0.5271	3.4827	1.29e-05
16	0.8907	0.6766	3.5814	1.29e-05
17	0.8900	0.3999	3.7259	1.20e-05
18	0.8905	0.5627	3.4032	1.20e-05
19	0.8837	0.3589	3.9234	1.12e-05

Table A.20: NB-BERT seed 1024 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.9031	38.1994	2.0490	2.00e-05
2	0.9033	8.8223	1.7134	2.00e-05
3	0.9227	5.3342	1.6697	2.00e-05
4	0.9193	3.4882	1.7666	1.86e-05
5	0.9055	2.4416	2.2019	1.86e-05
6	0.9118	1.8235	2.2214	1.73e-05
7	0.9230	1.4112	2.0177	1.73e-05
8	0.9251	1.3939	2.0775	1.61e-05
9	0.9125	1.1165	2.4483	1.61e-05
10	0.9210	0.9270	2.5080	1.50e-05
11	0.9201	0.7225	2.3101	1.50e-05
12	0.9152	0.6342	2.5197	1.39e-05
13	0.9169	0.7638	2.6066	1.39e-05
14	0.9159	0.6130	2.6678	1.29e-05
15	0.9220	0.4806	2.6560	1.29e-05
16	0.9113	0.4423	2.9390	1.20e-05
17	0.9089	0.5022	2.7538	1.20e-05

Table A.21: Scandi ner seed 1024 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.9138	37.8601	1.8551	2.00e-05
2	0.9184	9.0088	1.6223	2.00e-05
3	0.9242	5.0619	1.5922	2.00e-05
4	0.9118	3.4869	1.9371	1.86e-05
5	0.9111	2.1817	2.0293	1.86e-05
6	0.9147	1.8632	2.2819	1.86e-05
7	0.9152	1.4704	2.3048	1.73e-05
8	0.9181	1.4931	2.3081	1.73e-05
9	0.9179	1.2779	2.4468	1.61e-05
10	0.9121	1.1221	2.4660	1.61e-05
11	0.9188	0.7967	2.4408	1.50e-05
12	0.9002	1.0178	2.6835	1.50e-05

Table A.22: Scandi ner seed 37 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.9014	37.3559	2.0919	2.00e-05
2	0.9172	8.8492	1.7048	2.00e-05
3	0.9113	5.0458	1.7632	2.00e-05
4	0.9113	3.3763	1.9889	1.86e-05
5	0.9157	2.3441	2.1061	1.86e-05
6	0.9278	1.6974	2.0021	1.73e-05
7	0.9067	1.5295	2.4783	1.73e-05
8	0.9162	1.5023	2.3009	1.61e-05
9	0.9152	0.8927	2.6275	1.61e-05
10	0.8910	1.0230	2.8581	1.50e-05
11	0.9024	0.7763	2.9228	1.50e-05
12	0.9210	0.9232	2.5604	1.39e-05
13	0.9116	0.5362	2.6832	1.39e-05
14	0.9227	0.4502	2.5388	1.29e-05
15	0.9096	0.3985	2.7618	1.29e-05

Table A.23: Scandi ner seed 42 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.9084	35.2245	1.7453	2.00e-05
2	0.9208	8.5002	1.6040	2.00e-05
3	0.9186	4.9390	1.8040	2.00e-05
4	0.9225	3.2814	1.8502	1.86e-05
5	0.9184	3.2168	1.9225	1.86e-05
6	0.9167	1.7283	2.2564	1.73e-05
7	0.9232	1.3425	2.1985	1.73e-05
8	0.9169	1.1804	2.2560	1.61e-05
9	0.9058	1.0327	2.5422	1.61e-05
10	0.9121	1.0039	2.4887	1.50e-05
11	0.9130	0.9773	2.5562	1.50e-05
12	0.9077	0.5967	2.7443	1.39e-05
13	0.9186	0.5914	2.4214	1.39e-05
14	0.9048	0.7891	2.8587	1.29e-05
15	0.9172	0.6206	2.6314	1.29e-05
16	0.9014	0.4510	2.8386	1.20e-05

Table A.24: Scandi ner seed 8 training metrics

Epoch	Validation Recall	Train Loss	Validation Loss	Learning Rate
1	0.9021	36.9761	1.8941	2.00e-05
2	0.9070	8.8546	1.7946	2.00e-05
3	0.9193	4.9838	1.8070	1.86e-05
4	0.9188	3.3755	1.8770	1.86e-05
5	0.9155	2.5897	1.9740	1.86e-05
6	0.9009	1.9625	2.2673	1.86e-05
7	0.9232	1.6097	2.1114	1.86e-05
8	0.9251	1.3243	2.0473	1.86e-05
9	0.9099	1.1051	2.4391	1.73e-05
10	0.9184	1.1741	2.3794	1.73e-05
11	0.9205	1.0012	2.3404	1.61e-05
12	0.9164	0.7091	2.5236	1.61e-05
13	0.9113	0.5281	2.7371	1.50e-05
14	0.9196	0.8359	2.4104	1.50e-05
15	0.9111	0.6241	2.6469	1.39e-05
16	0.9215	0.4901	2.5247	1.39e-05
17	0.9174	0.3575	2.6027	1.29e-05

Table A.25: Scandi ner seed 101 training metrics

Appendix B

Extended Results

Appendix B provides additional details regarding BERT model results that where not highlighted in Chapter 6, *Results and Discussion*. Chapter 6 highlight the best models, whereas this appendix provides supplementary material.

