# Multilingual Deep Learning models for Entity Extraction in NLP

Arthur Leloup

Academic Year 2019-2020

# Contents

# Abstract

In the past few years, the development of neural language models (LMs) have had a large impact on the performance of many Natural Language Processing (NLP) tasks like Named Entity Recognition (NER), a classification tasks that aims to identify named entities like persons, locations or organizations in text. The ability of pretrained neural LMs to provide vector representations for the context-specific meaning of words has resulted in human-level performance on some NLP tasks. Only very recently, the training efficiency of neural LMs like Flair and BERT have allowed to even feed large corpora of multilingual text as training data, thus obtaining LMs that are hypothesized to provide contextualized representations for words in different languages in a single, high-dimensional, semantic vector space.

In this thesis, we aimed to compare how state-of-the-art contextualized monolingual and multilingual embeddings affect the performance of NER tasks on monolingual and multilingual data. We trained a Bidirectional Long Short-Term Memory NER tagger with a Conditional Random Field decoding layer (BiLSTM-CRF) on well-validated NER benchmark datasets in English (CoNLL2003), Dutch (CoNLL2002) and French (WikiNER), using monolingual as well as multilingual representations obtained by concatenating different contextualized, static as well as task-specific embeddings. These experiments were also performed using smaller "real-world" French, Dutch and multilingual datasets that were obtained from a document annotation application developed by the company Faktion.

We confirmed earlier observations that monolingual embeddings from pretrained LMs outperform multilingual embeddings on high-quality benchmark datasets like CoNLL. The difference was, however, surprisingly small and multilingual BERT embeddings clearly outperformed all monolingual embeddings when the input data was multilingual. For the Faktion datasets, the transfer learning approach did not seem to improve performance as compared to static or task-specific embeddings, suggesting that the "general" linguistic knowledge acquired by these LMs during pretraining was not very relevant in this context. However, concatenated vectors of different multilingual representations performed similarly or better on the monolingual and multilingual datasets, respectively, as compared to monolingual embeddings. These results indicate that for this specific application, multilingual embeddings omit the need to maintain language-specific pipelines, while maintaining or even improving the performance on the NER task.

# Chapter 1

# Introduction

Natural Language Processing (NLP) is a subfield of linguistics, computer science and statistics that aims to design algorithms that allow interaction and understanding between computers and human languages. Human languages are highly complex and the understanding of language typically requires a huge amount of prior knowledge that we gradually acquire during our lives. We are able to infer a massive amount of information from language, often only from subtle pieces of non-linguistic information, such as the context it is used in.

In this thesis, we focus on Named Entity Recognition (NER), a NLP-task that consists of identifying names of persons, locations, organizations or other so-called "named entities" in text. Since text is a form of unstructured data, a key challenge in NER (and NLP in general) is to find strategies that effectively impose structure and provide numeric representations of the semantic and syntactic properties of each word in the input text in such a way that it renders a statistical NER model able to predict the entity label associated with each word in the input text.

Around 2013, static word embeddings like word2vec (Mikolov et al., 2013) became a standard component of most NER systems due to their ability to capture semantic information in a dense, numeric vector. They are, however, limited in the fact that they aggregate all different meanings of a word into a single vector. This issue was addressed around 5 years later, when context-dependent word representations obtained from large, pretrained (deep) neural language models (LMs) began to outperform static word embeddings. Neural LMs are models that are trained on the LM objective, i.e. predicting the conditional distribution of a word in context, in a completely unsupervised way. This implicitely forces the model to internalize semantic and syntactic concepts. As such, the activations learned by these models provide very meaningful and context-dependent word representations that have shown to be very useful to represent words in typically smaller, supervised NLP tasks like NER. Since the release of ImageNet (Deng et al., 2009) in 2009, this concept of transfering knowledge from large, pretrained models to improve the performance of smaller - typically supervised - tasks has been applied succesfully in the field of computer vision. With the rise of the aforementioned pretrained (deep) neural LMs that provide excellent contextualized word embeddings, this concept of "transfer learning" has also been driving most of the progress in the NLP field during the past 3 years.

Recently, the improved training efficiency of character-based LMs like Flair (Akbik et al.,

2018) or Transformer-based BERT (Devlin et al., 2019) have even allowed to feed large corpora of text in many different languages to train multilingual LMs. The internal representations of these multilingual LMs are hypothesized to provide contextualized, language-independent semantic word representations (Pires et al., 2019). There are many potential applications where these multilingual embeddings could address many of the limitations of monolingual embeddings, including in NER. However, the difference of multilingual embeddings and monolingual embeddings in terms of downstream NER performance are not well-described.

The main goal of this thesis is to compare how the use of monolingual or multilingual contextualized word embeddings affects performance of a specific Named Entity Recognition (NER) task when applied on monolingual (English, Dutch and French) and multilingual data. We used a Bidirectional Long Short-Term Memory NER tagger with a Conditional Random Field decoding layer (BiLSTM-CRF), a neural network architecture that has shown to provide state-of-the-art performance on many NER tasks[1]. Monolingual and multilingual pretrained LMs were used to obtain contextualized monolingual and multilingual word representations. Since it has been shown that NER performance might benefit from concatenating these contextualized representations with task-specific (Lample et al., 2016) and/or static word vectors like word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), different so-called stacked embeddings will be evaluated as well.

We first reproduced benchmarks for English, Dutch and French NER using the widely-used human-annotated datasets CoNLL (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) (for Dutch and English) and WikiNER for French (Nothman et al., 2013). To evaluate how NER performance is affected by multilingual and monolingual embeddings when the input data itself is multilingual, we constructed multilingual datasets by merging the aforementioned monolingual datasets. Next, the pipelines were trained and evaluated on “real-world” Dutch and French monolingual datasets as well as a multilingual dataset, provided by the company Faktion.

In chapter 2, we provide a detailed overview of the research questions and introduce some important theoretical concepts related to monolingual and multilingual word embeddings in the context of NER. In chapter 3 we introduce the datasets used to evaluate our NER systems, provide a brief overview of some common neural LM architectures and how they have contributed to the state-of-the-art performance of many NER systems that are used today. We also introduce how the BiLSTM-CRF architecture was trained to perform NER and how we evaluated the performance of the different NER systems. In chapter 4, we report the main results. Lastly, in chapter 5, we summarize the main conclusions, discuss some of the limitations of this study and provide some future perspectives.

[1] http://nlpprogress.com/english/named_entity_recognition.html

# Chapter 2

# Research problem

## 2.1 Research hypotheses

The Antwerp-based company Faktion develops machine and deep learning algorithms in the field of computer vision, natural language and sensor data for a wide range of industries. In 2019, they developed Metamaze, a platform to automate complex manual workflows of both structured and unstructured documents. An important part of this platform consists of the automatic annotion of named entities in documents. The NER system consists of 2 consecutive steps: first, the (dominant) language of the document is predicted. Next, a monolingual NER system specific for the predicted language is used to predict entities in the document. This approach has some limitations. The 2-step procedure requires training and maintaining pipelines for each individual language. Also, the monolingual NER system might perform poorly when the system fails to predict the correct language in the first step or when the input document is multilingual.

Using multilingual embeddings might provide a solution to the aforementioned limitations. Even though this would reduce the complexity of the pipeline, the size of the training corpus of a (multilingual) LM is still limited, suggesting that monolingual LMs might internalize more language-specific linguistic knowledge and, hence, provide representations that results in better performance on monolingual NER tasks. In this thesis, we aim to investigate this claim.

We hypothesize that:

- Monolingual embeddings outperform multilingual embeddings when the input data is monolingual
- Multilingual embeddings outperform monolingual embeddings when the input data is multilingual

To investigate the first hypothesis, we will use high-quality, well-validated benchmark datasets for NER in English, Dutch and French. In addition, the different NER systems will be evaluated on smaller datasets provided by Faktion that were user-annotated in the Metamaze platform itself. For both the benchmark and Faktion datasets, multilingual dataset will be generated to evaluate the second hypothesis that for the specific (and not very uncommon) case of multilingual input data, multilingual representations outperform monolingual ones in terms of NER performance.

Below, we provide a brief overview of some important concepts related to the aforementioned problem statements. We will briefly introduce the concept of NER and discuss the basic idea of word embeddings, followed by a discussion on language models - probabilistic models that play a pivotal role in enabling computers to acquire linguistic knowledge. Lastly, we discuss how these language models can be used to obtain both monolingual as well as multilingual word embeddings that can be used as input for NER classifiers.

## 2.2 Named Entity Recognition

Named Entity Recognition (also known as Entity Extraction) is an NLP-task that aims to automatically locate and annotate pieces of text as *entities* such as persons (PER), organizations (ORG), geo-political entities (GPE) or locations (LOC), dates etc... (Figure 2.1).

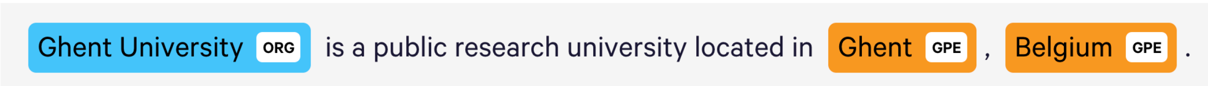


Figure 2.1: An example of a simple NER task with the displaCy Named Entity Visualizer.

This might appear somewhat trivial at first sight with regular string or pattern matching and a list of known entities, but there are quite some challenges involved. These are for example related to the fact that many words have different meanings (Washington, Apple, Jobs) and that many entities span several words which makes defining their boundaries difficult - even for humans (the New England Journal of Medicine, Swiss Federal Polytechnic School, etc...). NER is essentially a classification problem. The sequential nature of languange renders some algorithms more suited for the task than others. Today, most NER systems are based on neurall networks, as discussed in chapter 3. However, no matter which architecture is used, in order to render a NER classifier able to learn how to recognize named entities in text, the input representations must encode the semantic and syntactic properties of words in a meaningful way. Not surprisingly, a lot of the progress in the NER (or any other NLP task) during the past few decades can be attributed to methods that provide these meaningful word representations.

## 2.3 Static word representations

Possibly the simplest solution to impose structure and represent words in a quantitative way would be to use a single one-hot vector $\mathbf{w} \in \mathbb{R}^{|V|}$ for every word in a given vocabulary of size $|V|$, for example:

$$\begin{aligned} \mathbf{w}_{aardvark} &= [\ 1\ 0\ 0\ \dots\ 0\ 0\ \dots\ 0\ 0\ ]^T \\ \mathbf{w}_{cat} &= [\ 0\ 0\ 0\ \dots\ 1\ 0\ \dots\ 0\ 0\ ]^T \\ \mathbf{w}_{zoo} &= [\ 0\ 0\ 0\ \dots\ 0\ 0\ \dots\ 0\ 1\ ]^T \end{aligned}$$

This obviously has some limitations: with arguably millions of words in the English language, these vectors are extremely sparse. Secondly, any 2 one-hot vectors are orthogonal and do not encode concepts like semantic similarity:

$$\mathbf{w}_{cat}^T \mathbf{w}_{dog} = \mathbf{w}_{cat}^T \mathbf{w}_{airplane} = 0$$

It is not unreasonable to assume that there must exist some $n$-dimensional hyperspace (with $n$ much lower than the size of the entire English vocabulary) that can encode all semantics of a language. Many techniques have been proposed to encode words in a real-numbered space, with the two primary families being matrix factorization methods (such as latent semantic analysis (LSA) (Deerwester et al., 1990)) and iteration based methods based on local context (such as word2vec (Mikolov et al., 2013)). A key concept that many of these different methods have in common is the idea of *distributional similarity*, an important hypothesis in linguistics (popularized by the British linguist J.R. Firth (Firth, 1957)) that words with a similar meaning occur in a similar context and *vice versa*.

Matrix factorisation methods typically involve constructing a large matrix - often based on co-occurance counts of words in documents or smaller context windows. Then, dimension-reduction techniques such as truncated singular value decomposition are used to compress the information and obtain low-dimensional vector representations for all words in the vocabulary. Alternatively, instead of performing dimension-reduction on this large matrix, iteration-based methods rely on neural networks that learn iteratively how to represent words in a meaningful way. A major breakthrough - in particular with respect to computational efficiency - was the development of the word2vec framework in 2013 by a team at Google (Mikolov et al., 2013). The word2vec model is essentially a simplified language model (cf infra) trained to predict whether words are likely to occur together. It appeared that optimizing the model for this task yielded excellent word embeddings that captured a lot of semantic information, with the vector representations of semantically similar words being oriented closely together in typically 300-dimensional embedding space. In addition, the dimensions learned by the model (hence, the relative orientation of the word vectors in the embedding space) even showed to encode meaningful semantic or syntactic concepts, illustrated by low-dimensional linear mappings of some word embeddings as shown in Figure 2.2. Other widely-used word vectors include Stanford's typically low-dimensional (50-100) GloVe vectors (Pennington et al., 2014) that are partly based on word co-occurence count matrices, or Facebook's 100-300-dimensional fastText vectors (Bojanowski et al., 2017; Joulin et al., 2017).

A problem with these static word embeddings is the way out-of-vocabulary (OOV) words are handled: they don't provide embeddings for words that were not encountered during training. A possible solution is to learn subword embeddings. The intuition is straightforward: the suffix *-shire* lets you guess that Melfordshire is probably a location, or the suffix *-osis* that Myxomatosis might be a sickness[1]. Subword-based embeddings have been developed by e.g. fastText through representing words as bags of n-grams (Bojanowski et al., 2017). This largely solves the problem of OOV words but results in huge models. An arguably more elegant solution for this problem is provided by BytePair encoding, an unsupervised tokenization algorithm that iteratively merges frequent pairs of characters for a fixed number of iterations (Sennrich et al., 2016). Applying this algorithm on a large

[1]Example taken from: https://nlp.h-its.org/bpemb/

corpus of text for a well-chosen number of iterations typically yields reasonable subword segementations, where frequent words are represented as such, and rare words as subword tokens. When these tokens are subsequently used to learn embeddings, the resulting BytePair embeddings appear to perform very well when used as input for different NLP tasks.

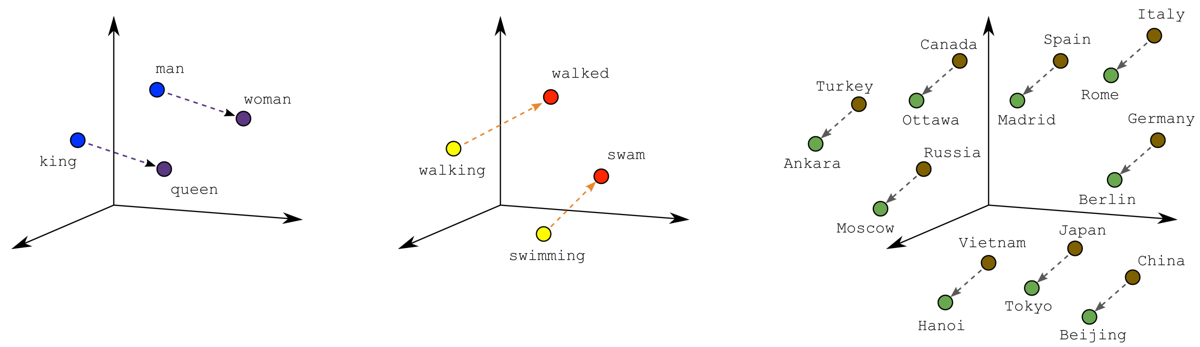


Figure 2.2: Low-dimensional linear mappings of word vectors elegantly demonstrate how the learned dimensions effectively encode meaningful semantic or syntactic concepts like "gender" (left), verb tense (middle) or country-capital relationships (right). Taken from shorturl.at/ivz38.

## 2.4 Contextualized word representations and language models

Even though word embeddings like word2vec and fastText led to significant improvements in the NLP field, they suffer from a major limitation: many words have different meanings and these static word embeddings aggregate all these different meanings into a single vector (Arora et al., 2018). This is especially problematic for polysemous words like "play", "Washington" or "mouse". These issues have been addressed in the recent years with the development of so-called contextualized word embeddings - i.e. embeddings that represent words in their context, meaning that a single polysemous word like "Washington" has a different word embedding, depending on the context it appears in. An alternative approach would be to obtain representations by including randomly-initialized vectors as parameters of a NER classifier or to train task-specific character-feature embeddings on the NER training data directly, as described in (Lample et al., 2016). We will refer to these task-specific representations as one hot word type embeddings (OHEs) and character embeddings, respectively, and discuss them further in chapter 3. However, learning these task-specific representations from scratch for every NER task is not very efficient and a much more successful approach is to use contextualized word representations obtained from pretrained neural language models (LMs), as discussed below.

Neural LMs are probabilistic classifiers that are trained to predict the probability of a word given the previous words. The concept of LMs has played a crucial role in NLP for many decades (long before the rise of deep learning in NLP) as they allow to predict words given some context as input or provide joint probabilities of word sequences (Equation (2.1)). A good LM will thus assign a higher joint probability to the sequence "I like bikes" as compared to “Airplane cat want is” because the former sequence is semantically and syntactically valid, while the latter sequence makes no sense. This renders them very useful for many NLP tasks. Yet, the underlying idea of the LM objective is surprisingly simple. The joint probability of a sequence of $m$ words $(w_1, w_2, ..., w_m)$ can be decomposed into a product of conditional probabilities (Jurafsky and Martin, 2019):

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, ..., w_{i-1}) \tag{2.1}$$

A so-called forward LM is trained to estimate the conditional probability of a word $w_i$ given the previous words $w_1, ..., w_{i-1}$. Since the number of all possible word sequences for a vocabulary of size $|V|$ is $|V|^m$, estimating these conditional probabilities by simply counting how often every sequence $(w_1, ..., w_{i-1})$ is followed by $w_i$ in a corpus is practically impossible (i.e. many valid sentences will never occur in a reasonably-sized training corpus). Traditionally, this was often solved by approximating the entire history of a word by a fixed window of size $n - 1$ (with n typically not larger than 5):

$$P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-n+1}, ..., w_{i-1}) \tag{2.2}$$

This so-called Markov assumption (Manning, 2019b) reduces the number of all possible sequences from $|V|^m$ to $|V|^n$, making it practically feasible to estimate these conditional probabilities by obtaining relative counts of all $n$-grams in a corpus (a simple application of Bayes’ rule). Even though it is clear that the Markov assumption is not strictly valid for language, these so-called $n$-gram models work surprisingly well, are very efficient to train and, hence, still used for some applications today.

However, for many NLP tasks, having access to information that might be arbitrarily far away in the sequence is required to perform well, and $n$-gram models are intrinsically limited in terms of the context they can consider. Luckily, more complex model architectures like Recurrent Neural Networks (RNNs), Long Short-term Memory (LSTM) or Transformers are able to handle much longer histories, rendering them much better at the LM task. These architectures will be discussed in chapter 3. There has been a steady rise in the quality and applications of (deep) neural LMs over the past 1-2 decades, with modern neural LMs being able to internalize an incredible amount of linguistic knowledge from huge corpora of text. While there are major differences in terms of the neural architectures and training algorithms, a key concept that drives a lot of these models’ ability to internalize linguistic knowledge is the LM task: predicting a word given some context as input. When trained well, these models are able to provide probability distributions over word sequences or predict words when given some context, rendering them extremely useful in applications such as auto-completion tasks, text generation, machine-translation or speech recognition. In addition, the contextual information (that may extend back to the very beginning of the sequence) that is considered by the model to predict the next word in a sequence is essentially encoded in the hidden states of these neural LMs. Recent attempts to use these representations as contextualized word embeddings for different NLP tasks - including NER - have been very successful, as discussed in chapter 3.

## 2.5 Monolingual versus multilingual embeddings

Today, pretrained static word embeddings like word2vec (Mikolov et al., 2013), fastText (Bojanowski et al., 2017) and GloVe (Pennington et al., 2014) are available for a wide range of languages. Similarly, many of the previously discussed neural LMs provide contextualized word representations. The recently developed character-based Flair LM (Akbik et al., 2018) or Transformer-based BERT model (Devlin et al., 2019) (both models are discussed in chapter 3) have been pre-trained in several languages and are often made freely available. Examples include CamemBERT (Martin et al., 2019) and BERTje (de Vries et al., 2019), a French and Dutch version of BERT, respectively, or the Dutch and French versions of Flair made available in the Flair's NLP library.

These previously-discussed frameworks provide word representations in a monolingual semantic space. As mentioned previously, this has some limitations that are directly related to the research problems formulated earlier. For example, implementing an existing NLP task for a similar problem in a different language involves starting from scratch. Training and maintaining multiple pipelines for different languages has to be done independently. Many words - especially named entities - are similar across languages, suggesting that having different independent monolingual embeddings to represent these entities in different languages might not be very efficient. Additional problems with monolingual embeddings arise when the input data is multilingual.

These problems have been partly addressed through unsupervised or semi-supervised algorithms that jointly map monolingual word embeddings into a common multilingual embedding space such that words with a similar meaning are aligned (Artetxe et al., 2018; Lample et al., 2018). In addition, the aforementioned BytePair encoding algorithm has been used to create (static) multilingual BytePair embeddings based on a single corpus of Wikipedia articles in 275 languages (Heinzerling and Strube, 2017). With the efficiency of BERT's transformer architecture and Flair's character-based LM objective, the question then arises whether these models can be jointly trained over a large corpus of text in many different languages to learn contextualized encodings for a multilingual vocabulary in a single embedding space without having to explicitely align different monolingual embeddings. This has been attempted for both the BERT and Flair models: multilingual BERT (mBERT)[2] was trained on a corpora consisting of the 100 most common languages on Wikipedia and the multilingual version of Flair (mFlair) was trained on over 300 languages[3]. Since a single multilingual vocabulary was provided as training data, these models are hypothesized to learn word meaning in a language-independent way such that the vector representations for e.g. *vélo*, *bike* and *fiets* are located close to each other in the embedding space. Potential applications for such language-independent semantic representations are for example so-called zero-shot cross-lingual model transfer. This involves training a downstream task like NER using data from a high-resource language like English and effectively applying the learned classifier in another language like Dutch. Experiments with mBERT have shown that this approach works surprisingly well, indicating that mBERT embeddings for words in different languages share a significant subspace that encodes semantic information in a language-independent way (Pires et al., 2019).

[2]GitHub: google-research/bert/multilingual
[3]Github: stefan-it/flair-pos-tagging

## 2.6 Summary

To summarize, in this thesis, we aim to evaluate how state-of-the-art monolingual and monolingual embeddings affect the performance of different NER tasks. To challenge the research hypotheses mentioned at the beginning of this chapter, we will use both monolingual and multilingual datasets to train and evaluate different NER classifiers using different embedding types. In order to acquire a better understanding on how different embedding types affect NER performance, we will not only evaluate single monolingual and multilingual embeddings, but also create so-called stacked embeddings by concatenating different word representations into a single, high-dimensional vector. In the next chapter, we introduce the different embedding types that were used for the experiments presented in this thesis.

# Chapter 3

# Methods

## 3.1 Data

### 3.1.1 Benchmark datasets

To evaluate how different monolingual and multilingual embeddings affect NER performance, we made use of large, well-annotated datasets for NER tasks. For Dutch and English, we used the data from the shared task of respectively the 2002 and 2003 Conference on Computational Natural Language Learning (CoNLL) (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). These are widely used benchmarks datasets for Dutch and English NER. They consist of separate train, development and test sets of human-annotated, tokenized sentences from news articles from Reuters Corpus (English CoNLL 2003) and four editions of the Belgian newspaper "De Morgen" of the year 2000 (Dutch CoNLL2002). For French, the WikiNER dataset (wiki-3 version) was used (Nothman et al., 2013). This dataset relied on automatic NER annotation algorithms based on Wikipedia's article metadata as well as some additional heuristics, which has been shown to yield excellent-quality NER annotations. Because this dataset is much larger as compared to the CoNLL datasets, we used a random sample of 10 % of the entire dataset to approximately match the size of the CoNLL datasets (Table 3.1).

We maintained the train, development and test split of the original CoNLL datasets. However, the CoNLL2002 dataset contained a low number of relatively long sentences. Since some of the Transformer architectures used to obtain contextualized token representations (cf infra) have a limited input sequence length, the sentences exceeding 250 tokens were removed from the CoNLL2002 dataset for all experiments. This resulted in the removal of 4/15806 sentences from the training set and 1/5195 sentences in the test set.

All three datasets were pre-labelled with 4 different NER tags: persons (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC), they were all converted to the same BIO2 format according to the original CoNLL2002 dataset (Tjong Kim Sang, 2002). This means that every line represented a word token and the NER tag, prefixed with either "B-" ("beginning") or "I-" ("inside") for respectively the first and subsequent tokens of entities spanning multiple tokens. Single-token entities were prefixed with "B-", all other tokens were tagged with the label "O" ("other"). Different sentences were separated by a blank line. Indicators of document boundaries, such as the DOCSTART token

Table 3.1: Overview of the characteristics of the train, development (dev) and test benchmark datasets used to evaluate different monolingual and multilingual embeddings on monolingual and multilingual NER tasks. The WikiNER dataset was downsampled to roughly match the size of the Dutch and English datasets, the multilingual datasets was created by randomly sampling 1/3th of every monolingual dataset (maintaining the same train, dev and test splits).

| | | # sentences | | | # tokens | | |
|---|---|---|---|---|---|---|---|
| | | Train | Dev | Test | Train | Dev | Test |
| En | CoNLL03 | 14041 | 3250 | 3453 | 203621 | 51362 | 46435 |
| Nl | CoNLL02 | 15802 | 2895 | 5194 | 199969 | 37687 | 68466 |
| Fr | WikiNER | 10713 | 1190 | 1323 | 279729 | 34824 | 30991 |
| Multi | Multilingual | 13518 | 2444 | 3323 | 232173 | 40141 | 49444 |

in the CoNLL datasets, were removed. An example sentence from the Dutch CoNLL2002 dataset is given below.

```
Onder O
de O
bruuske O
tempoversnellingen O
van O
Gilberto B-PER
Simoni I-PER
op O
de O
steilste O
gedeelten O
van O
de O
Pratonevoso B-LOC
moest O
Casagrande B-PER
passen O
. O
```

In addition to these three monolingual dataset, we combined them into a single trilingual dataset to perform multilingual NER. For efficiency reasons, the combined dataset was obtained by randomly sampling one third of the sentences from each monolingual dataset (maintaining the same train, development and test splits). The specific datasets used here - together with all the code to reproduce the experiments - are made available on the GitHub repository of this thesis (arthur-arthur/NER). The English CoNLL2003 dataset requires a licence from Reuters Corpus (free for research purposes) and was therefore not included in the repository.

### 3.1.2 Faktion datasets

In addition to these well-validated benchmark datasets, similar experiments were performed on smaller "real-word" datasets that were obtained from Metamaze, a document annotation application developed by Faktion. The data consisted of text obtained with an in-house developed optical character recognition (OCR) pipeline on relatively poor quality scans of building plans. Hence, many of the tokens consisted of single characters or only punctuation, resulting in long sequences and many tokens with very little semantic information. The data were user-annotated within the Metamaze application, i.e. based on the location of the tokens in the document scans, not on the text output from the OCR pipeline itself. Given the limited input sequence length of some of the Transformer models, we reduced the average sequence length (i.e. tokens per document) by removing all tokens that consisted of punctuation only. Another difference with the benchmark datasets described earlier was the number of NER labels to predict: while there were 4 different entity labels in the CoNLL and WikiNER datasets (PER, LOC, ORG and MISC), there was only a single category ("naam_bouwplan") in the Dutch Faktion dataset, and only one additional category ("naam_bouwplan_2") in the French dataset, with approximately 95% of all entities belonging to a single category. A representative fragment of the OCR output for one of the documents (after preprocessing) is given below:

```
310 B-naam_bouwplan
4 I-naam_bouwplan
Lewsrass O
wes O
BATIMENT B-naam_bouwplan
A I-5e57
aren O
seres O
Meg O
names O
808 O
0e O
side O
panama O
VERDIERING B-naam_bouwplan
3 I-naam_bouwplan
- I-naam_bouwplan
3 I-naam_bouwplan
1ewe I-naam_bouwplan
E74 I-naam_bouwplan
A O
0e O
58 O
A0 O
03 O
```

Another important characteristic of these datasets was that none of them were strictly monolingual. The language was decided on the dominant language (Table 3.2). A third

Table 3.2: Number of sequences (documents) and tokens per train, development (dev) and test splits for the Dutch (Nl), French (Fr) and multilingual Faktion datasets. The reported values are the averages of the 5 independent random splits. Note that even the monolingual datasets contained bilingual documents, the language was chosen based on the dominant language in each dataset.

| | | # documents | | | # tokens | | |
|---|---|---|---|---|---|---|---|
| | % bilingual | Train | Dev | Test | Train | Dev | Test |
| Nl | 58 | 21 | 9 | 6 | 2619 | 1062 | 789 |
| Fr | 9 | 91 | 32 | 29 | 9706 | 3273 | 3131 |
| Multi | | 112 | 40 | 36 | 11976 | 4435 | 4171 |

- bilingual - dataset was created by merging the French and Dutch datasets. Given the characteristics of the datasets and, hence, the expected imprecision of the extra-sample error estimates, all experiments were repeated 5 times on independent random (80/20) splits of the data.

## 3.2 Word embeddings

### 3.2.1 Static and task-specific embeddings

As introduced in the previous chapter, there are a wide range of possible methods to obtain word representations that allow to perform NER. Static word embeddings like word2vec or fastText have had a significant impact on the performance of NER systems and have long been the default choice for many NLP tasks, including NER. They are still used today in combination with contextualized representations to provide state-of-the-art performance (Akbik et al., 2018). We therefore included two types of static word embeddings, i.e. monolingual fastText word embeddings (English, Dutch and French) as well as monolingual BytePair and multilingual BytePair (mBytePair) embeddings (Table 3.4), as discussed previously.

In addition, two types of task-specific representations were obtained. This included 300-dimensional one hot word type embeddings (OHEs) that were obtained by encoding all words in the vocabulary as one hot vectors and introducing an embedding layer into the network to obtain 300-dimensional word type representations[1]. In addition, we obtained 50-dimensional task-specific representations based on character-features[2], as described in (Lample et al., 2016). The OHEs were included to confirm the well-established idea on the differences between learning word embeddings from scratch on the NER objective and using representations from pretrained LMs. The character-feature embeddings were included because they have been shown to improve NER performance when concatenated to other word representations (Akbik et al., 2018; Lample et al., 2016). Both

[1]https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/ONE_HOT_EMBEDDINGS.md

[2]https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/CHARACTER_EMBEDDINGS.md

these task-specific representations were evaluated in the context of both monolingual and multilingual embeddings (Table 3.4).

### 3.2.2 Contextualized word embeddings

Even though static and/or task-specific representations have long been the default, today, all state-of-the-art NER systems rely on some form of contextualized word representation obtained from a pretrained neural LM. Below, we briefly discuss some neural network architectures and LMs that played an important role in the progress of the field during the recent years, how these models can be used to obtain multilingual word embeddings and which embedding types were used for the experiments presented in this thesis.

#### 3.2.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a class of neural network architectures that deal especially well with the sequential nature of natural language. In contrast to traditional neural networks, the hidden layer units do not simply pass their states along to input-output axis (i.e. from the input layer to the output layer), but pass the output of every unit to the next unit within that same layer. An example for a simple RNN LM is given in Figure 3.1 (example taken from (Manning, 2019b)). For this example, the RNN has to predict the next word in the sentence (“The students opened their”), i.e.:

$$P\left(w_5 \mid the,\ students,\ opened,\ their\right)$$

First, words are represented as one-hot vectors $\mathbf{w}_i \in \mathbb{R}^{|V|}$ with $i$ indicating the position of the word in the sequence. Then, an embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times |V|}$ is used to obtain static $d$-dimensional word embeddings like word2vec (Mikolov et al., 2013).

$$\mathbf{e}_i = \mathbf{E}\mathbf{w}_i$$

The hidden state for every unit $\mathbf{h}_i$ is computed as a linear combination of the word vector for the $i$-th word in the sequence, $\mathbf{e}_i$, as well as the hidden state from the previous unit $\mathbf{h}_{i-1}$. This is passed through a nonlinear activation function $\sigma$, just like a regular feed-forward neural network:

$$\mathbf{h}_i = \sigma(\mathbf{W}_h\mathbf{h}_{i-1} + \mathbf{W}_e\mathbf{e}_i)$$

Lastly, a linear combination of the last hidden state $\mathbf{h}_4$ (that encodes information on the last input word, as well as the entire past context) is passed through a softmax activation function (Goodfellow et al., 2016) (i.e. essentially performing multinomial logistic regression) to predict a probability distribution over the vocabulary ($\hat{\mathbf{y}}_4 \in \mathbb{R}^{|V|}$) for the last word $\mathbf{w}_5$, i.e. more generally:

$$\hat{\mathbf{y}}_i = softmax(\mathbf{W}_o\mathbf{h}_i)$$

with $\mathbf{W}_e$, $\mathbf{W}_h$, $\mathbf{W}_o$ being the weight matrices to be learned by the network (biases are omitted). Introducing the matrix $\mathbf{W}_h$ allows the network to learn how to make use of this past context (encoded in $\mathbf{h}_{i-1}$) while computing the output of its current state. Since the weight matrix $\mathbf{W}_h$ is shared across all timesteps, the length of the input sequence does not increases the model size in terms of parameters. While the aforementioned $n$-gram language models are constrained by the assumption that a given word only depends on a limited number of previous words, RNNs can effectively consider *all* preceding words, making them well-suited for the LM objective.

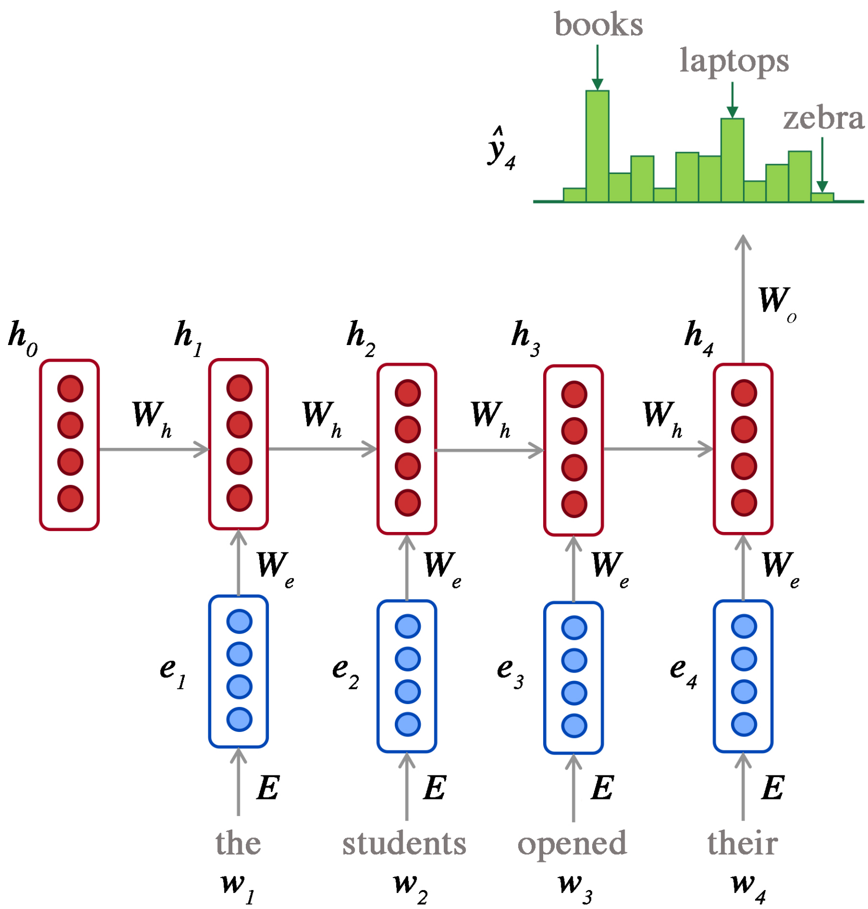


Figure 3.1: A schematic representation of a simple RNN LM with a single hidden layer. The output is passed through a softmax layer to generate a probability distribution over the entire vocabulary. The model is trained in a completely unsupervised way through minimizing the cross-entropy loss, i.e. the negative log probability that the model assigned to the correct word. This conceptually very simple idea forms the foundation of many state-of-the-art neural LMs that are used today. Adapted from (Manning, 2019a).