This includes:

- Tabular results for named entity recognition without considering token boundary averaged over the five used seeds for the models that where not included in Chapter 6. These include results for both NorNE and the annotated NPSC sample dataset.
- Tabular results for named entity recognition with strict match criteria, averaged over the five used seeds for the models that where not included in Chapter 6. These include results for both NorNE and the annotated NPSC sample dataset.
- Confusion matrix for Ner-Scandi's NorNE dataset results that has not been normalized.
- Confusion matrix for NB-BERT's NPSC sample dataset results that has not been normalized.

NE Type	Precision	Recall	F1	Support
B-PER	88.70 (±1.95)	90.99 (±0.92)	89.81 (±0.86)	961
I-PER	97.84 (±0.54)	94.47 (±1.53)	96.12 (±0.65)	510
B-ORG	77.35 (±2.99)	78.35 (±1.93)	77.78 (±1.16)	521
I-ORG	72.26 (±1.97)	86.84 (±1.49)	78.86 (±1.41)	248
B-GPE_LOC	87.99 (±1.50)	88.54 (±1.21)	88.26 (±1.09)	428
I-GPE_LOC	93.67 (±1.79)	87.32 (±2.42)	90.36 (±1.65)	79
B-LOC	69.30 (±4.60)	66.92 (±2.67)	67.92 (±1.61)	185
I-LOC	56.94 (±7.54)	85.07 (±3.45)	67.89 (±5.70)	85
B-PROD	40.46 (±3.05)	52.66 (±8.05)	45.27 (±2.56)	131
I-PROD	50.63 (±3.38)	65.80 (±5.57)	57.02 (±2.54)	126
B-DRV	70.77 (±2.35)	72.70 (±6.55)	71.47 (±2.66)	78
I-DRV	43.08 (±3.77)	55.19 (±8.79)	47.93 (±3.23)	13
B-GPE_ORG	72.46 (±4.57)	67.38 (±4.82)	69.74 (±3.92)	61
I-GPE_ORG	94.29 (±11.43)	66.06 (±9.37)	77.11 (±8.73)	7
B-EVT	24.29 (±15.39)	27.04 (±17.62)	24.32 (±14.54)	14
I-EVT	10 (±12.25)	4.86 (±6.10)	6.49 (±8.05)	4
B-MISC	1.43 (±2.86)	1.54 (±3.08)	1.48 (±2.96)	14
I-MISC	0 (±0)	0 (±0)	0 (±0)	3
Micro avg.	80.54 (±1.03)	83.84 (±1.11)	82.15 (±0.50)	3468

Table B.1: mBERT NER results without token span evaluation for the NorNE test dataset

NE Type	Precision	Recall	F1	Support
B-PER	79.53 (±3.42)	94.02 (±1.85)	86.11 (±1.85)	43
I-PER	91 (±2)	97.04 (±3.86)	93.84 (±1.10)	20
B-ORG	88.31 (±1.16)	80.61 (±1.70)	84.27 (±0.95)	77
I-ORG	78.18 (±4.45)	77.53 (±7.01)	77.56 (±3.30)	11
B-GPE_LOC	85 (±1.43)	82.96 (±4.90)	83.88 (±2.50)	28
I-GPE_LOC	66.67 (±0)	75.43 (±9.14)	70.49 (±4.48)	6
B-LOC	100 (±0)	33.71 (±3.64)	50.32 (±4.03)	2
I-LOC	0	0	0	0
B-PROD	80 (±0)	56.22 (±14.93)	64.95 (±9.95)	5
I-PROD	0	0	0	0
B-DRV	33.33 (±29.81)	60 (±48.99)	42 (±36)	3
I-DRV	0	0	0	0
B-GPE_ORG	71 (±10.20)	86.48 (±4.53)	77.67 (±7.41)	20
I-GPE_ORG	0	0	0	0
B-EVT	0	0	0	0
I-EVT	0	0	0	0
B-MISC	0 (±0)	0 (±0)	0 (±0)	1
I-MISC	0	0	0	0
Micro avg.	82.41 (±1.10)	81.38 (±1.26)	81.88 (±0.67)	216

Table B.2: mBERT NER results without token span evaluation for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
PER	87.80 (±1.87)	90.07 (±0.99)	88.91 (±0.83)	961
LOC	68.11 (±4.52)	65.81 (±3.37)	66.77 (±2.07)	185
ORG	74.97 (±2.79)	75.94 (±1.81)	75.39 (±0.91)	521
GPE_LOC	87.80 (±1.49)	88.35 (±1.26)	88.07 (±1.11)	428
GPE_ORG	72.13 (±4.64)	67.05 (±4.53)	69.41 (±3.77)	61
PROD	38.32 (±2.87)	49.87 (±7.58)	42.87 (±2.26)	131
EVT	22.86 (±14.57)	24.82 (±14.26)	22.59 (±12.53)	14
DRV	70.51 (±2.15)	72.45 (±6.62)	71.21 (±2.64)	78
MISC	1.43 (±2.86)	1.54 (±3.08)	1.48 (±2.96)	14
micro avg	78.93 (±1.01)	81.06 (±1.25)	79.97 (±0.41)	2393