Unfortunately, in practice, RNN tend to encode relatively local information in their hidden states. The backpropagation step during training involves computing gradients and multiplying these (since the loss at a given time step is a function of the previous hidden state in the sequence). When the number of units in the sequence increases, these sequential multiplications can easily result in very large or very small numbers, a phenomenon commonly referred to as exploding or vanishing gradients, respectively (Bengio et al., 1993). Since the model parameters are iteratively updated according to this gradient, this can cause the algorithm to diverge or dramatically slow down the learning process.

#### 3.2.2.2 Long Short Term Memory models

Long Short Term Memory (LSTM) networks involve an additional level of architectural complexity and expand the parameter space, but provide an elegant solution to the aforementioned vanishing gradient issue. They learn how to actively manage contextual information in a so-called "memory-cell" - an additional vector that encodes historical information. The model is allowed to learn how to efficiently manage this historical information by parametrizing it with its own set of weights that control the flow of information in and out of the units of the hidden layers. This renders LSTMs better at handling contextual information and encoding long-range dependencies in its hidden states, as compared to simple RNNs (Goodfellow et al., 2016). LSTMs are widely used for many applications, including classification tasks such as NER. In fact, one of the best-performing NER classifier models today is based on a LSTM and this specific architecture was used for the experiments presented in this thesis. A schematic overview of this architecture is given in Figure 3.4. In addition, the ability of LSTMs to effectively model very long-range dependencies generally translates into excellent performance on the LM task. Today, many deep neural LMs consist of stacked layers of sequential LSTM units, and are often optimized for two objectives: to predict a word given the previous words, and to predict a word given the next words. Combining these forward and backward LMs yield so-called BiLSTM LMs that are used in many NLP applications, such as text autocompletion tasks, machine translation or speech recognition. In addition, their hidden states provide excellent contextualized word embeddings, as dicussed below.

#### 3.2.2.3 TagLM

One of the first architectures to effectively exploit the hidden states of a BiLSTM LM as contextual word vectors to train a NER classifier was the TagLM model (Peters et al., 2017). In short, the TagLM model made use of a large BiLSTM that was pre-trained using a large corpus of text on a LM objective. A second neural classifier was trained to perform NER. Instead of just feeding static word embeddings (e.g. word2vec or fastText) into this classifier, or training task-specific representations from scratch on the NER objective, they fed their input sequences into the pre-trained LM and extracted the hidden states. By feeding both static word embeddings as well as the concatenated forward and backward hidden states of the LM into the NER classifier, they achieved state-of-the-art performance on the NER task (Figure 3.2).

#### 3.2.2.4 ELMo

The ELMo (Embeddings from Language Models) architecture was an improved version of the TagLM model that achieved incredible performance on many NLP-tasks, including NER. ELMo is based on a deep, multi-layer BiLSTM LM to obtain context- and task-specific word representations from the hidden states of the LM layers (Peters et al., 2018). The excellent performance was partly achieved through representing words as a weighted average of the internal representations of the LM. It turned out that by introducing these weights as trainable parameters of the downstream task, the downstream model can make optimal use of the semantic and syntactic information that are encoded in the different layers of the LM. ELMo embeddings were able to improve the state-of-the-art of virtually all possible NLP tasks with a very significant margin.

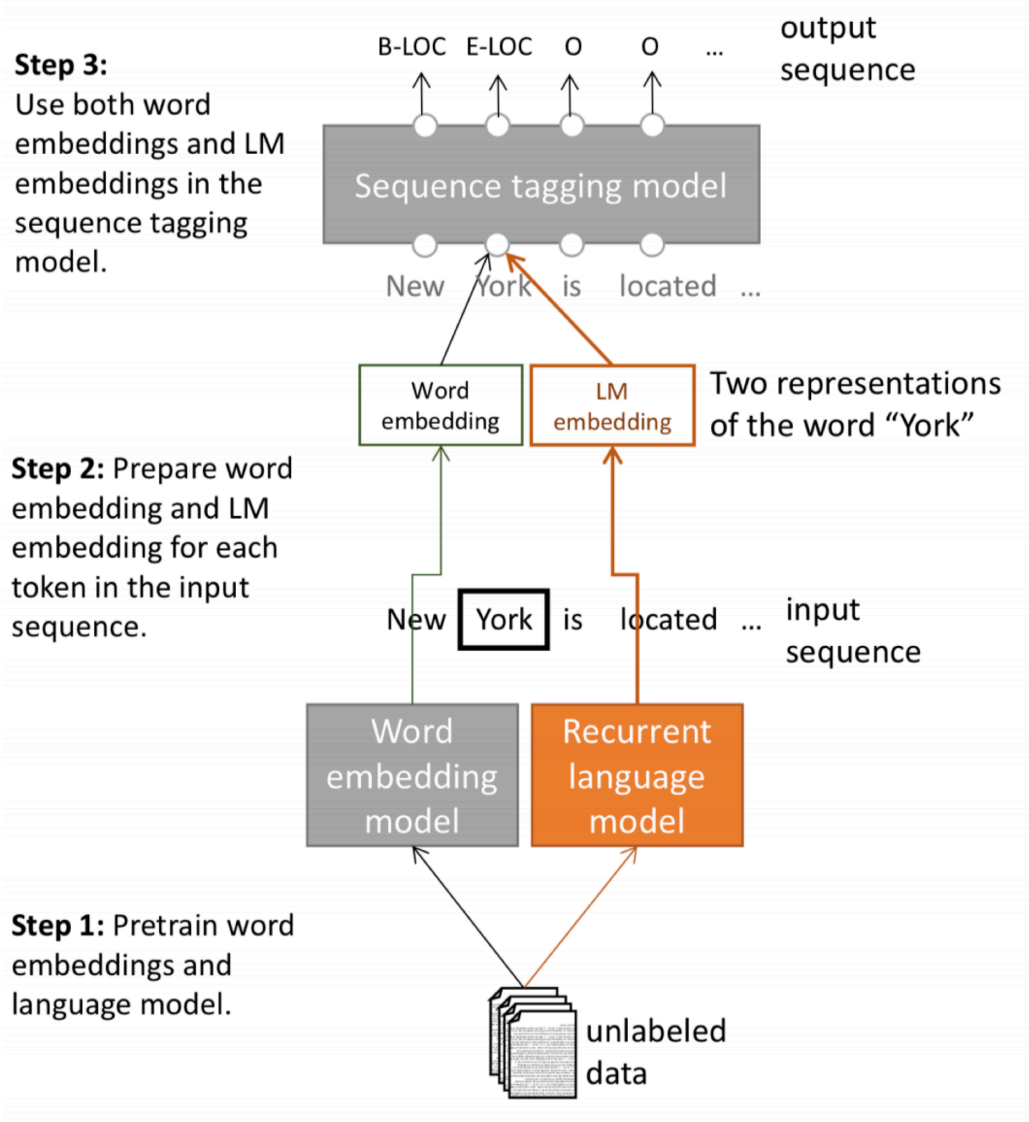


Figure 3.2: Schematic overview of the TagLM model. The NER task is based on both static word embeddings as well as contextualized word representations from the hidden states of a pre-trained bidirectional recurrent language model. Taken from (Peters et al., 2017)

#### 3.2.2.5 Flair

Another succesful approach to obtain contextual word embeddings was the Flair architecture, developed by Alan Akbik and colleagues in 2018 (Akbik et al., 2018). The general idea of Flair is - again - to obtain contextual word representations from a large, pre-trained language model that can subsequently be used to train a supervised NLP task such as NER.

In contrast to ELMo, the authors used a character-based BiLSTM LM, i.e. the model was trained to predict a character given the previous characters in the sequence (and *vice versa*, since it was a bidirectional LM) - without given any explicit notion of words. When a lot of training data is provided, these models have been shown to succesfully internalize linguistic concepts such as words, sentences, grammar, punctuation and even more complex, long-range linguistic dependencies (Graves, 2014; Sutskever et al., 2014). Advantages of using a character-level language model are the improved handling of misspelled and rare words. In addition, since the "vocabulary" of these models is fixed and very compact (characters instead of words), these models are very efficient to train, independent from tokenization. By concatenating the internal states of the language model from both the forward and backward LSTM, they obtained context-specific word vectors, as illustrated in Figure 3.3.

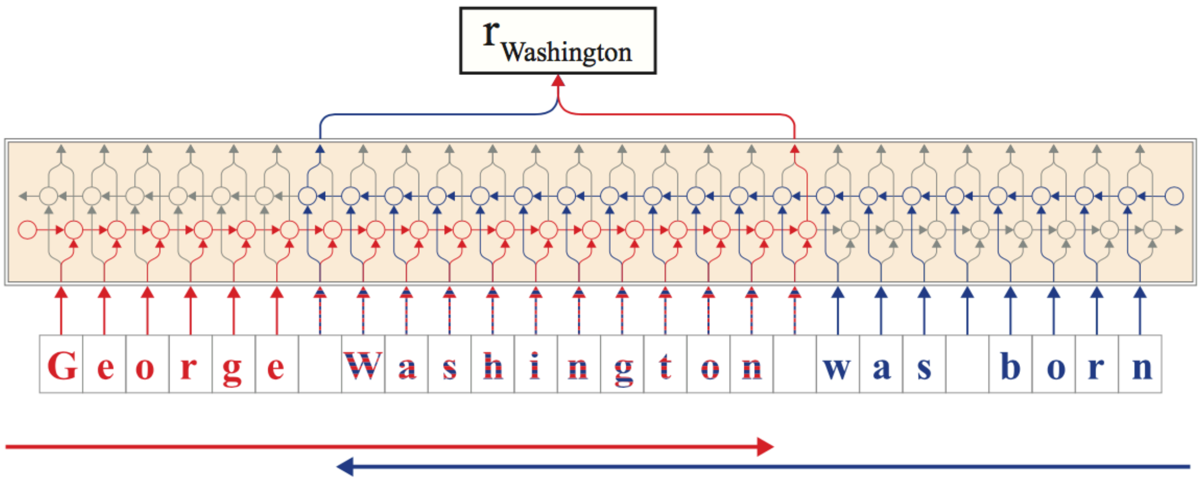


Figure 3.3: The character-based bi-directional Flair LM yields word representations (**r**) that contain information from the word itself, as well as from surrounding characters in the sentence, thus encoding information from the entire sequence. Taken from (Akbik et al., 2018).

They proposed an architecture for NER that was based on their character-based LM and a downstream BiLSTM-CRF classifier. The authors experimented with concatenating their contextual character-based word representations with 'static' GloVe word embeddings as well as the task-specific character-feature embeddings discussed previously. They demonstrated that a stacked embedding containing the proposed character-level contextualized word embeddings as well as the trainable character-feature embeddings and static GloVe embeddings resulted in state-of-the-art performance on the CoNLL 2003 NER task (Akbik et al., 2018).

In addition to training the aforementioned language model, the authors also released an open source PyTorch-based NLP library (Akbik et al., 2019) that supports a wide range of pre-trained embeddings and LMs. The high training efficiency of the character-based LMs is one of the main reasons that several Flair LMs have been pretrained in different languages. For the experiments reported in this thesis, we used the original (English) Flair model described here, as well as the Dutch, French and multilingual versions in different configurations with other static or task/specific embeddings, as explained below (Table 3.4). The Dutch, French and multilingual versions of Flair were trained by members of the community and made freely available in the Flair library[3].

#### 3.2.2.6 BERT

One of the main factors driving improvement in the performance of deep neural LMs is their scale or the amount of training data they are able to learn from. Neural language modelling is an unsupervised task which means that they are not restricted by the scarcity of human-annotated data. Wikipedia or news articles provide an enormous amount of qualitative training data, which means that much of the progress in the last few years can be attributed to the efficiency and scalability of the training algorithms. Even though RNNs like LSTMs provide excellent LMs, they have a major disadvantage: the input of each hidden state relies on the output of the preceding hidden state, making them very slow to train and difficult to parallelize. A major breakthrough with respect to training efficiency of neural LMs was the development of the so-called Transformer architecture in 2017 (Vaswani et al., 2017). For an overview of the Transformer architecture we refer to Jay Alammar's excellent post "The Illustrated Transformer"[4]. In short, the Transformer architecture completely omits the recurrent nature of RNNs, while keeping the ability to "memorize" contextual information. This was achieved through the introduction of an encoder-decoder architecture (typically used in machine translation tasks) and a so-called Multi-Head Attention block, a self-attention mechanism that keeps track of the relationship between input and output, allowing the netwerk to consider specific parts of the input sequence (that can be far away) to improve its performance on the LM objective. A crucial advantage of omiting the recurrency is the fact that training can be parallelized. This facilitates scaling and this is essentially what drove the steady increase in the scale of neural LMs during the past few years.

A very succesful application of the Transformer architecture was Google's BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). BERT was innovative in the sense that it used a bidirectional masked LM objective. Instead of predicting a word given the previous words (or *vice versa* for a backward LM objective), they randomly masked 15 % of the words in the corpus. By training the model to predict these masked words from the entire context, the model could jointly learn from both the left and right side of the target word. This is in contrast to a BiLSTM, in the sense that both the forward and backward LM learn independently from either history or future, as illustrated in Fig. 3.4. The contextualized representations learned by BERT have been used extensively in NLP - often through adding a task-specific layer on top and fine-tuning the entire BERT model on a specific NLP task. BERT or some of its many successors (ALBERTA, RoBERTa, XLM, etc…) are able to exceed human-like performance on high-level NLP tasks like question answering[5]. This illustrates that these models are able to

[3]More information on these models can be found in the documentation of flairNLP on GitHub
[4]jalammar.github.io/illustrated-transformer/
[5]SQuAD benchmark leaderbord

Table 3.3: Overview of the dimensions of all English (En), French (Fr), Dutch (Nl) and multilingual (Multi) word vectors used in this thesis.

| Embedding | En | Fr | Nl | Multi |
|---|---|---|---|---|
| **Static** | | | | |
| fastText | 300 | 300 | 300 | |
| BytePair | 100 | 100 | 100 | 3672 |
| **Contextualized** | | | | |
| Flair | 4096 | 2048 | 4096 | 6144 |
| BERT | 768 | 768 [a] | 768 [b] | 4440 |
| **Task-specific** | | | | |
| One hot word type | 300 | 300 | 300 | 300 |
| Character | 50 | 50 | 50 | 50 |

[a] CamemBERT; [b] BERTje

internalize a large amount of linguistic knowledge - in a completely unsupervised way - and achieve a level of natural language understanding that - for some tasks - comes very close or even exceeds the level of natural language understanding that humans are able to acquire throughout their lives.

For the experiments presented in this report, we used, for English, the case-sensitive BERT base model (Huggingface Transformer library[6] reference: `bert-base-cased`), the CamemBERT (`camembert-base`) model for French (Martin et al., 2019) and BERTje (`wietsedv/bert-base-dutch-cased`) for Dutch (de Vries et al., 2019). Multilingual BERT (mBERT) (`bert-base-multilingual-cased`) was used to obtain multilingual embeddings. We did not perform any fine-tuning of the weights of the BERT models, but obtained contextualized token representations by extracting the activations from the top layer of the Transformer models. Since the BERT models are based on subword tokenization and the NER model operates on the token-level, we used the representations of the first subtoken as token-level embeddings, similar to the feature-based approach described in the original BERT paper (Devlin et al., 2019). The dimensions of the obtained embeddings from the different BERT-models are given in Table 3.3.

[6] GitHub: huggingface/transformers

Table 3.4: Overview of the different embeddings evaluated in the context of English, Dutch, French and multilingual NER. The columns indicate the type of LM used to obtain contextualized embeddings (if any). The different configurations w.r.t. static and/or task-specific embedding types are indicated by the different rows. *Char: character-feature embeddings; OHE: one hot word type embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; multilingual embeddings are prefixed with 'm'*

| No contextualized embeddings | BERT | Flair | BERT + Flair |
|---|---|---|---|
| **English** | | | |
| n.a. | BERT | Flair | BERT + Flair |
| Char | BERT + Char | Flair + Char | BERT + Flair + Char |
| OHE | BERT + OHE | Flair + OHE | BERT + Flair + OHE |
| BPEmb (En) | BERT + BPEmb (En) | Flair + BPEmb (En) | BERT + Flair + BPEmb (En) |
| fastT (En) | BERT + fastT (En) | Flair + fastT (En) | BERT + Flair + fastT (En) |
| All [a] | BERT + All [a] | Flair + All [a] | BERT + Flair + All [a] |
| **Dutch** | | | |
| n.a. | BERTje | Flair (Nl) | BERTje + Flair (Nl) |
| Char | BERTje + Char | Flair (Nl) + Char | BERTje + Flair (Nl) + Char |
| OHE | BERTje + OHE | Flair (Nl) + OHE | BERTje + Flair (Nl) + OHE |
| BPEmb (Nl) | BERTje + BPEmb (Nl) | Flair (Nl) + BPEmb (Nl) | BERTje + Flair (Nl) + BPEmb (Nl) |
| fastT (Nl) | BERTje + fastT (Nl) | Flair (Nl) + fastT (Nl) | BERTje + Flair (Nl) + fastT (Nl) |
| All [b] | BERTje + All [b] | Flair (Nl) + All [b] | BERTje + Flair (Nl) + All [b] |
| **French** | | | |
| n.a. | CamemBERT | Flair (Fr) | CamemBERT + Flair (Fr) |
| Char | CamemBERT + Char | Flair (Fr) + Char | CamemBERT + Flair (Fr) + Char |
| OHE | CamemBERT + OHE | Flair (Fr) + OHE | CamemBERT + Flair (Fr) + OHE |
| BPEmb (Fr) | CamemBERT + BPEmb (Fr) | Flair (Fr) + BPEmb (Fr) | CamemBERT + Flair (Fr) + BPEmb (Fr) |
| fastT (Fr) | CamemBERT + fastT (Fr) | Flair (Fr) + fastT (Fr) | CamemBERT + Flair (Fr) + fastT (Fr) |
| All [c] | CamemBERT + All [c] | Flair (Fr) + All [c] | CamemBERT + Flair (Fr) + All [c] |
| **Multilingual** | | | |
| n.a. | mBERT | mFlair | mBERT + mFlair |
| Char | mBERT + Char | mFlair + Char | mBERT + mFlair + Char |
| OHE | mBERT + OHE | mFlair + OHE | mBERT + mFlair + OHE |
| mBPEmb | mBERT + mBPEmb | mFlair + mBPEmb | mBERT + mFlair + mBPEmb |
| All [d] | mBERT + All [d] | mFlair + All [d] | mBERT + mFlair + All [d] |

[a] Char + OHE + BPEmb (En) + fastT (En) [b] Char + OHE + BPEmb (Nl) + fastT (Nl) [c] Char + OHE + BPEmb (Fr) + fastT (Fr) [d] Char + OHE + mBPEmb

## 3.3 NER classifier

NER is essentially a classification problem. The sequential nature of languange renders some algorithms more suited for the task than others. Today, state-of-the-art systems for sequence labeling tasks like NER are typically based on a BiLSTM with a Conditional Random Field (CRF) decoding layer (Lafferty et al., 2001), as illustrated in Figure 3.4.

For a given sequence of words $(w_1,\ ...,\ w_m)$, the BiLSTM provides a representations $\mathbf{H} = (\mathbf{h}_1, ...,\ \mathbf{h}_m)$ with $\mathbf{h}_i = [\mathbf{h}_i^f;\ \mathbf{h}_i^b]$ i.e. the concatenated output states of the $i$-th unit of the forward and backward LSTM (Figure 3.4). One could simply use the representation $\mathbf{h}_i$ as a feature vector to predict the tag for the $i$-th output, i.e. model the conditional distributions $P(y_i \mid \mathbf{h}_i)$ independently for $i = (1,\ ...,\ m)$ by passing the BiLSTM output states through a nonlinear softmax activation function (Goodfellow et al., 2016). Alternatively, one could model the conditional distribution of an entire sequence of output labels $\mathbf{y} = (y_1,\ ...,\ y_m)$ given a sequence of word representations $\mathbf{H}$, i.e. $P(\mathbf{y} \mid \mathbf{H}) = P(y_1,\ ...,\ y_m \mid \mathbf{h}_1,\ ...,\ \mathbf{h}_m)$. By using this so-called CRF over all possible tag sequences and jointly model the full sequence of labels for some input sequence, the model is able to consider the strong dependencies between neighboring tags (e.g. B-LOC cannot be followed by I-PER), which has been shown to generally improve performance on NER tasks (Huang et al., 2015; Reimers and Gurevych, 2017). The parametrization and estimation procedure of the CRF in the context of NER is described in (Ma and Hovy, 2016). For the experiments reported in this thesis, we used the BiLSTM-CRF implementation from the Flair NLP library (version 0.4.5 / 0.5.0). More information on this specific implementation can be found in (Akbik et al., 2018; Huang et al., 2015).

For the benchmark datasets (CoNLL2002, CoNLL2003, WikiNER), the model hyperparameters were based on the best known configuration for the CoNLL tasks as reported by the authors of the Flair library. In what follows we provide the parameter names of the `SequenceTagger` and `ModelTrainer` classes of the Flair library. A BiLSTM-CRF model with a forward and backward hidden layer of 256 units (`hidden_size`) was used. Training was done using stochastic gradient descent with a `mini_batch_size` of 32 instances, i.e. at each iteration, 32 sentences from the training set were propagated through the network (sentences were shuffled at each epoch). The initial `learning_rate` was set to 0.1. Early stopping criteria were specified by setting `patience` = 3 and `anneal_factor` = 0.5 which means that the learning rate was multiplied by a factor 0.5 when the validation F1-score did not improve for 3 consecutive epochs. When the learning rate dropped below the `min_learning_rate` of 0.0001 or after 100 epochs (as specified by the `max_epochs` parameter), the training procedure was stopped. The final model was refitted on the training and validation set and performance was estimated through computing the micro-average precision, recall and F1-score (cf infra) on the held-out test set. The approach was similar for the Faktion datasets, with some minor changes in terms of the NER classifier architecture and the early stopping criteria to optimize performance for these datasets. The hyperparameters for both the benchmark and Faktion datasets are summarized in table 3.5.

All models were trained using the cloud computational environment of Kaggle, either on a CPU (4 cores, 16 Gb RAM) or on a single NVIDIA Tesla P100 GPU (2 CPU cores, 16 Gb RAM) (Python 3.7.6). We made use of the Kaggle API to automatically configure Python scripts for each experiment and push it to the Kaggle environment from a simple command line interface. All code and information to reproduce the experiments can be

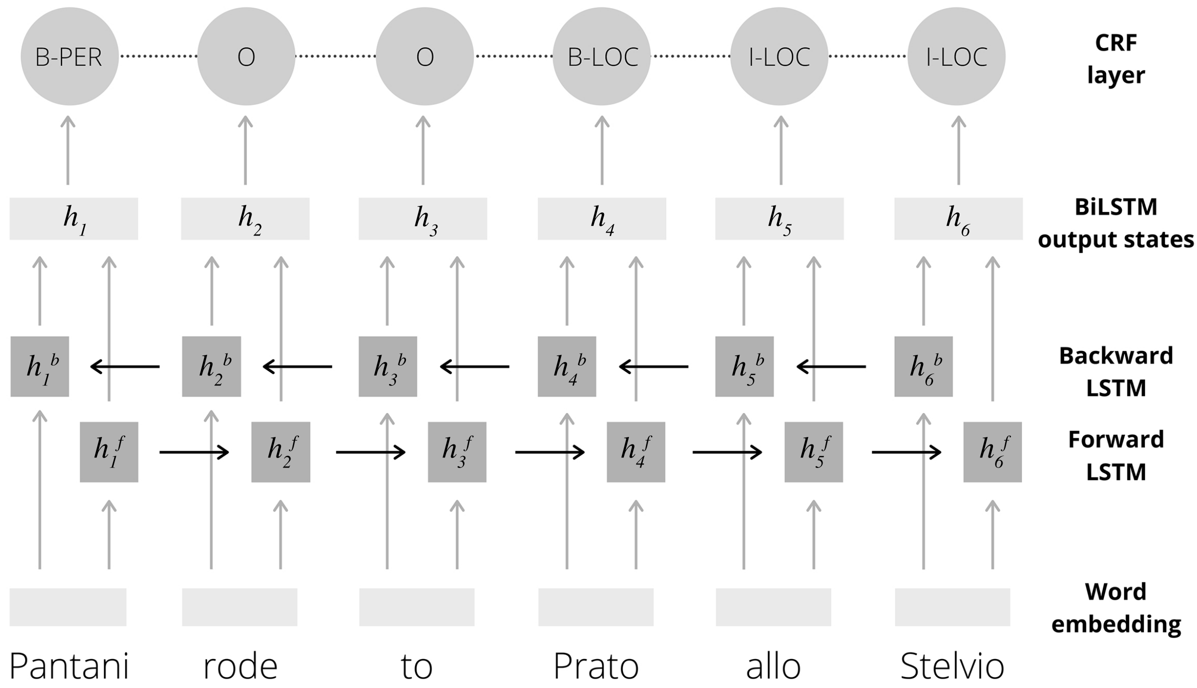


Figure 3.4: A simplified representation of the BiLSTM-CRF architecture that was used to perform NER for the experiments reported in this thesis. The input of the BiLSTM are either static, task-specific or contextualized word embeddings (or a concatenation of different embedding types). The forward ($\mathbf{h}_i^f$) and backward ($\mathbf{h}_i^b$) output states of the BiLSTM are concatenated into a contextualized representation ($\mathbf{h}_i = [\mathbf{h}_i^f; \mathbf{h}_i^b]$) that is used to predict the correct tag sequence using a Conditional Random Field (CRF), as described in the text.

Table 3.5: Overview of the hyperparameter configuration for all NER experiments on the benchmark datasets and Faktion datasets. The terminology used here is consistent with the parameters of the `SequenceTagger` and `ModelTrainer` classes of the Flair NLP library. Further details can be found in the documentation of the Flair NLP library or in the GitHub repository of this thesis.

| Parameter | Benchmark | Faktion |
|---|---|---|
| hidden_size | 256 | 32 |
| learning_rate | 0.1 | 0.1 |
| anneal_factor | 0.5 | 0.8 |
| patience | 3 | 3 |
| min_learning_rate | 1e-04 | 1e-04 |
| mini_batch_size | 32 | 8 |
| max_epochs | 100 | 100 |

found on the Github repository (arthur-arthur/NER) of this thesis.

## 3.4 NER evaluation

According to the CoNLL guidelines, evaluation of NER is performed at the entity-level, not at the token or word-level (to clarify this, an example is provided at the end of this paragraph). This has some consequences on the type of metrics that are used to evaluate NER performance, as well as on their interpretation.

Firstly, since entity-level evaluation is based on the sets of predicted and true entities, there is no clear definition of a “true negative” entity (Esuli and Sebastiani, 2010). Hence, evaluation metrics that are a function of the number of true negatives (ROC analysis, classification accuracy, Matthews Correlation Coefficient,...) are not very meaningful in this context. For this reason, NER performance is commonly evaluated in terms of entity-level precision, recall and F1-score. Precision gives the proportion of all predicted entities that are indeed correct:

$$P = \frac{TP}{TP + FP}$$

with TP and FP being true positives and false positives, respectively. Recall indicates how much of the entities in the dataset were correctly predicted:

$$R = \frac{TP}{TP + FN}$$

with FN being false negatives. Note that one can easily design a system that achieves high recall by predicting all instances as entities such that $FN = 0$. This will typically result in a high number of false positives and, hence, a low precision. Conversely, when only a single entity from the entire dataset is correctly predicted, $P = 1$. In that case, the number of false negatives is typically high and, hence, recall will be low. The goal is therefore to find a system that has an optimal balance between precision and recall, which is commonly expressed in terms of the F1-score - the gold standard evaluation metric for NER (Tjong Kim Sang and De Meulder, 2003). The F1-score is defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2PR}{P + R}$$

To illustrate how gold standard entity-level evaluation of NER systems is performed, we provide the following example sentence (for simplicity, the example includes only a single entity class, this extends naturally to cases with multiple classes):

- **Gold sentence:**
  “From Bormio [B-LOC] to Prato [B-LOC] allo [I-LOC] Stelvio [I-LOC]”.
- **Predicted sentence:**
  “From Bormio [B-LOC] to Prato [B-LOC] allo Stelvio [B-LOC]”

The set of true entities thus includes two instances: “Bormio” and “Prato allo Stelvio”. The NER system correctly identified the token “Bormio” as a location, i.e. this is a true positive (TP) prediction. The system failed to correctly predict the segment "Prato allo Stelvio" as a location spanning three tokens, i.e. this is a false negative (FN) prediction. In addition, the NER tagger predicted "Prato [B-LOC]" and "Stelvio [B-LOC]" as two separate entities, while these were not in the set of true entities. Hence, these are two false positive (FP) predictions. Overall, the precision for this example is $\frac{1}{1+2} = 0.33$, the recall is $\frac{1}{1+1} = 0.5$ and the F1-score is $\frac{2 \times 0.33 \times 0.50}{0.33+0.50} = 0.40$.

When there are multiple entity classes to predict (which is the case for most NER systems), the overall performance of the system can be expressed in terms of the macro-average or micro-average precision, recall or F1-scores. Macro-averaging involves simply averaging the scores per entity class, while the micro-average refers to the weighted average of the class scores, with the weights being determined by the number of instances per class. Since the latter takes the class balance into account. NER performance is typically evaluated and reported in terms of the micro-average F1-score. We will adhere to this convention and report the micro-average precision, recall and F1-scores for the NER experiments performed during this thesis. For the benchmark datasets, only a single training run was performed. When multiple training runs were performed (i.e. for the Faktion datasets), the F1-scores were reported as mean $\pm$ standard error of the mean (sem) for $k = 5$ runs. All scores reported in this thesis were computed on the held-out test set.

# Chapter 4

# Results

## 4.1 Benchmark datasets

### 4.1.1 CoNLL2003 - English

The best results obtained on the CoNLL2003 task (Fig 4.1) using state-of-the-art contextualized embeddings from pretrained LMs were similar to the state-of-the-art results reported in literature (Table 4.1). As a proof of concept to demonstrate the added value of "transfer learning" in the context of NER, the BiLSTM-CRF classifier was trained without any input from pretrained (either static or contextualized) embeddings, i.e. only using randomly initialized embeddings that were updated during training to minimize the loss w.r.t. to the NER objective. Both the one hot word type and character-feature embeddings performed poorly, with F1-scores of 74.0 % and 75.4 %, respectively (Fig 4.1). Performance improved drastically when using static BytePair or fastText embeddings alone (Fig 4.1). However, as expected, best performance was provided when contextualized representations from pretrained LMs were used, with the performance obtained with embeddings based on BERT, Flair or both being similar when they were concatenated into large vectors containing all static and task-specific embedding types evaluated here (Table 4.1).

While the monolingual embeddings appeared to outperform the multilingual embeddings, the differences between large stackings of respectively BERT and mBERT embeddings were surprisingly small (F1-scores of 92.1 % vs. 91.4 %, respectively), with mBERT-based embeddings being clearly superior to mFlair-based embeddings (Table 4.1). Interestingly, adding mFlair embeddings to the stacked embedding consisting of mBytePair, OHE and character embeddings (F1-score: 88.0 %) did not improve performance (F1-score: 87.9 %) (Fig 4.1, Table 4.1).

### 4.1.2 CoNLL2002 - Dutch

Results were similar for the Dutch CoNLL2002 task (Fig 4.2, Table 4.2) in the sense that task-specific representations (i.e. character embeddings (F1: 67.8 %) or OHE (F1: 61.7 %)) were clearly inferior to embeddings obtained from pretrained LMs, with the best-performing monolingual system being based on the large (i.e. 5614-dimensional) concatenations of embeddings from both BERTje and Dutch Flair embeddings, stacked with

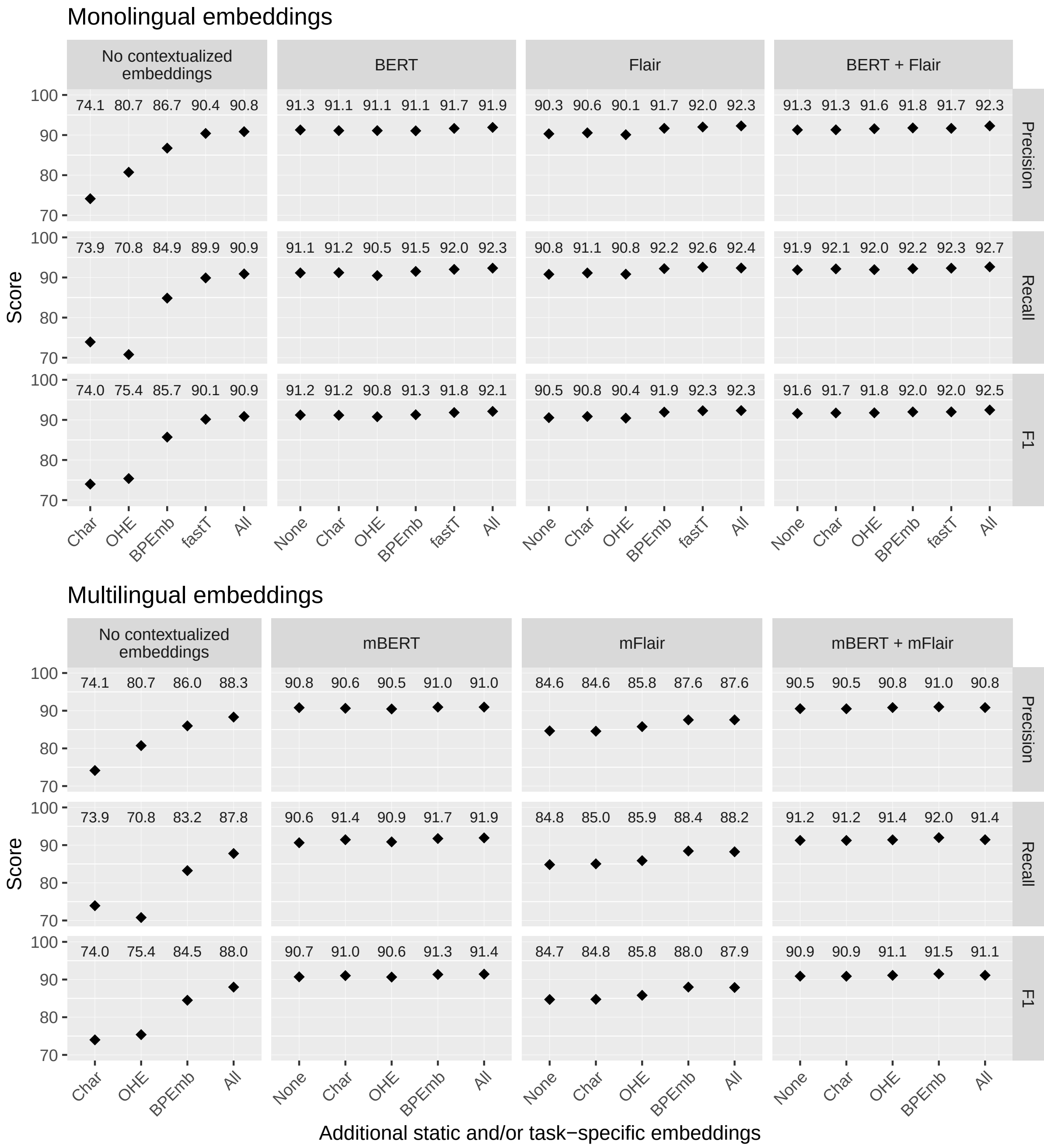


Figure 4.1: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the English CoNLL2003 NER task. The results are grouped according to the constituents of each (stacked) embedding, consistent with Table 3.4. The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, the results for the language-independent OHE and Char embeddings are indicated on both the upper (monolingual) and lower (multilingual) panels. *Char: character embeddings; OHE: One Hot Embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; All: all embeddings indicated on the x-axis for this panel concatenated into a single vector.*

Table 4.1: Results acquired on the English CoNLL2003 task with monolingual and multilingual NER systems. The results for the best-performing model for the original CoNLL2003 shared task is included for comparison (Florian et al., 2003). A complete overview of all results is provided in Table A.1.