Table B.3: mBERT Strict Match evaluation for the NorNE test dataset

NE Type	Precision	Recall	F1	Support
PER	78.60 (±4.00)	92.91 (±2.68)	85.10 (±2.78)	43
LOC	100.00 (±0.00)	33.71 (±3.64)	50.32 (±4.03)	2
ORG	86.75 (±1.51)	79.19 (±1.93)	82.78 (±1.37)	77
GPE_LOC	80.71 (±1.75)	78.85 (±5.74)	79.68 (±3.53)	28
GPE_ORG	71.00 (±10.20)	86.48 (±4.53)	77.67 (±7.41)	20
PROD	80.00 (±0.00)	56.22 (±14.93)	64.95 (±9.95)	5
EVT	0	0	0	0
DRV	33.33 (±29.81)	60.00 (±48.99)	42.00 (±36.00)	3
MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	1
micro avg	80.67 (±1.15)	79.90 (±1.43)	80.27 (±0.87)	179

Table B.4: mBERT Strict Match evaluation for the NPSC sample

NE Type	Precision	Recall	F1	Support
B-PER	94.48 (±1.03)	93.26 (±0.92)	93.86 (±0.53)	961
I-PER	97.18 (±0.56)	95.57 (±0.68)	96.36 (±0.19)	510
B-ORG	81.65 (±1.73)	79.54 (±3.77)	80.49 (±1.45)	521
I-ORG	71.21 (±3.31)	79.89 (±6.86)	74.97 (±1.93)	248
B-GPE_LOC	90.65 (±0.87)	86.06 (±2.43)	88.28 (±1.48)	428
I-GPE_LOC	94.94 (±1.79)	87.35 (±3.35)	90.94 (±1.83)	79
B-LOC	63.78 (±4.24)	67.39 (±3.50)	65.53 (±3.84)	185
I-LOC	56.94 (±8.63)	84.38 (±1.31)	67.59 (±6.22)	85
B-PROD	48.70 (±6.38)	50.03 (±6.02)	48.65 (±2.10)	131
I-PROD	38.41 (±2.68)	61.56 (±7.80)	46.90 (±1.55)	126
B-DRV	71.54 (±2.05)	64.44 (±3.98)	67.77 (±2.89)	78
I-DRV	49.23 (±16.57)	49.26 (±5.32)	47.21 (±9.04)	13
B-GPE_ORG	68.52 (±3.65)	63.21 (±3.69)	65.64 (±2.51)	61
I-GPE_ORG	74.29 (±38.76)	62.90 (±31.87)	67.95 (±34.75)	7
B-EVT	17.14 (±7.28)	31.57 (±8.49)	21.54 (±8.11)	14
I-EVT	20 (±29.15)	11.67 (±14.53)	14.23 (±18.67)	4
B-MISC	30 (±24.91)	50.64 (±41.42)	37.49 (±30.76)	14
I-MISC	0 (±0)	0 (±0)	0 (±0)	3
Micro avg.	82.58 (±0.58)	83.69 (±1.98)	83.12 (±1.10)	3468

Table B.5: NorBERT NER results without token span evaluation for the NorNE test dataset

NE Type	Precision	Recall	F1	Support
B-PER	88.37 (±1.47)	91.60 (±5.22)	89.90 (±3.10)	43
I-PER	96 (±3.74)	100 (±0)	97.92 (±1.96)	20
B-ORG	86.75 (±1.51)	79.22 (±4.49)	82.74 (±2.54)	77
I-ORG	81.82	70.67 (±13.37)	75.16 (±7.67)	11
B-GPE_LOC	87.86 (±2.86)	84.91 (±2.22)	86.31 (±1.34)	28
I-GPE_LOC	66.67 (±0)	80 (±0)	72.73(±0)	6
B-LOC	90 (±20)	31.10 (±5.18)	45.87 (±7.81)	2
I-LOC	0	0	0	0
B-PROD	80 (±12.65)	46.89 (±4.06)	58.65 (±4.90)	5
I-PROD	0	0	0	0
B-DRV	66.67 (±21.08)	55.24 (±9.69)	58.10 (±9.79)	3
I-DRV	0	0	0	0
B-GPE_ORG	62 (±15.68)	86.86 (±9.98)	71.60 (±12.87)	20
I-GPE_ORG	0	0	0	0
B-EVT	0	0	0	0
I-EVT	0	0	0	0
B-MISC	0 (±0)	0 (±0)	0 (±0)	1
I-MISC	0	0	0	0
Micro avg.	84.17 (±2.08)	79.43 (±2.15)	81.71 (±1.74)	216.00 (±0.00)

Table B.6: NorBERT NER results without token span evaluation for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
PER	94.05 (±0.99)	92.83 (±0.94)	93.43 (±0.51)	961
LOC	62.27 (±4.75)	65.77 (±3.95)	63.96 (±4.34)	185
ORG	79.58 (±1.79)	77.51 (±3.64)	78.45 (±1.44)	521
GPE_LOC	90.65 (±0.87)	86.06 (±2.43)	88.28 (±1.48)	428
GPE_ORG	66.89 (±3.50)	61.91 (±6.01)	64.19 (±4.48)	61
PROD	45.65 (±6.39)	46.79 (±5.19)	45.54 (±1.46)	131
EVT	17.14 (±7.28)	31.57 (±8.49)	21.54 (±8.11)	14
DRV	70.77 (±2.49)	63.74 (±4.02)	67.04 (±3.08)	78
MISC	25.71 (±21.48)	43.29 (±35.38)	32.10 (±26.43)	14
micro avg	82.88 (±0.71)	81.53 (±2.11)	82.19 (±1.26)	2393

Table B.7: NorBERT Strict Match evaluation for the NorNE test dataset

NE Type	Precision	Recall	F1	Support
PER	86.98 (±2.37)	90.18 (±5.89)	88.50 (±3.88)	43
LOC	90.00 (±20.00)	31.10 (±5.18)	45.87 (±7.81)	2
ORG	83.90 (±1.56)	76.67 (±5.35	) 80.05 (±3.44)	77
GPE_LOC	82.14 (±2.26)	79.40 (±2.16)	80.70 (±0.99)	28
GPE_ORG	60.00 (±17.03)	84.00 (±13.40)	69.24 (±14.84)	20
PROD	80.00 (±12.65)	46.89 (±4.06)	58.65 (±4.90)	5
EVT	0	0	0	0
DRV	66.67 (±21.08)	55.24 (±9.69)	58.10 (±9.79)	3
MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	1
micro avg	80.89 (±2.19)	77.24 (±2.32)	79.00 (±1.74)	179

Table B.8: NorBERT Strict Match the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
B-PER	93.09 (±1.25)	93.20 (±0.80)	93.14 (±0.64)	961
I-PER	97.37 (±0.70)	94.12 (±1.63)	95.71 (±0.73)	510
B-ORG	84.41 (±1.91)	81.56 (±1.57)	82.93 (±0.54)	521
I-ORG	73.87 (±3.33)	87.79 (±2.93)	80.12 (±1.01)	248
B-GPE_LOC	91.87 (±0.73)	89.50 (±0.97)	90.66 (±0.57)	428
I-GPE_LOC	93.92 (±1.68)	90.11 (±1.71)	91.95 (±0.32)	79
B-LOC	77.19 (±1.38)	73.11 (±1.29)	75.08 (±0.80)	185
I-LOC	74.12 (±6.78)	88.65 (±4.02)	80.44 (±3.01)	85
B-PROD	57.56 (±3.08)	59.14 (±4.85)	58.14 (±2.15)	131
I-PROD	36.83 (±3.84)	62.03 (±4.88)	46.10 (±3.67)	126
B-DRV	78.21 (±1.62)	71.09 (±3.54)	74.43 (±2.21)	78
I-DRV	60 (±7.54)	47.74 (±7.24)	52.86 (±6.57)	13
B-GPE_ORG	68.52 (±3.34)	79.55 (±3.84)	73.59 (±3.20)	61
I-GPE_ORG	100 (±0)	78.56 (±7.84)	87.77 (±4.91)	7
B-EVT	42.86 (±18.07)	48.02 (±15.02)	44.79 (±16.86)	14
I-EVT	30 (±29.15)	14.60 (±14.03)	19.58 (±18.83)	4
B-MISC	77.14 (±5.35)	84.90 (±5.45)	80.57 (±2.78)	14
I-MISC	0 (±0)	0 (±0)	0 (±0)	3
Micro avg.	84.91 (±0.47)	86 (±0.34)	85.45 (±0.23)	3468

Table B.9: NorBERT2 NER results without token span evaluation for the NorNE dataset

NE Type	Precision	Recall	F1	Support
B-PER	84.65 (±3.15)	90.11 (±2.14)	87.27 (±2.31)	43
I-PER	100 (±0)	100 (±0)	100 (±0)	20
B-ORG	90.13 (±2.41)	78.47 (±2.66)	83.83 (±1.27)	77
I-ORG	85.45 (±7.27)	69.43 (±8.28)	76.03 (±4.14)	11
B-GPE_LOC	87.14 (±1.75)	77.11 (±6.15)	81.72 (±3.83)	28
I-GPE_LOC	66.67	92 (±9.80)	77.09 (±3.56)	6
B-LOC	100 (±0)	32.76 (±4.20)	49.21 (±4.68)	2
I-LOC	0	0	0	0
B-PROD	80 (±0)	59.52 (±6.39)	68.07 (±4.24)	5
I-PROD	0	0	0	0
B-DRV	13.33 (±16.33)	16.67 (±21.08)	14.67 (±18.09)	3
I-DRV	0	0	0	0
B-GPE_ORG	45 (±8.94)	83.01 (±7.13)	58.06 (±8.72)	20
I-GPE_ORG	0	0	0	0
B-EVT	0	0	0	0
I-EVT	0	0	0	0
B-MISC	0 (±0)	0 (±0)	0 (±0)	1
I-MISC	0	0	0	0
Micro avg.	82.87 (±1.78)	78.79 (±0.48)	80.77 (±0.81)	216.00 (±0.00)

Table B.10: NorBERT2 NER results without token span evaluation for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
PER	92.55 (±0.92)	92.66 (±0.96)	92.60 (±0.45)	961
LOC	74.81 (±1.51)	70.86 (±1.75)	72.77 (±1.28)	185
ORG	82.19 (±1.98)	79.41 (±1.50)	80.74 (±0.67)	521
GPE_LOC	91.68 (±0.76)	89.31 (±0.90)	90.48 (±0.53)	428
GPE_ORG	68.52 (±3.34)	79.55 (±3.84)	73.59 (±3.20)	61
PROD	53.28 (±2.53)	54.85 (±5.43)	53.87 (±2.70)	131
EVT	37.14 (±13.85)	42.32 (±11.68)	39.10 (±13.06)	14
DRV	76.15 (±1.03)	69.27 (±4.14)	72.50 (±2.62)	78
MISC	61.43 (±3.50)	67.97 (±7.77)	64.33 (±4.58)	14
micro avg	84.96 (±0.69)	83.80 (±0.62)	84.38 (±0.37)	2393