| Description | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual** | | | |
| CoNLL2003 best - IBM Florian [a] | 89.0 | 88.5 | 88.7 |
| Flair - BiLSTM-CRF [b] | | | 93.1 |
| BERT base - finetuned [c] | | | 92.4 |
| [all-en]: fastT + BPEmb + Char + OHE | 90.8 | 90.9 | 90.9 |
| BERT | 91.3 | 91.1 | 91.2 |
| BERT + [all-en] | 91.9 | 92.3 | 92.1 |
| Flair | 90.3 | 90.8 | 90.5 |
| Flair + [all-en] | 92.3 | 92.4 | 92.3 |
| BERT + Flair | 91.3 | 91.9 | 91.6 |
| BERT + Flair + [all-en] | 92.3 | 92.7 | **92.5** |
| **Multilingual** | | | |
| mBERT - finetuned [d] | | | 92.0 |
| [all]: mBPEmb + Char + OHE | 88.3 | 87.8 | 88.0 |
| mBERT | 90.8 | 90.6 | 90.7 |
| mBERT + [all] | 91.0 | 91.9 | **91.4** |
| mFlair | 84.6 | 84.8 | 84.7 |
| mFlair + [all] | 87.6 | 88.2 | 87.9 |
| mBERT + mFlair | 90.5 | 91.2 | 90.9 |
| mBERT + mFlair + [all] | 90.8 | 91.4 | 91.1 |

*References*

[a] Florian et al (2003)
[b] Akbik et al (2018)
[c] Devlin et al (2019)
[d] Wu et al (2019)

Table 4.2: Results acquired on the Dutch CoNLL2002 task with monolingual and multilingual NER systems. The best-performing model for the original CoNLL2002 shared task (Carreras et al., 2002) is included for comparison. A complete overview of all results is provided in Table A.2.

| Description | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual** | | | |
| CoNLL2002 best - AdaBoost [a] | 77.8 | 76.3 | 77.1 |
| Flair NL - BiLSTM-CRF [b] | | | 89.6 |
| fine-tuned BERTje [c] | | | 90.2 |
| [all-nl]: fastT + BPEmb + Char + OHE | 88.1 | 86.8 | 87.4 |
| BERTje | 91.8 | 91.3 | 91.5 |
| BERTje + [all-nl] | 93.1 | 92.2 | 92.6 |
| Flair | 86.9 | 86.2 | 86.4 |
| Flair + [all-nl] | 90.2 | 89.3 | 89.7 |
| BERTje + Flair | 92.4 | 92.5 | 92.4 |
| BERTje + Flair + [all-nl] | 93.1 | 93.0 | **93.0** |
| **Multilingual** | | | |
| fine-tuned mBERT [d] | | | 90.9 |
| [all]: mBPEmb + Char + OHE | 82.8 | 80.0 | 81.3 |
| mBERT | 89.2 | 88.5 | 88.8 |
| mBERT + [all] | 90.6 | 89.4 | 90.0 |
| mFlair | 79.8 | 78.8 | 79.1 |
| mFlair + [all] | 84.7 | 82.6 | 83.6 |
| mBERT + mFlair | 89.8 | 89.4 | 89.5 |
| mBERT + mFlair + [all] | 90.4 | 89.8 | **90.1** |

*References*
[a] Carreras et al 2002
[b] GitHub: stefan-it/flair-experiments
[c] GitHub: wietsedv/bertje
[d] Wu et al (2019)

Dutch fastText, Dutch BytePair and task-specific OHE and character embeddings (F1: 93.0 %, Table 4.2). Even though this is the result of only a single training run, all the monolingual embeddings based on BERTje consistently outperformed the result reported in the BERTje paper itself (F1: 90.2 %) (de Vries et al., 2019) as well as the current best result on the CoNLL2002 task, obtained by the fine-tuned mBERT model (F1-score: 90.9 %) (Wu and Dredze, 2019). In contrast to the English CoNLL2003 task, monolingual embeddings from the Flair model performed poorly as compared to the BERT-based embeddings, with the difference being even more pronounced when comparing mFlair-based embeddings with mBERT-based embeddings (Table 4.2).

### 4.1.3 WikiNER - French

To evaluate how monolingual and multilingual embeddings performed on a French NER task, we used a small subset of the French WikiNER dataset. Even though the en-

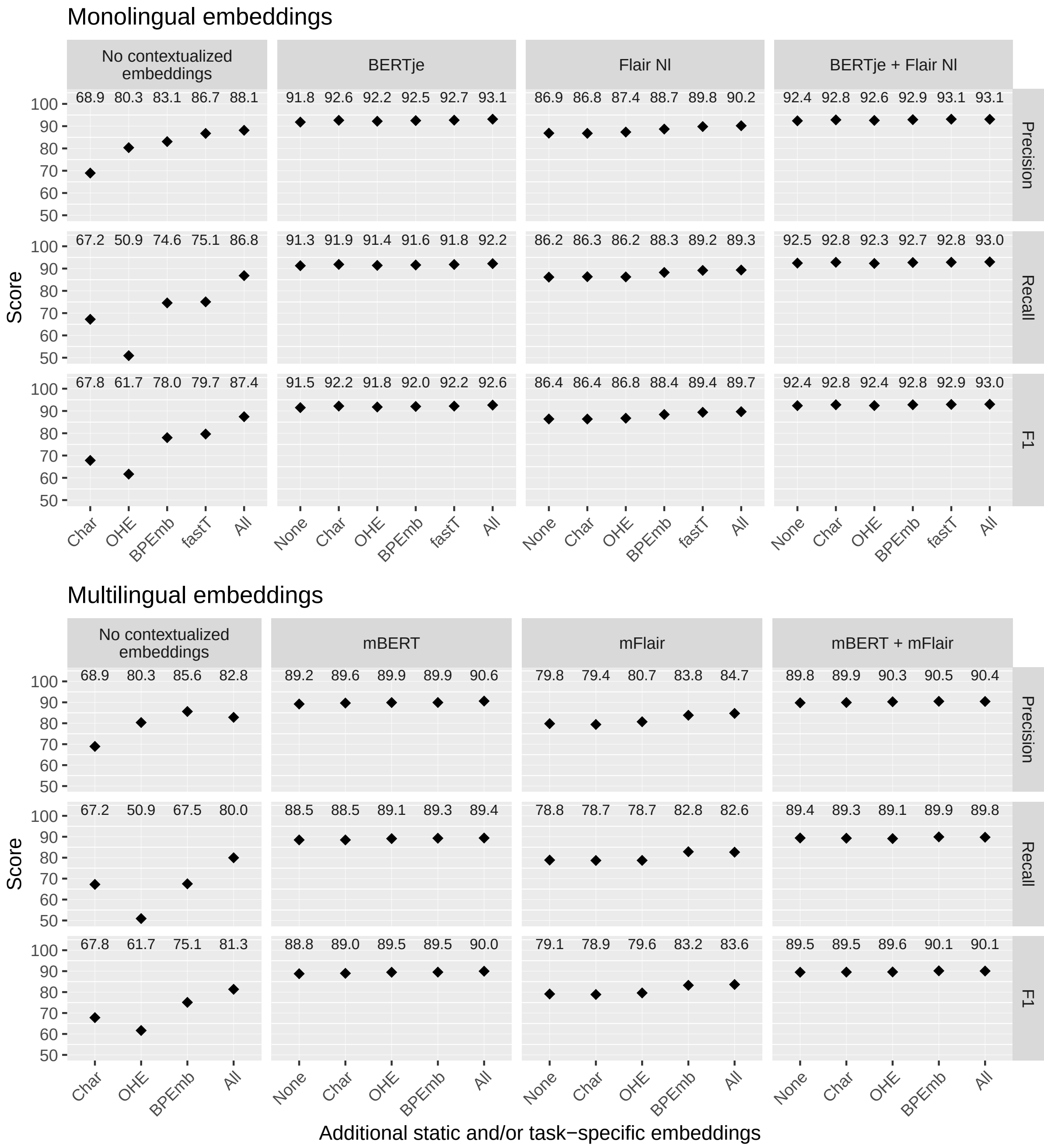


Figure 4.2: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the Dutch CoNLL2002 NER task. The results are grouped according to the constituents of each (stacked) embedding, consistent with Table 3.4. The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, the results for the language-independent OHE and Char embeddings are indicated on both the upper (monolingual) and lower (multilingual) panels. *Char: character embeddings; OHE: One Hot Embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; All: all embeddings indicated on the x-axis for this panel concatenated into a single vector.*

Table 4.3: Overview of results acquired on the French WikiNER task with monolingual and multilingual NER systems. Note that we used a subset consisting of only 10 % of the entire WikiNER dataset. A complete overview of all results is provided in Table A.3.

| Description | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual** | | | |
| SpaCy fr [a] | 82.2 | 81.6 | 81.9 |
| Flair [b] | 87.9 | 87.7 | 87.8 |
| [all-fr]: fastT + BPEmb + Char + OHE | 84.0 | 83.9 | 83.9 |
| CamemBERT | 84.5 | 84.8 | 84.6 |
| CamemBERT + [all-fr] | 86.3 | 86.4 | 86.3 |
| Flair | 79.1 | 80.0 | 79.4 |
| Flair + [all-fr] | 85.2 | 85.3 | 85.2 |
| CamemBERT + Flair | 86.2 | 86.2 | 86.1 |
| CamemBERT + Flair + [all-fr] | 87.4 | 87.4 | **87.4** |
| **Multilingual** | | | |
| SpaCy XX [c] | 80.3 | 79.5 | 79.9 |
| [all]: mBPEmb + Char + OHE | 80.2 | 80.8 | 80.4 |
| mBERT | 85.5 | 85.4 | 85.4 |
| mBERT + [all] | 85.3 | 85.0 | 85.1 |
| mFlair | 74.4 | 75.7 | 74.7 |
| mFlair + [all] | 81.0 | 81.6 | 81.2 |
| mBERT + mFlair | 84.8 | 84.9 | 84.8 |
| mBERT + mFlair + [all] | 86.2 | 86.3 | **86.2** |

*References*
[a] GitHub: flairNLP/flair/issues/238
[b] GitHub: flairNLP/flair/issues/238
[c] spacy.io/models/xx

tity classes (i.e. PER, ORG, LOC and MISC) were identical to the CoNLL2002 and CoNLL2003 datasets, performance was - on average - lower for the WikiNER dataset. However, the general trend that stacking multiple embedding types together typically improves performance was also observed here (Fig 4.3). This was less pronounced for monolingual embeddings based on CamemBERT or multilingual embeddings based on mBERT, where performance was already relatively high at baseline (i.e. when no additional static or task-specific embeddings were included) (Table 4.2).

In general - for all three monolingual datasets - high-dimensional vectors based on contextualized representations obtained from pretrained LMs clearly improved performance as compared to high-dimensional stackings of static and task-specific representations. BERT-based embeddings generally outperformed Flair-based embeddings, except for monolingual English NER, where Flair- and BERT-based embeddings performed similarly.

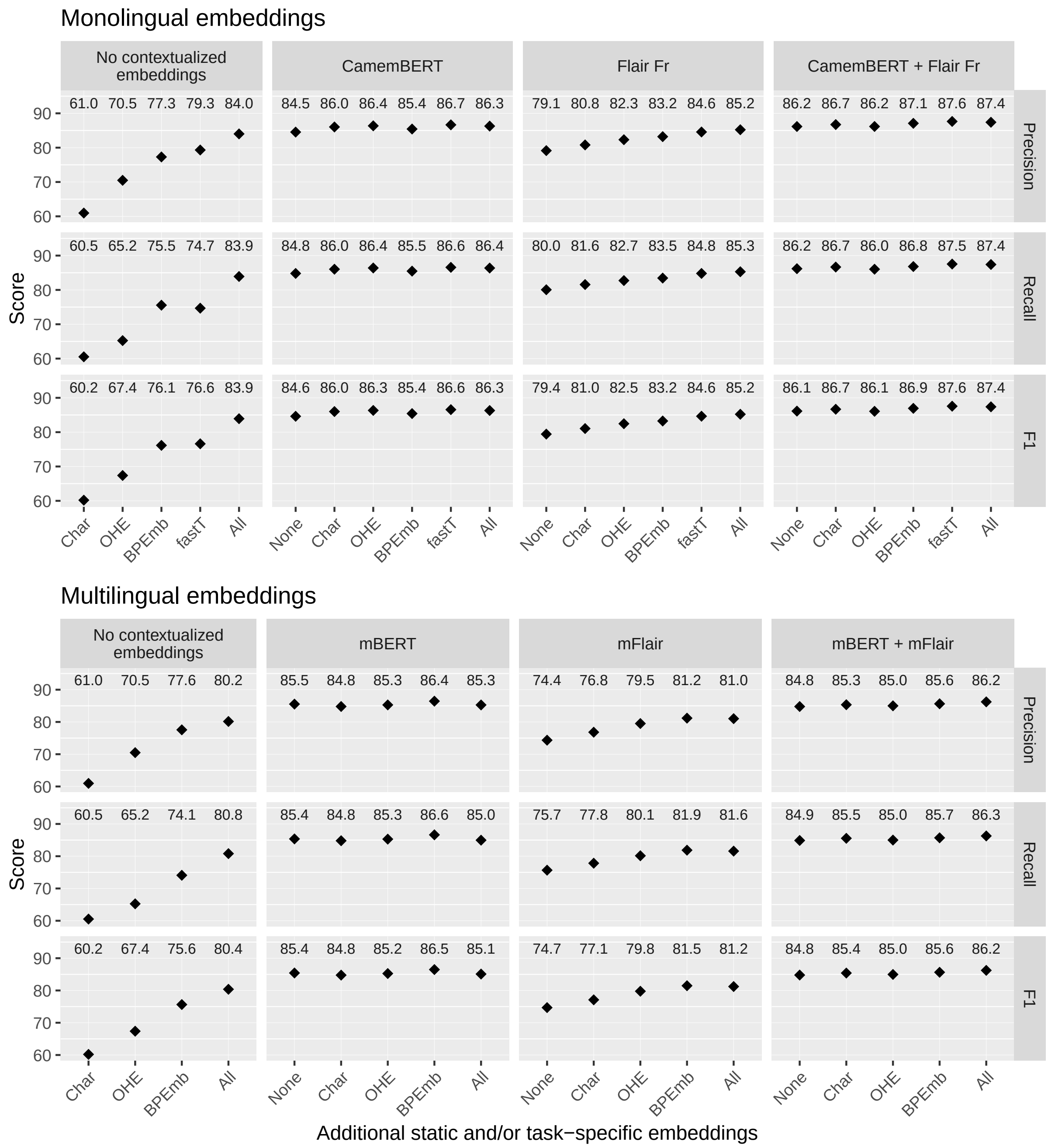


Figure 4.3: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the French WikiNER task. The results are grouped according to the constituents of each (stacked) embedding, consistent with Table 3.4. The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, the results for the language-independent OHE and Char embeddings are indicated on both the upper (monolingual) and lower (multilingual) panels. *Char: character embeddings; OHE: One Hot Embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; All: all embeddings indicated on the x-axis for this panel concatenated into a single vector.*

### 4.1.4 Trilingual NER

As expected, monolingual embeddings outperformed multilingual embeddings on the aforementioned (monolingual) NER tasks, which might reflect the lack of language-specific knowledge of the multilingual embeddings. The difference, however, was small for multilingual representations based on mBERT embeddings. To test the hypothesis that the multilingual knowledge acquired by mBERT during pretraining is particularly beneficial when the input data is multilingual, we combined random subsets of the CoNLL2002, CoNLL2003 and WikiNER dataset into a single, multilingual dataset. We evaluated NER performance using all monolingual and multilingual embeddings used for the monolingual NER tasks. Evaluating the different monolingual embeddings on the multilingual dataset provides an indication on how NER performance might suffer from incorrect predictions by an upstream language-prediction system in a document annotation pipeline like Metamaze (cf supra).

For clarity, only the F1-scores of a subset of the results are included in Fig 4.4. Consistently with previous results, the precision, recall and F1-score for either the different contextualized embeddings as such, or concatenated with all monolingual or multilingual static/task-specific embeddings are provided in Table 4.4. Task-specific representations (OHE and Character embeddings) performed, as expected, poorly, with F1-scores of 60.4 % and 60.8 %, respectively (Table A.4). Adding contextualized embeddings from pretrained LMs improved results drastically, again demonstrating that the acquired knowledge of the LM provided representations that encoded information highly relevant for the NER classifier to perform well - even when this LM was trained in the language that represented merely one third of the entire dataset.

For all monolingual embeddings, concatenating both Flair and BERT embeddings together improved performance as compared to the representations obtained from either Flair- or BERT-based embeddings alone (Table 4.4). When increasing the dimensionality of the BERT + Flair stacked embeddings further through concatenating fastText and BytePair embeddings (in the respective languages) and task-specific OHE and character embeddings, performance increased further to roughly 85 % for all monolingual systems (Fig 4.4). Interestingly, even mBERT embeddings alone performed better on this multilingual dataset (F1-score: 86.5 %), confirming our hypothesis that, indeed, pretraining on a multilingual dataset improves performance for multilingual NER as compared to the monolingual LMs. An additional (slight) performance increase was achieved through stacking mBERT embeddings with mBytePair, OHE and character embeddings (F1-score: 87.5 %) (Table 4.4). The performance of mFlair-based embeddings was again low, in general. Not only compared to mBERT-based embeddings, but even compared to monolingual embeddings containing no BERT- or Flair-based embeddings at all (Figure 4.4).