Table B.11: NorBERT2 strict match NorNE

NE Type	Precision	Recall	F1	Support
PER	84.65 (±3.15)	90.11 (±2.14)	87.27 (±2.31)	43
LOC	100.00 (±0.00)	32.76 (±4.20)	49.21 (±4.68)	2
ORG	86.49 (±1.76)	75.33 (±3.15)	80.47 (±1.65)	77
GPE_LOC	84.29 (±3.64)	74.52 (±5.83)	79.01 (±4.13)	28
GPE_ORG	45.00 (±8.94)	83.01 (±7.13)	58.06 (±8.72)	20
PROD	80.00 (±0.00)	59.52 (±6.39)	68.07 (±4.24)	5
EVT	0	0	0	0
DRV	13.33 (±16.33)	16.67 (±21.08)	14.67 (±18.09)	3
MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	1
micro avg	79.33 (±1.69)	75.94 (±0.52)	77.59 (±0.73)	179

Table B.12: NorBERT2 Strict Match for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
B-PER	97.11 (±0.82)	94.77 (±0.75)	95.92 (±0.34)	961
I-PER	97.96 (±0.47)	95.69 (±1.09)	96.81 (±0.54)	510
B-ORG	83.57 (±1.42)	84.84 (±1.75)	84.17 (±0.49)	521
I-ORG	77.34 (±2.69)	87.36 (±2.68)	81.96 (±0.38)	248
B-GPE_LOC	92.38 (±1.70)	90.21 (±0.82)	91.27 (±0.46)	428
I-GPE_LOC	94.18 (±2.35)	89.90 (±1.50)	91.96 (±1.08)	79
B-LOC	78.27 (±2.35)	75.51 (±1.08)	76.84 (±1.19)	185
I-LOC	71.53 (±6.67)	89.46 (±4.23)	79.12 (±2.77)	85
B-PROD	58.93 (±3.87)	57.59 (±2.23)	58.14 (±1.64)	131
I-PROD	48.10 (±5.40)	61.29 (±5.01)	53.43 (±2.47)	126
B-DRV	77.69 (±3.94)	75.11 (±3.66)	76.23 (±1.66)	78
I-DRV	43.08 (±6.15)	49.48 (±11.19)	45.80 (±7.63)	13
B-GPE_ORG	76.07 (±3.82)	75.77 (±4.31)	75.83 (±3.06)	61
I-GPE_ORG	100.00 (±0.00)	70.32 (±18.01)	81.30 (±12.09)	7
B-EVT	62.86 (±9.48)	51.53 (±6.26)	56.00 (±4.29)	14
I-EVT	55.00 (±24.49)	39.19 (±22.65)	43.29 (±22.01)	4
B-MISC	47.14 (±21.95)	92.36 (±9.37)	58.29 (±20.91)	14
I-MISC	26.67 (±24.94)	53.33 (±45.22)	33.33 (±27.89)	3
Micro avg.	86.83 (±0.80)	87.19 (±0.45)	87.00 (±0.30)	3468

Table B.13: NB-BERT NER results without token span evaluation for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
PER	96.88 (±0.85)	94.55 (±0.79)	95.69 (±0.41)	961
LOC	77.19 (±1.76)	74.49 (±1.27)	75.79 (±0.74)	185
ORG	81.46 (±1.31)	82.70 (±2.02)	82.05 (±0.85)	521
GPE_LOC	92.38 (±1.70)	90.21 (±0.82)	91.27 (±0.46)	428
GPE_ORG	76.07 (±3.82)	75.77 (±4.31)	75.83 (±3.06)	61
PROD	55.57 (±3.29)	54.34 (±2.34)	54.84 (±1.39)	131
EVT	62.86 (±9.48)	51.53 (±6.26)	56.00 (±4.29)	14
DRV	75.38 (±3.57)	72.89 (±3.61)	73.97 (±1.38)	78
MISC	40.00 (±21.95)	71.55 (±12.81)	48.63 (±22.65)	14
micro avg	87.17 (±0.75)	85.84 (±0.55)	86.50 (±0.34)	2393

Table B.14: NB-BERT Strict Match for the NorNE test dataset

NE Type	Precision	Recall	F1	Support
B-PER	92.56 (±2.71)	93.46 (±2.17)	92.98 (±1.95)	43.00 (±0.00)
I-PER	100.00 (±0.00)	99.05 (±1.90)	99.51 (±0.98)	20.00 (±0.00)
B-ORG	88.31 (±1.84)	78.02 (±2.23)	82.83 (±1.87)	77.00 (±0.00)
I-ORG	98.18 (±3.64)	69.12 (±10.01)	80.54 (±5.91)	11.00 (±0.00)
B-GPE_LOC	88.57 (±1.43)	80.06 (±1.76)	84.07 (±0.64)	28.00 (±0.00)
I-GPE_LOC	66.67 (±0.00)	84.00 (±8.00)	74.18 (±2.91)	6.00 (±0.00)
B-LOC	100.00 (±0.00)	28.33 (±4.08)	44.00 (±4.90)	2.00 (±0.00)
I-LOC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
B-PROD	84.00 (±8.00)	60.29 (±12.89)	69.59 (±9.92)	5.00 (±0.00)
I-PROD	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
B-DRV	33.33 (±0.00)	100.00 (±0.00)	50.00 (±0.00)	3.00 (±0.00)
I-DRV	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
B-GPE_ORG	73.00 (±4.00)	92.92 (±5.37)	81.55 (±2.19)	20.00 (±0.00)
I-GPE_ORG	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
B-EVT	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
I-EVT	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
B-MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	1.00 (±0.00)
I-MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)
Micro avg.	87.59 (±0.80)	80.18 (±1.34)	83.72 (±1.00)	216.00 (±0.00)

Table B.15: Ner-Scandi NER results without token span evaluation for the annotated NPSC sample dataset

NE Type	Precision	Recall	F1	Support
PER	92.56 (±2.71)	93.46 (±2.17)	92.98 (±1.95)	43
LOC	100.00 (±0.00)	28.33 (±4.08)	44.00 (±4.90)	2
ORG	84.94 (±1.76)	75.03 (±2.07)	79.67 (±1.75)	77
GPE_LOC	83.57 (±1.75)	75.56 (±2.58)	79.34 (±1.75)	28
GPE_ORG	73.00 (±4.00)	92.92 (±5.37)	81.55 (±2.19)	20
PROD	84.00 (±8.00)	60.29 (±12.89)	69.59 (±9.92)	5
EVT	0	0	0	0
DRV	33.33 (±0.00)	100.00 (±0.00)	50.00 (±0.00)	3
MISC	0.00 (±0.00)	0.00 (±0.00)	0.00 (±0.00)	1
micro avg	84.02 (±1.25)	78.35 (±1.68)	81.08 (±1.33)	179

Table B.16: Ner-Scandi Strict Match for the annotated NPSC sample dataset

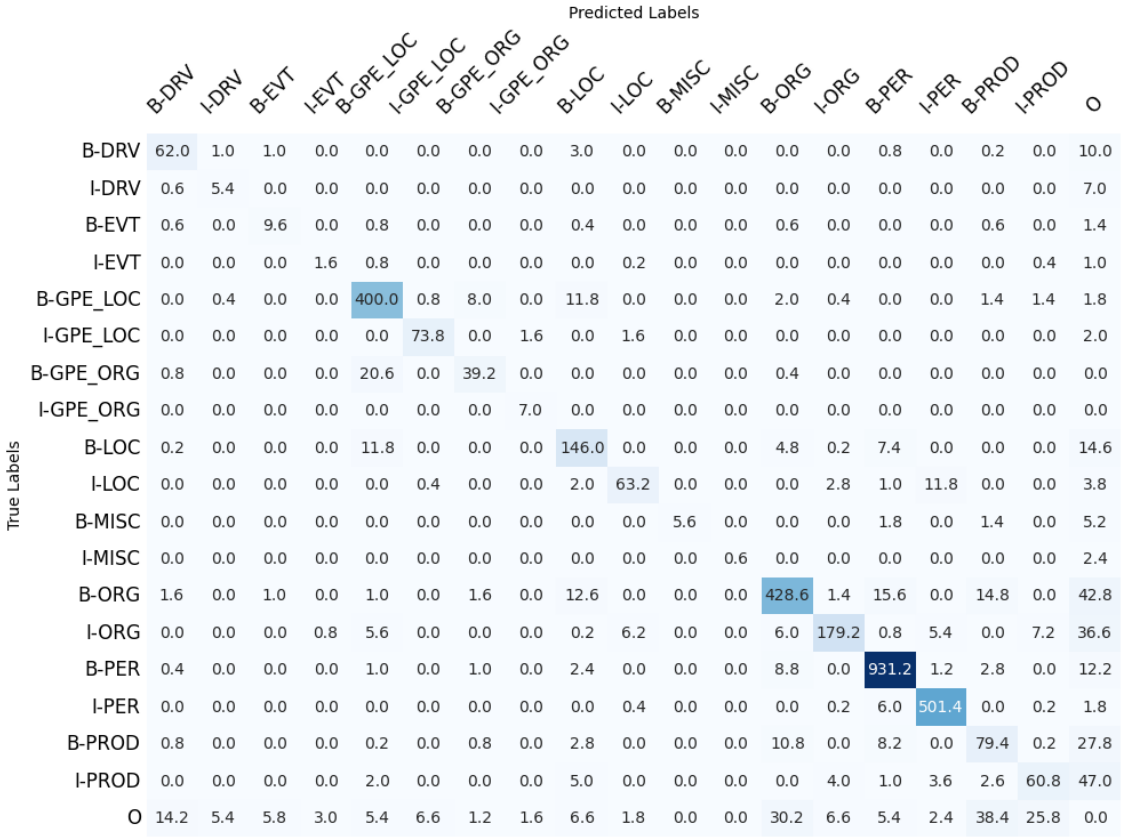


Figure B.1: NorNE results for Ner-Scandi combined average confusion matrix at token level classification