## 4.2 Faktion datasets

As discussed in chapter 3, the Faktion datasets were considerably smaller and more noisy compared to the benchmark datasets presented earlier. Therefore, it was not very surprising that performance on the Dutch Faktion dataset was - on average - much lower as compared to the CoNLL 2002 task (Fig 4.5, Table 4.5). However, there was a clear ad-

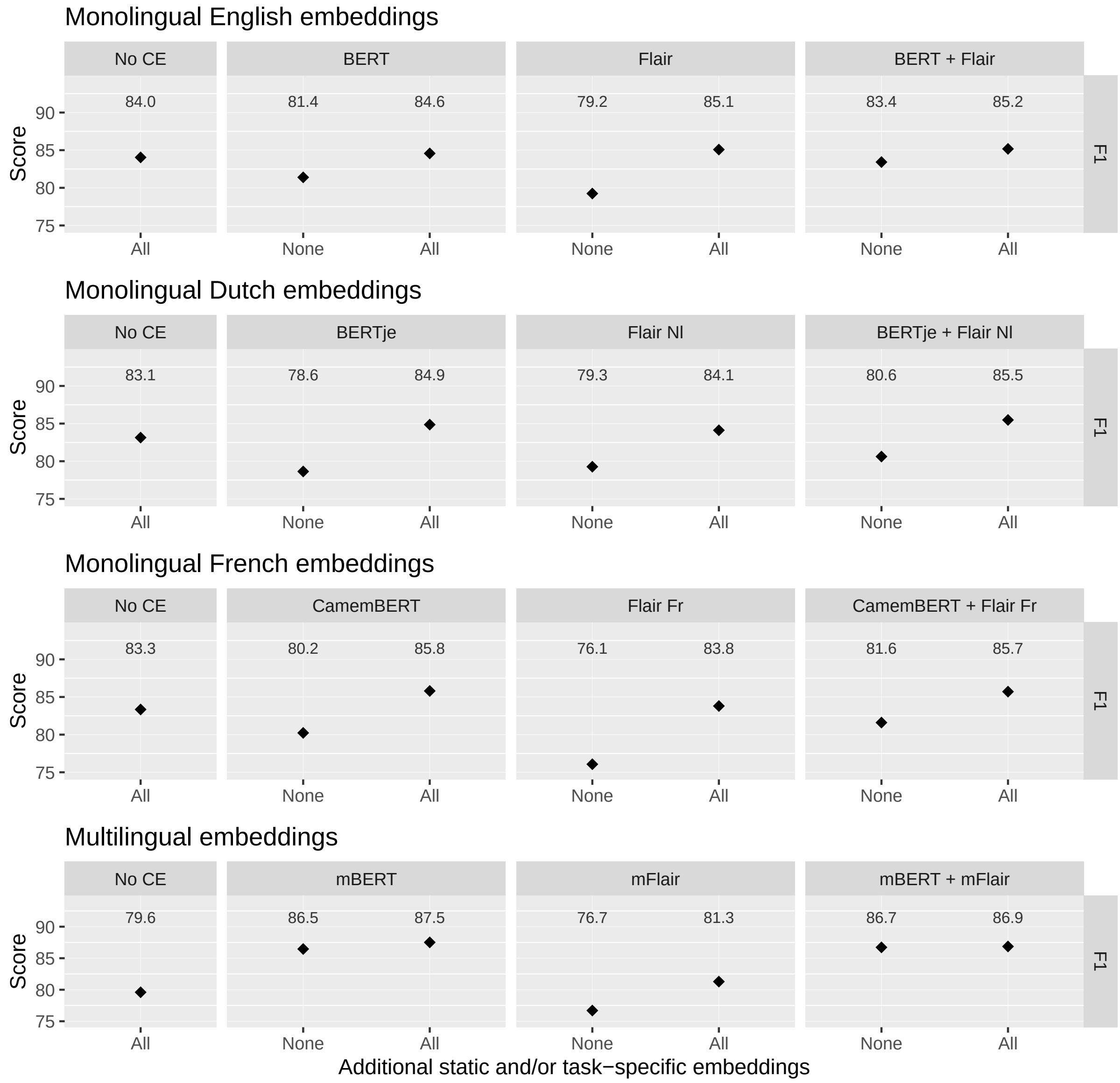


Figure 4.4: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the multilingual (English + Dutch + French) benchmark dataset. The results are grouped according to the constituents of each (stacked) embedding, consistent with Table 3.4. The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, only the results for experiments that did not include any static/task-specific embeddings (None) and for the experiments where all static/task-specific embeddings were included (All) are shown. *No CE: No contextualized embeddings*

Table 4.4: Results obtained on the multilingual dataset, consisting of a random sample of the aformentioned English, Dutch and French benchmark datasets. A complete overview of all results is provided in Table A.4.

| Embedding type | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual English embeddings** | | | |
| [all-en]: fastT + BPEmb + Char + OHE | 84.4 | 83.8 | 84.0 |
| BERT | 82.0 | 81.0 | 81.4 |
| BERT + [all-en] | 85.0 | 84.2 | 84.6 |
| Flair | 79.5 | 79.2 | 79.2 |
| Flair + [all-en] | 85.2 | 85.1 | 85.1 |
| BERT + Flair | 83.5 | 83.4 | 83.4 |
| BERT + Flair + [all-en] | 85.3 | 85.1 | **85.2** |
| **Monolingual Dutch embeddings** | | | |
| [all-nl]: fastT + BPEmb + Char + OHE | 83.5 | 82.9 | 83.1 |
| BERTje | 80.1 | 77.4 | 78.6 |
| BERTje + [all-nl] | 85.0 | 84.8 | 84.9 |
| Flair | 79.1 | 79.7 | 79.3 |
| Flair + [all-nl] | 84.1 | 84.2 | 84.1 |
| BERTje + Flair | 80.7 | 80.6 | 80.6 |
| BERTje + Flair + [all-nl] | 85.2 | 85.8 | **85.5** |
| **Monolingual French embeddings** | | | |
| [all-fr]: fastT + BPEmb + Char + OHE | 83.8 | 83.0 | 83.3 |
| CamemBERT | 81.0 | 79.6 | 80.2 |
| CamemBERT + [all-fr] | 86.1 | 85.6 | **85.8** |
| Flair | 76.5 | 75.8 | 76.1 |
| Flair + [all-fr] | 84.4 | 83.3 | 83.8 |
| CamemBERT + Flair | 82.0 | 81.2 | 81.6 |
| CamemBERT + Flair + [all-fr] | 86.0 | 85.5 | 85.7 |
| **Multilingual embeddings** | | | |
| [all]: mBPEmb + Char + OHE | 80.2 | 79.1 | 79.6 |
| mBERT | 86.6 | 86.4 | 86.5 |
| mBERT + [all] | 87.6 | 87.4 | **87.5** |
| mFlair | 76.8 | 77.0 | 76.7 |
| mFlair + [all] | 81.2 | 81.5 | 81.3 |
| mBERT + mFlair | 86.6 | 86.9 | 86.7 |
| mBERT + mFlair + [all] | 86.9 | 86.9 | 86.9 |

Table 4.5: Results acquired on the Dutch Faktion NER task with both monolingual and multilingual NER systems. Results are reported as mean ± standard error of the mean (k = 5). A complete overview of all results is provided in Table B.1.

| Embedding type | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual embeddings** | | | |
| [all]: fastT + BPEmb + Char + OHE | 76.4 ± 7.5 | 50.2 ± 6.1 | 59.8 ± 5.4 |
| BERTje | 81.5 ± 3.8 | 51.5 ± 4.8 | 62.3 ± 3.7 |
| BERTje + [all] | 79.1 ± 2.9 | 56.2 ± 4.6 | 65.2 ± 3.1 |
| Flair | 71.6 ± 5.2 | 59.9 ± 7.1 | 64.8 ± 6.1 |
| Flair + [all] | 78.7 ± 7.3 | 67.8 ± 6.1 | **72.0 ± 4.8** |
| BERTje + Flair | 79.0 ± 4.0 | 50.9 ± 4.1 | 61.3 ± 2.9 |
| BERTje + Flair + [all] | 84.6 ± 2.2 | 60.2 ± 5.4 | 69.6 ± 3.3 |
| **Multilingual embeddings** | | | |
| [all]: mBPEmb + Char + OHE | 84.7 ± 5.0 | 62.2 ± 7.3 | **70.8 ± 4.8** |
| mBERT | 80.5 ± 5.0 | 57.3 ± 5.4 | 66.5 ± 5.1 |
| mBERT + [all] | 81.6 ± 4.6 | 58.5 ± 5.7 | 67.7 ± 4.7 |
| mFlair | 81.4 ± 5.3 | 51.0 ± 4.1 | 62.4 ± 4.3 |
| mFlair + [all] | 82.2 ± 5.0 | 60.4 ± 7.8 | 68.3 ± 5.7 |
| mBERT + mFlair | 87.0 ± 3.3 | 53.8 ± 9.2 | 65.1 ± 8.0 |
| mBERT + mFlair + [all] | 80.4 ± 5.6 | 60.9 ± 6.5 | 69.0 ± 5.9 |

vantage of using pretrained language models to provide word embeddings. Indeed, fixed (sub)word embeddings such as monolingual BytePair or fastText embeddings performed poorly with F1-scores of roughly 50 % (Fig 4.5). On the other hand, performance did benefit from concatenating these fixed (sub)word embeddings with contextualized word vectors obtained from - especially - the pretrained Dutch Flair model. This was in contrast to the results on the CoNLL2002 task, where Flair-based embeddings were clearly outperformed by embeddings obtained from BERTje (Fig 4.2). Even though some of the embeddings seemed to benefit from further increasing the dimensionality of the vectors with trainable character-feature representations, there was no clear pattern accross the types of contextualized embeddings or across monolingual and multilingual embeddings. This, combined with the observation that character-feature embeddings alone performed very poorly, might indicate that this specific representation was not very informative for the BiLSTM-CRF classifier. The fact that the dataset was not completely monolingual is further reflected in the results for the multilingual embeddings: not only was the performance of the mBERT/mFlair-based embeddings on a par with the monolingual embeddings (Table 4.5), multilingual BytePair embeddings also clearly outperformed their monolingual counterparts (Fig **??**). Performance was, on average, highest for the monolingual Flair embeddings, concatenated with all monolingual static and task-specific representations (F1-score: 72.0 ± 4.8 %). Among all multilingual embeddings, the best performance was, on average, obtained with the concatenation of all multilingual static and task-specific representations (F1-score: 70.8 ± 4.8 %) (Table 4.5). A complete overview of all results is given in Table B.1.

The performance on the French Faktion dataset was, on average, lower in comparison with the Dutch dataset. However, it should be noted that the French dataset consisted of

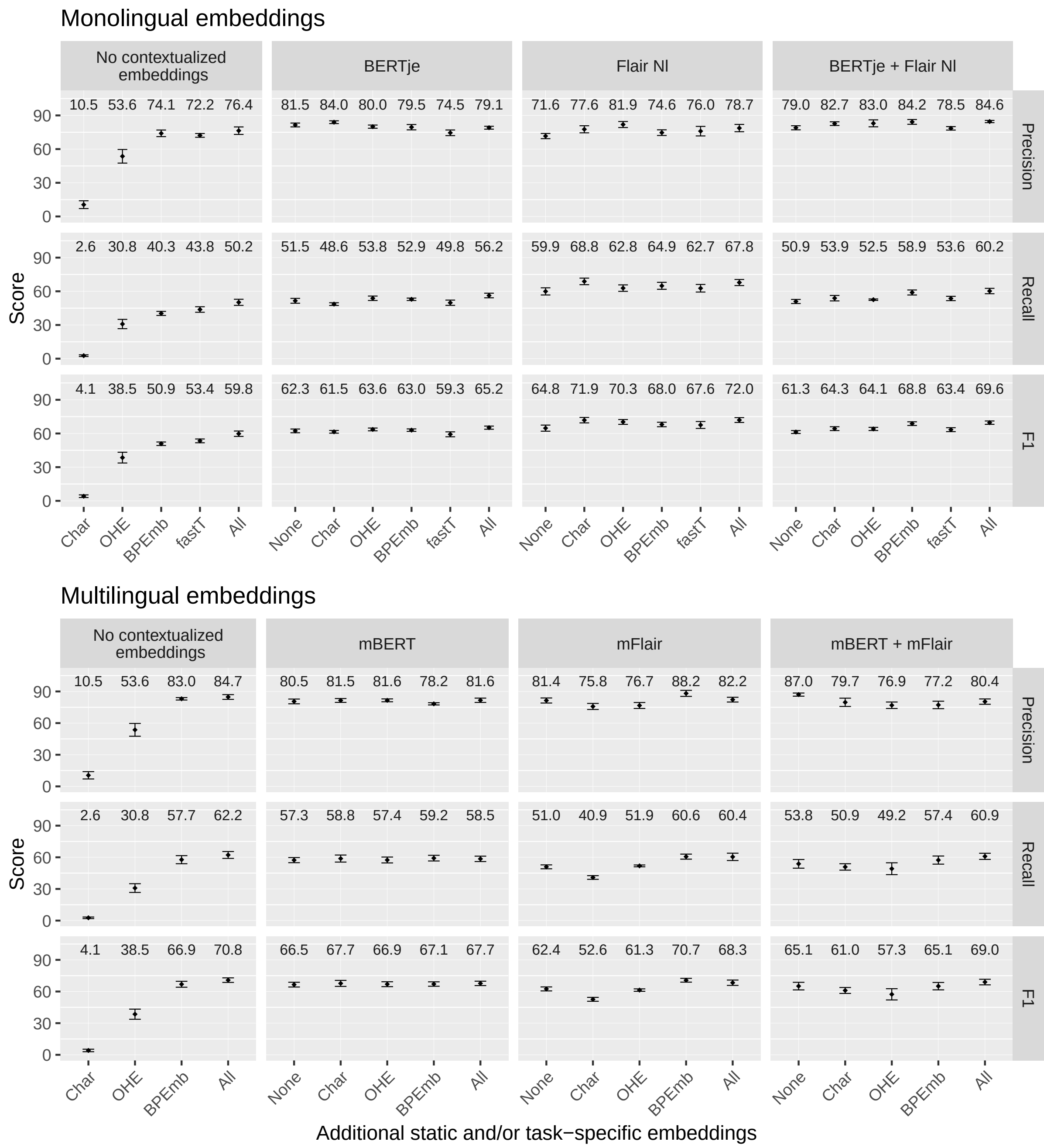


Figure 4.5: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the Dutch Faktion NER task. The results are grouped according to the constituents of each (stacked) embedding (Table 3.4). The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, the results for the language-independent OHE and Char embeddings are indicated on both the upper (monolingual) and lower (multilingual) panels. The symbol and error bars indicate the mean and standard error of the mean, respectively. *Char: character embeddings; OHE: One Hot Embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; All: all embeddings indicated on the x-axis for this panel concatenated into a single vector.*

Table 4.6: Results acquired on the French Faktion NER task with both monolingual and multilingual NER systems. Results are reported as mean ± standard error of the mean (k = 5). A complete overview of all results is provided in Table B.1.

| Embedding type | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual embeddings** | | | |
| [all]: fastT + BPEmb + Char + OHE | 53.7 ± 3.6 | 44.9 ± 4.0 | 48.6 ± 3.5 |
| CamemBERT | 49.9 ± 5.4 | 26.9 ± 3.2 | 34.7 ± 3.8 |
| CamemBERT + [all] | 60.1 ± 6.3 | 50.5 ± 5.0 | **54.6 ± 5.4** |
| Flair | 62.3 ± 6.6 | 32.2 ± 2.8 | 42.3 ± 3.8 |
| Flair + [all] | 56.2 ± 5.2 | 47.7 ± 5.1 | 51.4 ± 5.1 |
| CamemBERT + Flair | 60.2 ± 4.5 | 35.8 ± 4.9 | 44.6 ± 5.1 |
| CamemBERT + Flair + [all] | 59.6 ± 5.4 | 50.9 ± 4.7 | 54.6 ± 4.8 |
| **Multilingual embeddings** | | | |
| [all]: mBPEmb + Char + OHE | 52.1 ± 5.4 | 44.6 ± 3.9 | 47.6 ± 4.1 |
| mBERT | 62.3 ± 7.3 | 39.7 ± 3.4 | 48.3 ± 4.7 |
| mBERT + [all] | 59.9 ± 4.4 | 49.0 ± 3.8 | 53.7 ± 3.8 |
| mFlair | 50.3 ± 5.8 | 28.6 ± 5.2 | 36.2 ± 5.8 |
| mFlair + [all] | 60.6 ± 6.6 | 49.6 ± 5.3 | **54.5 ± 5.8** |
| mBERT + mFlair | 62.4 ± 2.6 | 41.5 ± 2.7 | 49.6 ± 2.2 |
| mBERT + mFlair + [all] | 61.4 ± 3.7 | 47.0 ± 2.7 | 53.0 ± 2.7 |

2 entity categories while the Dutch dataset only contained a single entiy class to predict. The high-dimensional concatenations of CamemBERT together with all static and task-specific embeddings performed best, on average (F1-score: 54.6 ± 4.8 %). In line with the previous observations, the differences between monolingual and multilingual embeddings were negligible (Table 4.6), with the multilingual BytePair embeddings performing surprisingly well (Fig 4.7). Among all multilingual embeddings that were tested for this dataset, estimated performance was, on average, highest for the concatenations of mFlair embeddings with mBytePair, OHE and character embeddings (F1-score: 54.5 ± 5.8 %).

For the benchmark datasets, the performance as expressed in terms of the F1-score typically reflected a good balance between precision and recall. This was, however, not always the case for this dataset. Indeed, while the the French Flair-embeddings performed poorly in terms of the average F1-score (42.3 ± 3.8 %), they did provide the highest precision, on average, among all monolingual systems (62.3 ± 6.6 %). This is a direct result of the fact that the F1-score is defined as the harmonic mean of precision and recall, with the latter being low, on average, for this system (32.2 ± 2.8 %), resulting in a low overall F1-score. For specific applications that require either a high precision or recall, it might make sense in these situations to not only consider their harmonic mean alone, but also evaluate the performance of the system in terms of precision and recall separately. An overview of all results is given in Table B.2.

Lastly, a bilingual dataset was constructed by merging the entire Dutch and French Faktion datasets. Consistently with the results on the multilingual benchmark dataset, we provide the F1-scores for a subset of embedding types in Figure 4.7 and the precision, recall and F1-scores for this subset in Table 4.7. Multilingual stacked embeddings (e.g. mBERT, mBPEmb and character-feature embeddings) outperformed (either French

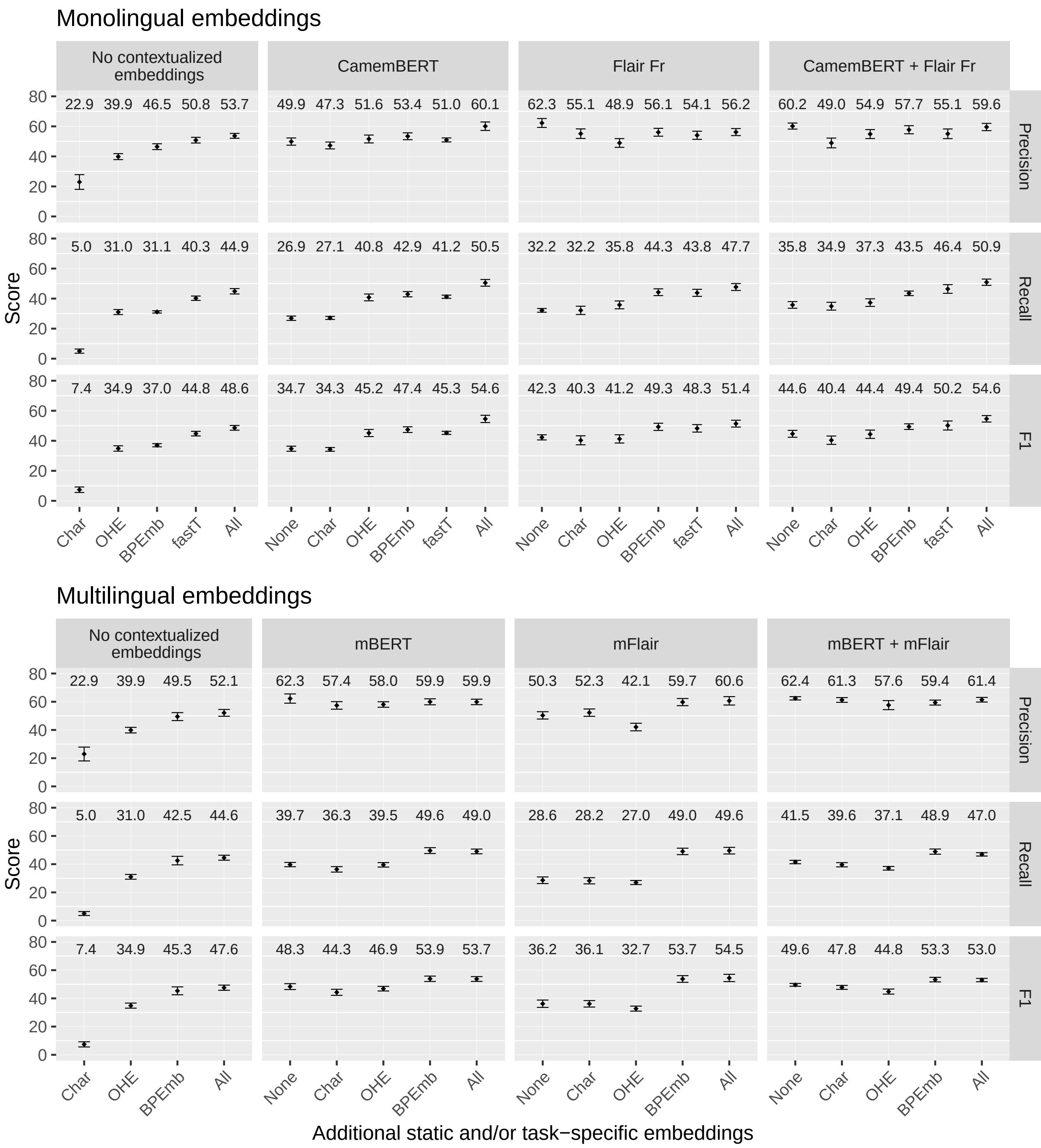


Figure 4.6: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the French Faktion NER task. The results are grouped according to the constituents of each (stacked) embedding (Table 3.4). The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, the results for the language-independent OHE and Char embeddings are indicated on both the upper (monolingual) and lower (multilingual) panels. The symbol and error bars indicate the mean and standard error of the mean, respectively. *Char: character embeddings; OHE: One Hot Embeddings; BPEmb: BytePair embeddings; fastT: fastText embeddings; All: all embeddings indicated on the x-axis for this panel concatenated into a single vector.*

Table 4.7: Results acquired on the multilingual (Dutch + French) Faktion NER task with both monolingual and multilingual NER systems. Results are reported as mean ± standard error of the mean (k = 5). A complete overview of all results is provided in Table B.1.

| Embedding type | Precision | Recall | F1-score |
|---|---|---|---|
| **Monolingual Dutch embeddings** | | | |
| [all-nl]: fastT + BPEmb + Char + OHE | 67.3 ± 3.4 | 51.7 ± 4.3 | 58.3 ± 3.9 |
| BERTje | 67.8 ± 3.6 | 32.5 ± 1.8 | 43.7 ± 1.6 |
| BERTje + [all-nl] | 65.9 ± 2.8 | 41.0 ± 3.3 | 50.2 ± 2.8 |
| Flair | 67.1 ± 2.6 | 43.9 ± 3.6 | 52.7 ± 2.8 |
| Flair + [all-nl] | 69.8 ± 2.1 | 54.5 ± 1.6 | **61.1 ± 1.4** |
| BERTje + Flair | 67.1 ± 2.1 | 28.7 ± 1.8 | 40.1 ± 2.0 |
| BERTje + Flair + [all-nl] | 67.7 ± 2.6 | 39.5 ± 3.3 | 49.7 ± 3.0 |
| **Monolingual French embeddings** | | | |
| [all-fr]: fastT + BPEmb + Char + OHE | 66.0 ± 2.6 | 51.5 ± 3.4 | 57.7 ± 3.0 |
| CamemBERT | 52.8 ± 5.2 | 22.2 ± 1.6 | 30.9 ± 1.8 |
| CamemBERT + [all-fr] | 67.4 ± 3.6 | 53.4 ± 3.0 | **59.5 ± 2.8** |
| Flair | 61.9 ± 4.1 | 28.0 ± 2.9 | 38.0 ± 2.8 |
| Flair + [all-fr] | 68.4 ± 3.9 | 49.9 ± 3.9 | 57.3 ± 3.3 |
| CamemBERT + Flair | 72.0 ± 4.4 | 40.6 ± 3.8 | 51.6 ± 3.7 |
| CamemBERT + Flair + [all-fr] | 68.3 ± 1.8 | 51.8 ± 3.0 | 58.6 ± 1.6 |
| **Multilingual embeddings** | | | |
| [all]: mBPEmb + Char + OHE | 73.7 ± 4.6 | 53.3 ± 3.4 | **61.6 ± 3.5** |
| mBERT | 68.1 ± 2.3 | 45.8 ± 2.6 | 54.6 ± 2.3 |
| mBERT + [all] | 72.3 ± 3.2 | 52.3 ± 2.9 | 60.2 ± 2.0 |
| mFlair | 68.2 ± 4.6 | 35.5 ± 3.1 | 46.3 ± 2.9 |
| mFlair + [all] | 69.0 ± 2.9 | 50.8 ± 3.9 | 58.3 ± 3.3 |
| mBERT + mFlair | 71.0 ± 2.6 | 41.8 ± 1.6 | 52.5 ± 1.5 |
| mBERT + mFlair + [all] | 71.1 ± 3.3 | 51.2 ± 3.2 | 59.4 ± 3.0 |

or Dutch) monolingual embeddings (Table 4.7), with the the concatenation of mBERT, mBytePair and character embeddings performing best, on average (F1-score: 62.8 ± 1.9 %, B.3). These results are in line with the results obtained on the multilingual benchmark dataset, thus confirming our research hypothesis that these multilingual embeddings are more effective than monolingual embeddings when the input data itself is multilingual. A complete overview of all results is provided in Table B.3.

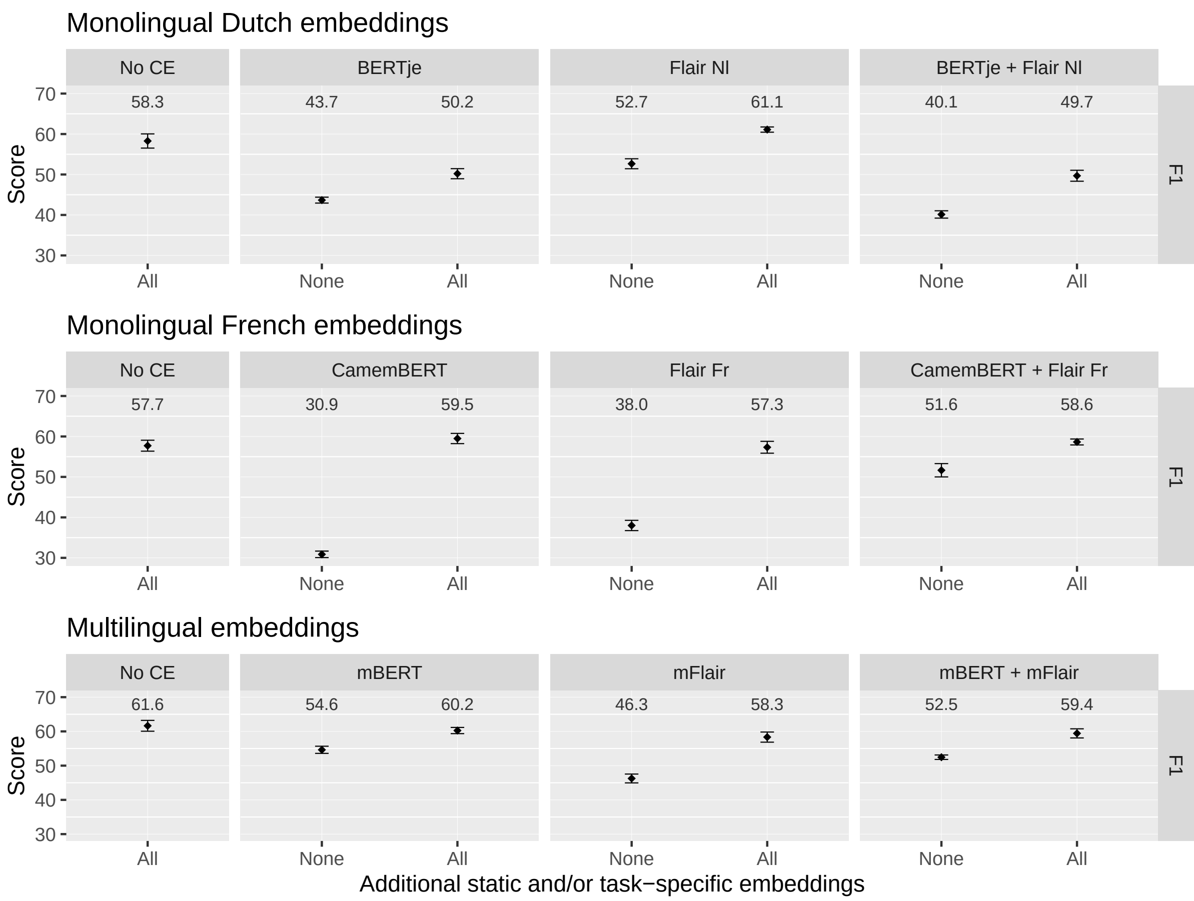


Figure 4.7: Test-set micro-average precision, recall and F1-score for monolingual (top) and multilingual (bottom) embeddings on the multilingual (Dutch + French) Faktion dataset. The results are grouped according to the constituents of each (stacked) embedding (Table 3.4). The contextualized embedding type, if present, is indicated on the different panels from left to right. The stacked/task-specific embedding type (if present) is shown on the x-axis. For clarity, only the results for experiments that did not include any static/task-specific embeddings (None) and for the experiments where all static/task-specific embeddings were included (All), are shown. *No CE: No contextualized embeddings*

# Chapter 5

# Discussion

In this thesis, we aimed to evaluate how different monolingual and multilingual embeddings affected the performance of different monolingual and multilingual NER tasks. We addressed this question by performing a wide range of NER experiments on well-validated English, Dutch and French benchmark datasets, using different combinations of either monolingual or multilingual contextualized, static and/or task-specific word representations. Next, these systems were compared when the input data itself was multilingual. We were able to achieve F1-scores that were in accordance with state-of-the-art results reported in literature and confirmed earlier studies demonstrating that NER performance typically benefits from concatenating different embeddings together into single, high-dimensional representations. Monolingual embeddings outperformed multilingual embeddings on each of the monolingual NER tasks, but the differences were surprisingly small. Conversely, multilingual embeddings clearly outperformed monolingual embeddings on a multilingual NER task.

In addition, we evaluated how our monolingual and multilingual NER implementations performed on "real-world" datasets that were generated from OCR on scans of building plans in the context of a document annotation application developed by Faktion. Not surprisingly, the results on these small, noisy datasets were - on average - much lower as compared to the benchmark datasets. However, concatenating multiple embeddings together into single, high-dimensional representations again showed to be a successful approach to optimize NER performance. The fact that these datasets were not strictly monolingual was reflected in the performance differences between monolingual and multilingual embeddings: they performed similarly on the predominantly monolingual datasets, but multilingual embeddings outperformed monolingual embeddings when the input data was multilingual. These results suggest that state-of-the-art multilingual representations available today are ready to be implemented in practical applications, thus greatly reducing pipeline complexity at no cost of reduced performance.

## 5.1 Benchmarkt datasets

By using a state-of-the-art BiLSTM-CRF sequence labeling architecture (Akbik et al., 2018; Huang et al., 2015; Lample et al., 2016), we were able to achieve results that were similar or slightly below results reported in literature for the CoNLL2003 task. While a direct comparison of our results with literature data was not the primary goal, it does indicate that our implementations provided a very useful framework to compare state-of-the-art monolingual and multilingual embeddings. The single point estimates of the extra-sample performances that were obtained for the different monolingual and multilingual embeddings did not allow us to thoroughly study the score distributions of all the implementations. We specifically opted to evaluate a wide range of different stacked combinations of monolingual and multilingual embeddings, in order to acquire a general overview on:

- Baseline performance with task-specific representations (i.e. without transferring knowledge from static or contextualized representations)
- The added value of static word or subword embeddings
- The added value of state-of-the-art contextualized representations from pretrained Flair or BERT models

For most tasks, the Transformer-based BERT models (BERTje, CamemBERT, BERT and mBERT) outperformed the character-based Flair LMs, except for monolingual English, where the Flair model provided excellent representations that resulted in a similar performance on the NER task as compared to BERT-based representations. Interestingly, the difference between BERT-based and Flair-based embeddings was even more pronounced for multilingual models: mBERT clearly outperformed mFlair on all monolingual as well as multilingual benchmark datasets. In many cases, mFlair-based stacked embeddings were even outperformed by embeddings that did not include any contextualized representation obtained from a pretrained LM. There are several reasons that may account for this. Firstly, mBERT was trained on roughly 100 of the most common languages on Wikipedia, while mFlair was provided with the JW300 training corpus containing just over 2 billion tokens in more than 300 languages (Agić and Vulić, 2019). Given the significant cost involved into training a neural LM from scratch, the size of the training corpus is often limited. While corpus sizes in terms of number of tokens are not always directly comparable between models (e.g. mBERT relied on subword tokenization and resampling strategies to account for the imbalance between different languages in the multilingual corpus), most models are pretrained on a corpora of at most a few billion tokens. This means that - in practice - the amount of training data for each specific language in a multilingual training corpus is inversely proportional to the number of different languages. Among many other differences between the mFlair and mBERT models, this might explain why mBERT seems to provide better representations for very common languages like English, Dutch or French. It might be interesting to evaluate how mBERT and mFlair embeddings perform on low-resource languages that are not among the 100 most commong languages on Wikipedia.

An interesting results was obtained with the Dutch version of BERT's Transformer architecture, BERTje (de Vries et al., 2019). Using the representations from its top layer, we achieved - to the best of our knowledge - a new state-of-the art on the Dutch CoNLL2002 task, outperforming the previous best result obtained with fine-tuning of mBERT (Wu and Dredze, 2019) by 2.1 percentage points (from 90.9 % to 93.0 %). However, one should be cautious directly comparing these results. Firstly, computational resources did not allow us to perform multiple training runs for the different configurations. The stochastic nature of the training algorithm renders the BiLSTM-CRF architecture non-deterministic. Even though the obtained F1 point estimates are statistically unbiased, they are not necesarily representative for the entire score distribution. Indeed, empirical evidence has shown that the seed value chosen for the random number generator for state-of-the-art BiLSTM-CRF systems can result in F1-score differences as large as 1 percentage points on the CoNLL2003 task, which translates into a system being perceived as mediocre or state-of-the-art (Reimers and Gurevych, 2017). In addition, our results on the CoNLL2002 dataset were obtained after (minor) preprocessing: 4/15806 sentences in the training set and 1/5195 sentences from the test set were removed for all experiments because they exceeded the maximum input sequence length for some models included in our evaluation. Nevertheless, all results for the different stackings based on BERTje embeddings (ranging from 91.5 % (for BERTje embeddings alone) to 93.0 % (for the largest stacked embedding)) consistently outperformed the result reported before (de Vries et al., 2019; Wu and Dredze, 2019). The importance of the sequence labeling architecture has been acknowledged already by the authors of the BERTje model (de Vries et al., 2019) and our results seem to confirm that feeding the activations from BERTje's top layer into a BiLSTM-CRF model yields a considerable increase in performance as compared to fine-tuning BERTje or mBERT and using a regular softmax layer on top of the Transformer model to predict the NER labels.

Since we only used the activations for every first subtoken extracted from BERTje's top layer, we hypothesize that this result can be optimized even further. Indeed, earlier studies have indicated that performance of many NLP tasks can significanlty benefit from strategies that aim to specifically extract representations from the different layers of the pretrained Transformer LM. This includes manual per-layer analysis to find which layers encode information most relevant for the specific task or, alternative, automativally learning a scalar mixture of the activations from the different layers to find the representation most relevant for the downstream task. Scalar mixing seems to outperform the best individual layer's representation on a wide range of NLP tasks (Liu et al., 2019) and was shown to be one of the main factors that determined the success of the representations obtained from the deep BiLSTM LM ELMo (Peters et al., 2018).

The conclusions for the French WikiNER dataset were in accordance with the Dutch and English NER: monolingual systems outperformed multilingual ones, with a pronounced difference between Flair-based and Transformer-based representations. Given that this difference was not present in the results of the English CoNLL2003 task suggests that this is not merely a result of the intrinsic properties of the neural network architecture as such, but that - indeed - a character-based LM is able to provide excellent representations for downstream NER, if trained well. Given that the Dutch, French and multilingual Flair models have been pretrained by members of the Flair NLP community, while all other models (English Flair and the monolingual and multilingual versions of BERT) have been pretrained by research institutions might suggest that this difference can be - at least partly - attributed to the availability of resources to train these models. Even though the training of character-based LMs like Flair is order of magnitudes more efficient than traditional word-based BiLSTM LMs (Akbik et al., 2018), it still requires the availability of enormous amounts of data and training for more than one week on a powerful GPU[1].