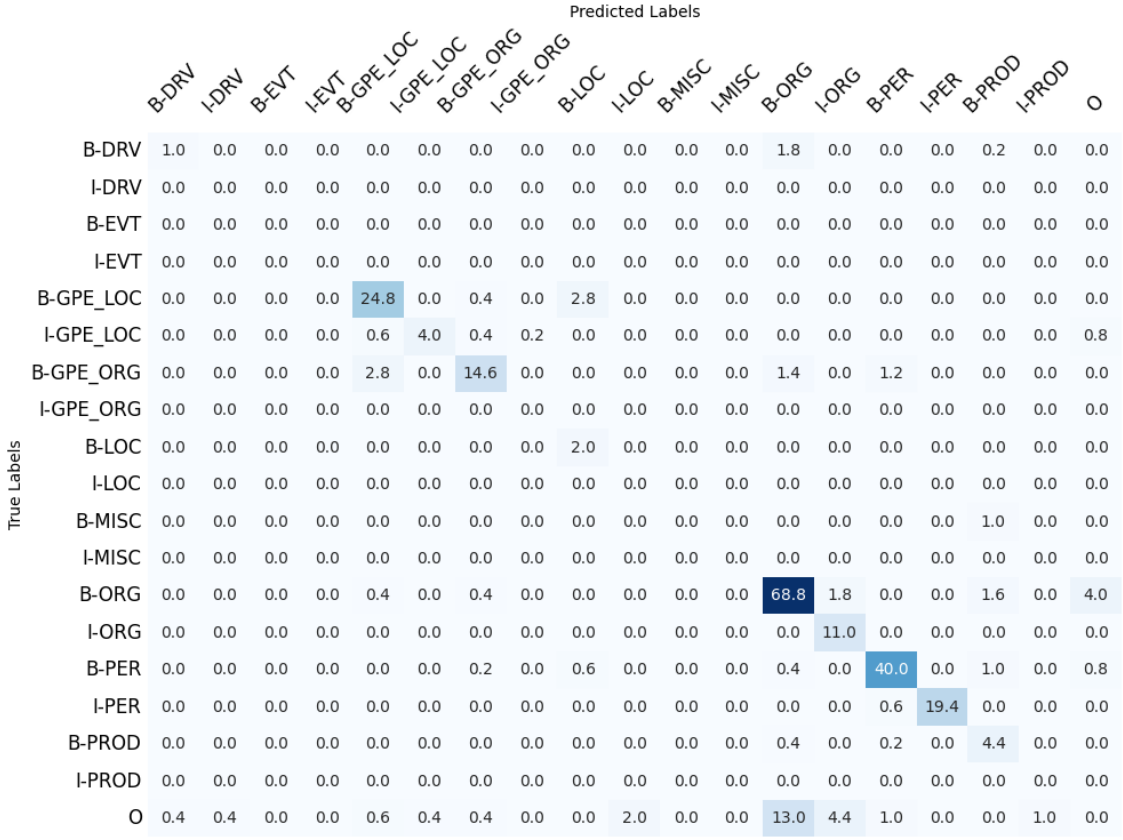


Figure B.2: NB-BERT confusion matrix token level classification NPSC sample

Bibliography

Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186. 2.7

Andrew L. Beam, Arjun K. Manrai, and Marzyeh Ghassemi. Challenges to the Reproducibility of Machine Learning Models in Health Care. *JAMA*, 323(4): 305–306, 01 2020. doi: 10.1001/jama.2019.20866. URL https://doi.org/10.1001/jama.2019.20866. 4.6

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922. 2.10.3

Sravan Bodapati, Hyokun Yun, and Yaser Al-Onaizan. Robustness to capitalization errors in named entity recognition. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 237–242, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5531. URL https://aclanthology.org/D19-5531. 2.7, 2.8.3

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 06 2017. doi: 10.1162/tacl_a_00051. URL https://doi.org/10.1162/tacl_a_00051. 2.9.1

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish,

Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165. 2.10, 2.10.3

Stefan Brüggemann. *Collaboration and the Semantic Web: Social Networks, Knowledge Networks, and Knowledge Resources: Social Networks, Knowledge Networks, and Knowledge Resources*. Advances in Human and Social Aspects of Technology (2328-1316). Information Science Reference, 2012. ISBN 9781466608955. URL https://books.google.no/books?id=yqOeBQAAQBAJ. 2.8.1

Jason P.C. Chiu and Eric Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, 07 2016. doi: 10.1162/tacl_a_00104. URL https://doi.org/10.1162/tacl_a_00104. 2.7

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(null):24932537, nov 2011. 2.7

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. 1, 2.7, 2.9.1, 2.10, 2.10.1

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005. doi: https://doi.org/10.1016/j.artint.2005.03.001. URL https://www.sciencedirect.com/science/article/pii/S0004370205000366. 2.6

Venkatesh Ganti, Arnd C. König, and Rares Vernica. Entity Categorization over Large Document Collections. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 274–282, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401927. URL https://doi.org/10.1145/1401890.1401927. event-place: Las Vegas, Nevada, USA. 2.8.1

Ian Goodfellow, Yoshua Bengio, and Aaaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN 9780262035613. URL https://books.google.no/books?id=Np9SDQAAQBAJ. 2.7

Archana Goyal, Vishal Gupta, and Manish Kumar. Recent named entity recognition and classification techniques: A systematic review. *Computer Science Review*, 29:21–43, 2018. doi: https://doi.org/10.1016/j.cosrev.2018.06.001. URL https://www.sciencedirect.com/science/article/pii/S1574013717302782. 2.6

Zellig S. Harris. Distributional structure. WORD, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL https://doi.org/10.1080/00437956.1954.11659520. 2.9.1

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv e-prints*, art. arXiv:1508.01991, August 2015. doi: 10.48550/arXiv.1508.01991. 2.7

Christoph Hube. Bias in wikipedia. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 717721, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349147. doi: 10.1145/3041021.3053375. URL https://doi.org/10.1145/3041021.3053375. 2.10.3

Rasmus Hvingelby, Amalie Brogaard Pauli, Maria Barrett, Christina Rosted, Lasse Malm Lidegaard, and Anders Søgaard. DaNE: A named entity resource for Danish. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4597–4604, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.565. 4.5.5

Daniel Jurafsky and James H. Martin. Speech and language processing (3rd ed. draft), 2022. URL https://web.stanford.edu/~jurafsky/slp3/. 2.1, 2.2, 2.2.2, 2.2.2, 2.3, 2.6, 2.7, 2.9, 2.9.1, 2.9.1, 2.9.1, 2.10

Fredrik Jørgensen, Tobias Aasmoe, Anne-Stine Ruud Husevåg, Lilja Øvrelid, and Erik Velldal. NorNE: Annotating Named Entities for Norwegian. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4547–4556, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.559. 2.3, 2.5, 2.7, 3.1, 4.2, 4.4, 4.5.5, 1

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. 4.6

Per E Kummervold, Javier De la Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. Operationalizing a national digital library: The case for a Norwegian transformer model. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 20–29, Reykjavik, Iceland

(Online), May 31–2 June 2021. Linköping University Electronic Press, Sweden. URL https://aclanthology.org/2021.nodalida-main.3. 2.10.2, 4.2, 4.5.3, 4.5.4

Andrey Kutuzov, Jeremy Barnes, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. Large-scale contextualised language modelling for Norwegian. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 30–40, Reykjavik, Iceland (Online), May 31–2 June 2021. Linköping University Electronic Press, Sweden. URL https://aclanthology.org/2021.nodalida-main.4. (document), 2.7, 2.2, 2.10.2, 4.2, 4.5.1

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition, June 2016. URL https://aclanthology.org/N16-1030. 2.7

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Trans. on Knowl. and Data Eng.*, 34(1):5070, jan 2022. doi: 10.1109/TKDE.2020.2981314. URL https://doi.org/10.1109/TKDE.2020.2981314. 2.6

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 367–377. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/li20c.html. 4.6

Alexandra Luccioni and Joseph Viviano. What's in the box? an analysis of undesirable content in the Common Crawl corpus. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 182–189, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.24. URL https://aclanthology.org/2021.acl-short.24. 2.10.3

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090. 2.9.1

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30, 08 2007. doi: 10.1075/li.30.1.03nad. 2.6

Dennis Pauly and Guido Nottbusch. The influence of the german capitalization rules on reading. 15:1–15, 03 2020. doi: 10.3389/fcomm.2020.00015. 2.8.3

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL http://arxiv.org/abs/1802.05365. 2.9.1

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL https://aclanthology.org/P19-1493. 4.2

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-3516. 2.4

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2.10.3

Kashif Riaz. Rule-based named entity recognition in urdu. In *Proceedings of the 2010 Named Entities Workshop*, NEWS '10, page 126135, USA, 2010. Association for Computational Linguistics. ISBN 9781932432787. 2.6, 2.8.3

Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. On challenges in machine learning model management. *IEEE Data Engineering Bulletin*, 2015. URL https://www.amazon.science/publications/on-challenges-in-machine-learning-model-management. 4.6

Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/S13-2056. 2.2.2

Satoshi Sekine and Chikashi Nobata. Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon,

Portugal, May 2004. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/65.pdf. 2.6

Larry Smith, Lorraine Tanabe, Rie Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig Struble, Richard Povinelli, Andreas Vlachos, William Baumgartner Jr, Lawrence Hunter, Bob Carpenter, and W. Wilbur. Overview of biocreative ii gene mention recognition. *Genome biology*, 9 Suppl 2:S2, 09 2008. doi: 10.1186/gb-2008-9-s2-s2. 2.1

Per Erik Solberg and Pablo Ortiz. The norwegian parliamentary speech corpus. *CoRR*, abs/2201.10881, 2022. URL https://arxiv.org/abs/2201.10881. 3.2

Per Erik Solberg, Arne Skjærholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen. The Norwegian dependency treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 789–795, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2014/pdf/303_Paper.pdf. 3.1

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html. 2.9

Simone Tedeschi, Valentino Maiorca, Niccolò Campolungo, Francesco Cecconi, and Roberto Navigli. WikiNEuRal: Combined Neural and Knowledge-based Silver Data Creation for Multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2521–2533, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.215. URL https://aclanthology.org/2021.findings-emnlp.215. 2.4

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003a. URL https://aclanthology.org/W03-0419. 2.4

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003b. URL https://aclanthology.org/W03-0419. 2.4

Samia Touileb, Lilja Øvrelid, and Erik Velldal. Occupational biases in norwegian and multilingual language models. pages 200–211, 01 2022. doi: 10.18653/v1/2022.gebnlp-1.21. 2.10.3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762. 1, 2.9.1, 2.10

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. CrossWeigh: Training Named Entity Tagger from Imperfect Annotations. Technical Report arXiv:1909.01441, arXiv, September 2019. URL http://arxiv.org/abs/1909.01441. arXiv:1909.01441 [cs] type: article. 2.4

Jie Yang and Yue Zhang. NCRF++: An Open-source Neural Sequence Labeling Toolkit. Technical Report arXiv:1806.05626, arXiv, June 2018. URL http://arxiv.org/abs/1806.05626. arXiv:1806.05626 [cs] type: article. 2.7, 4.2

Jie Yang, Shuailong Liang, and Yue Zhang. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3879–3889, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1327. 2.7

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2.10.1