Overall, our results for the benchmark datasets are in accordance with results reported in literature. Performance of the NER classifier consistently increased from representations learned from scratch directly onto the NER objective (OHE or character embeddings), to pretrained (static) vectors like fastText or BytePair embeddings, to contextualized rep-

[1]GitHub: flairNLP/flair-lms

resentations obtained from large, pretrained neural LMs, as anticipated. Even though static word vectors provide only a single, "average" representation for polysemous words and are, hence, typically outperformed by contextualized representations on NER tasks, interestingly, performance seems to increase when static word vectors are concatenated to these contextualized representations, suggesting that they do add relevant information. This has been reported before, i.e. Akbik and colleagues obtained state-of-the-art performance on the CoNLL2003 task using concatenated Flair, GloVe and character-level representations (Akbik et al., 2018). An interesting study by Kawin Ethayarajh investigated how the different context-specific representations of the same word (obtained from e.g. BERT) compared against static - non-contextualized - embeddings. They found that BERT's representations (especially the representations in the upper layers) were highly contextualized, in the sense that a single static embedding (the first principal component of the different context-dependent representations of a word) explained, on average, less than 5 % of the total variance (Ethayarajh, 2019). This might explain why concatenating the apparently highly context-dependent representation obtained from the last layer of BERT with a static word embedding that encodes more latent word-level semantics helps the NER classifier to perform well on the task.

Even though the large, high-dimensional stackings of different embedding types generally outperformed lower-dimensional representations obtained from a single pretrained model, the difference was typically small. However, the computational requirements to train models using these high-dimensional representations are much higher. In particular, the task-specific representations that are iteratively updated during training on the NER objective are essentially parameters of the BiLSTM-CRF model. As such, expanding the parameter space slows down the training procedure considerably and even though these high-dimensional representations seem to consistently (but slightly) outperform lower-dimensional embeddings, this balance between training efficiency and performance might be considered a limitation for some practical applications.

## 5.2 Faktion datasets

There were major differences between the characteristics of the Faktion datasets and the benchmark datasets discussed previously, which translated - not surprisingly - into different conclusions with respect to:

- The performance of monolingual and multilingual embeddings
- The added value of obtaining contextualized representations from pretrained LMs

Firstly, the monolingal datasets were not strictly monolingual. This was reflected in the performance for monolingual and multilingual embeddings: while the former consistently outperformed the latter on the CoNLL and WikiNER datasets, there was virtually no difference for the Faktion datasets. In both the trilingual benchmark dataset as well as the bilingual Faktion dataset, the multilingual embeddings clearly outperformed any monolingual embedding. Currently, the NER-step in the Faktion document annotation application relies on a two-step procedure: first, the language of the document is predicted, and this determines which (monolingual) NER system is used. Even though the advantage of monolingual embeddings over multilingual embeddings might be more pronounced when the training datasets are strictly monolingual, the results obtained here provide strong indications that a multilingual NER pipeline might provide similar performance while completely omitting the need to maintain a separate language prediction step. In addition to this simplification, the risk of poor NER performance caused by errors in the upstream language prediction step is greatly reduced, as suggested by the presults obtained on the multilingual dataset.

Again - there was a clear trend of increasing NER performance when concatenating multiple embedding types into large, stacked embeddings. It was, however, surprising that the added value of contextualized embeddings from either pretrained Flair or BERT-like architectures was much less pronounced as compared to the benchmark NER tasks. Performance of the multilingual BytePair embeddings was, in general, surprisingly high, not only as compared directly to either French or Dutch monolingual BytePair embeddings, but especially when compared to the contextualized embeddings obtained from pretrained Flair or BERT. Condidering the nature of the Faktion datasets (i.e. relatively noisy data obtained from OCR on scans of building plans) this might not come as a big surprise: many of the tokens contained punctuation or digits, were very short or even consisted of a single character. Since the “linguistic knowledge” acquired by these pretrained neural LMs are based on corpora that are chosen to be as representative as possible for “general language” (i.e. news articles, Wikipedia, books, …), they are expected to work best in this context. The fact that the semantic dependencies between tokens in the documents of the Faktion datasets are not necessarily representative for “general language”, might partly explain why the acquired linguistic knowledge of pretrained BERT and Flair to provide contextualized token representations does not seem to help the NER classifier to perform well on this task. This has also been described in situations of e.g. jargon-rich language like biomedical text. BioBERT - a BERT architecture pretrained on biomedical data - has been shown to outperform regular BERT for many NLP tasks specific to the field, including biomedical NER (Lee et al., 2020). Unfortunately, training such a model entirely from scratch requires enormous amounts of data and comes at a significant cost. Since the weights of pretrained BERT have already very sensible values for many problems, a more efficient approach is to finetune the parameters of pretrained models using task-specific data. Indeed, this approach has shown to be very succesful for different scientific domains (Beltagy et al., 2019) or to enrich the language-specific vocabulary of multilingual models (Wang et al., 2019).

As mentioned before, the Faktion datasets consisted of only one (for the Dutch dataset) or two (for the French and bilingual dataset) entity categories. For the latter, the vast majority of all entities (approximately 95 %) belonged to a single category. In order to be consistent with the gold-standard evaluation procedure for NER tasks reported in literature, we reported micro-average entity-level precision, recall and F1-scores. Even though the F1-score is typically used to compare different NER systems and rank the performance on benchmark tasks like CoNLL, it should be mentioned that for some cases - including the highly imbalanced Faktion datasets - this does not provide a good representation of the per-class performance of the model. Since it is a weighted average of the F1-score per entity class, poor performance on the minority class is not well-reflected in the overall micro-average score. When optimizing NER classifiers for practical applications and depending on the relative cost of wrongly classifying entities belonging to the minority class, this should be taken into account. Also, as mentioned before, NER evaluation is typically performed on the entity-level, not on the token level. This means that optimizing an NER system for this metric essentially penalizes the model to be "partially correct". Indeed, a system that predicts "In Prato [B-LOC] allo [I-LOC] Stelvio [I-LOC]" as "In Prato [B-LOC] allo Stelvio [B-LOC]" makes 2 false positive predictions (Prato [B-LOC] and Stelvio [B-LOC]) and one false negative prediction (Prato [B-LOC] allo [I-LOC] Stelvio [I-LOC]). As a result, the system is penalized both in terms of precision (+ 2 FPs) and recall (+ 1 FN). In contrast, when the NER system predicts no entities at all, only a single false negative prediction is counted because the system failed to correctly predict the entity "Prato [B-LOC] allo [I-LOC] Stelvio [I-LOC]", thus only penalizing recall (+ 1 FN). This means that by optimizing a NER system on the entity-level F1-score, the system is essentially discouraged to predict these partial matches. One can argue that for many practical applications, the behavior of the first system is preferred, and optimizing for the F1-score might not be the best strategy in this case. This issue has been raised previously[2] and alternative evaluation strategies have been proposed to allow these partial matches to be taken into account (Chinchor and Sundheim, 1993; Doddington et al., 2004; Esuli and Sebastiani, 2010). However, these systems are typically quite complex and the vast majority of researchers adheres to the convention of evaluating NER systems according to micro-average F1-score. It might be interesting, however, to explore how other evaluation metrics compare to the way humans would naturally judge the performance of a NER system and how a lower F1-score might actually be preferable for practical applications like Metamaze.

## 5.3 Conclusion

To conclude, we discuss some limitations of the work presented here as well as some future perspectives. The main goal of this thesis was to acquire an overview on how different monolingual and multilingual embeddings affect NER performance in different situations. With this goal in mind, we aimed to keep the model parameters and embedding types consistent between experiments. While this provided a good overview on the relative contribution of the different embedding types to NER performance in many different situations, we did not specifically aim to maximize performance of each individual system or provide a robust analysis on the score distributions of each system. However, given the stochastic nature of the training algorithms, one should consider the entire score distribution in order to formally infer whether different monolingual and multilingual systems perform - on average - differently. It might be interesting to perform such a study in the future. In addition, when it comes to implementing a system for a practical application, it is very likely that performance will benefit from optimization. In particular, we did not explore which layers of the Transformer models provided the most informative representations for the NER task but focused predominantly on the comparison between monolingual and multilingual systems based on the activations from the top layer only. However, using different combinations of layers or weighted averages of the activations from different layers has been shown to improve performance in many cases (Devlin et al., 2019; Liu et al., 2019). Hence, it might be very interesting to explore to what extend these strategies can further improve both monolingual as well as multilingual NER systems based on BERT-embeddings. These optimization steps might be equally effective in improving the results on the Faktion datasets, even though - in general - the advantage of these state-of-the-art

[2] "Doing Named Entity Recognition? Don't optimize for F1" - Chris Manning (2006)

contextualized embeddings from pretrained neural LMs was more limited. Considering the nature of the Faktion datasets and the limited overlap between the type of text data that was used to train these LMs and the Faktion datasets, it might be very interesting to explore how fine-tuning the weights of mBERT with data more relevant to this specific problem might result in representations that are able to improve the performance of the NER system in this specific context even further. Nevertheless, even without extensive optimization, our results indicate that the multilingual embeddings that were evaluated during this thesis have the ability to significantly reduce the complexity of the current document annotation pipeline of Metamaze, by omitting the need for a separate language-prediction step, while achieving excellent performance compared to monolingual systems, in particular when the language heterogeneity of the dataset grows.

# Appendix A

# Benchmark results

In this appendix, we provide an overview of the entity-level test-set precision, recall and F1-scores for all NER experiments on the English CoNLL2003 dataset (Table A.1), the Dutch CoNLL2002 dataset (Table A.2), the French WikiNER dataset (Table A.3) and the multilingual benchmark dataset (Table A.4), as described in chapter 3. Large stackings of static and task-specific embeddings are referred to as 'All', indicating - for monolingual embeddings - the concatenation of character (Char) embeddings, one hot word type embeddings (OHE), BytePair embeddings (BPEmb) and fastText (fastT) embeddings for the respective languages. For multilingual embeddings, "All" refers to the concatenation of multilingual BytePair embeddings (mBPEmb), together with OHE and character embeddings. Details are given in the text.

Table A.1: Overview of all results for the English CoNLL2003 task.
The reported scores are the result of a single training run.

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 74.1 | 73.9 | 74.0 |
| None | OHE | 80.7 | 70.8 | 75.4 |
| **Monolingual embeddings** | | | | |
| None | BPEmb | 86.7 | 84.9 | 85.7 |
| None | fastT | 90.4 | 89.9 | 90.1 |
| None | All | 90.8 | 90.9 | 90.9 |
| BERT | None | 91.3 | 91.1 | 91.2 |
| BERT | Char | 91.1 | 91.2 | 91.2 |
| BERT | OHE | 91.1 | 90.5 | 90.8 |
| BERT | BPEmb | 91.1 | 91.5 | 91.3 |
| BERT | fastT | 91.7 | 92.0 | 91.8 |
| BERT | All | 91.9 | 92.3 | 92.1 |
| Flair | None | 90.3 | 90.8 | 90.5 |
| Flair | Char | 90.6 | 91.1 | 90.8 |
| Flair | OHE | 90.1 | 90.8 | 90.4 |
| Flair | BPEmb | 91.7 | 92.2 | 91.9 |
| Flair | fastT | 92.0 | 92.6 | 92.3 |
| Flair | All | 92.3 | 92.4 | 92.3 |
| BERT + Flair | None | 91.3 | 91.9 | 91.6 |
| BERT + Flair | Char | 91.3 | 92.1 | 91.7 |
| BERT + Flair | OHE | 91.6 | 92.0 | 91.8 |
| BERT + Flair | BPEmb | 91.8 | 92.2 | 92.0 |

Table A.1: Overview of all results for the English CoNLL2003 task. The reported scores are the result of a single training run. *(continued)*

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| BERT + Flair | fastT | 91.7 | 92.3 | 92.0 |
| BERT + Flair | All | 92.3 | 92.7 | 92.5 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 86.0 | 83.2 | 84.5 |
| None | All | 88.3 | 87.8 | 88.0 |
| mBERT | None | 90.8 | 90.6 | 90.7 |
| mBERT | Char | 90.6 | 91.4 | 91.0 |
| mBERT | OHE | 90.5 | 90.9 | 90.6 |
| mBERT | BPEmb | 91.0 | 91.7 | 91.3 |
| mBERT | All | 91.0 | 91.9 | 91.4 |
| mFlair | None | 84.6 | 84.8 | 84.7 |
| mFlair | Char | 84.6 | 85.0 | 84.8 |
| mFlair | OHE | 85.8 | 85.9 | 85.8 |
| mFlair | BPEmb | 87.6 | 88.4 | 88.0 |
| mFlair | All | 87.6 | 88.2 | 87.9 |
| mBERT + mFlair | None | 90.5 | 91.2 | 90.9 |
| mBERT + mFlair | Char | 90.5 | 91.2 | 90.9 |
| mBERT + mFlair | OHE | 90.8 | 91.4 | 91.1 |
| mBERT + mFlair | BPEmb | 91.0 | 92.0 | 91.5 |
| mBERT + mFlair | All | 90.8 | 91.4 | 91.1 |

Table A.2: Overview of all results for the Dutch CoNLL2002 task. The reported scores are the result of a single training run.

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 68.9 | 67.2 | 67.8 |
| None | OHE | 80.3 | 50.9 | 61.7 |
| **Monolingual embeddings** | | | | |
| None | BPEmb | 83.1 | 74.6 | 78.0 |
| None | fastT | 86.7 | 75.1 | 79.7 |
| None | All | 88.1 | 86.8 | 87.4 |
| BERTje | None | 91.8 | 91.3 | 91.5 |
| BERTje | Char | 92.6 | 91.9 | 92.2 |
| BERTje | OHE | 92.2 | 91.4 | 91.8 |
| BERTje | BPEmb | 92.5 | 91.6 | 92.0 |
| BERTje | fastT | 92.7 | 91.8 | 92.2 |
| BERTje | All | 93.1 | 92.2 | 92.6 |
| Flair Nl | None | 86.9 | 86.2 | 86.4 |
| Flair Nl | Char | 86.8 | 86.3 | 86.4 |
| Flair Nl | OHE | 87.4 | 86.2 | 86.8 |
| Flair Nl | BPEmb | 88.7 | 88.3 | 88.4 |
| Flair Nl | fastT | 89.8 | 89.2 | 89.4 |
| Flair Nl | All | 90.2 | 89.3 | 89.7 |
| BERTje + Flair Nl | None | 92.4 | 92.5 | 92.4 |
| BERTje + Flair Nl | Char | 92.8 | 92.8 | 92.8 |
| BERTje + Flair Nl | OHE | 92.6 | 92.3 | 92.4 |
| BERTje + Flair Nl | BPEmb | 92.9 | 92.7 | 92.8 |
| BERTje + Flair Nl | fastT | 93.1 | 92.8 | 92.9 |
| BERTje + Flair Nl | All | 93.1 | 93.0 | 93.0 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 85.6 | 67.5 | 75.1 |
| None | All | 82.8 | 80.0 | 81.3 |
| mBERT | None | 89.2 | 88.5 | 88.8 |
| mBERT | Char | 89.6 | 88.5 | 89.0 |
| mBERT | OHE | 89.9 | 89.1 | 89.5 |
| mBERT | BPEmb | 89.9 | 89.3 | 89.5 |
| mBERT | All | 90.6 | 89.4 | 90.0 |
| mFlair | None | 79.8 | 78.8 | 79.1 |
| mFlair | Char | 79.4 | 78.7 | 78.9 |
| mFlair | OHE | 80.7 | 78.7 | 79.6 |
| mFlair | BPEmb | 83.8 | 82.8 | 83.2 |
| mFlair | All | 84.7 | 82.6 | 83.6 |
| mBERT + mFlair | None | 89.8 | 89.4 | 89.5 |
| mBERT + mFlair | Char | 89.9 | 89.3 | 89.5 |
| mBERT + mFlair | OHE | 90.3 | 89.1 | 89.6 |
| mBERT + mFlair | BPEmb | 90.5 | 89.9 | 90.1 |
| mBERT + mFlair | All | 90.4 | 89.8 | 90.1 |

Table A.3: Overview of all results for the French WikiNER task. The reported scores are the result of a single training run.

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 61.0 | 60.5 | 60.2 |
| None | OHE | 70.5 | 65.2 | 67.4 |
| **Monolingual embeddings** | | | | |
| None | BPEmb | 77.3 | 75.5 | 76.1 |
| None | fastT | 79.3 | 74.7 | 76.6 |
| None | All | 84.0 | 83.9 | 83.9 |
| CamemBERT | None | 84.5 | 84.8 | 84.6 |
| CamemBERT | Char | 86.0 | 86.0 | 86.0 |
| CamemBERT | OHE | 86.4 | 86.4 | 86.3 |
| CamemBERT | BPEmb | 85.4 | 85.5 | 85.4 |
| CamemBERT | fastT | 86.7 | 86.6 | 86.6 |
| CamemBERT | All | 86.3 | 86.4 | 86.3 |
| Flair Fr | None | 79.1 | 80.0 | 79.4 |
| Flair Fr | Char | 80.8 | 81.6 | 81.0 |
| Flair Fr | OHE | 82.3 | 82.7 | 82.5 |
| Flair Fr | BPEmb | 83.2 | 83.5 | 83.2 |
| Flair Fr | fastT | 84.6 | 84.8 | 84.6 |
| Flair Fr | All | 85.2 | 85.3 | 85.2 |
| CamemBERT + Flair Fr | None | 86.2 | 86.2 | 86.1 |
| CamemBERT + Flair Fr | Char | 86.7 | 86.7 | 86.7 |
| CamemBERT + Flair Fr | OHE | 86.2 | 86.0 | 86.1 |
| CamemBERT + Flair Fr | BPEmb | 87.1 | 86.8 | 86.9 |
| CamemBERT + Flair Fr | fastT | 87.6 | 87.5 | 87.6 |
| CamemBERT + Flair Fr | All | 87.4 | 87.4 | 87.4 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 77.6 | 74.1 | 75.6 |
| None | All | 80.2 | 80.8 | 80.4 |
| mBERT | None | 85.5 | 85.4 | 85.4 |
| mBERT | Char | 84.8 | 84.8 | 84.8 |
| mBERT | OHE | 85.3 | 85.3 | 85.2 |
| mBERT | BPEmb | 86.4 | 86.6 | 86.5 |
| mBERT | All | 85.3 | 85.0 | 85.1 |
| mFlair | None | 74.4 | 75.7 | 74.7 |
| mFlair | Char | 76.8 | 77.8 | 77.1 |
| mFlair | OHE | 79.5 | 80.1 | 79.8 |
| mFlair | BPEmb | 81.2 | 81.9 | 81.5 |
| mFlair | All | 81.0 | 81.6 | 81.2 |
| mBERT + mFlair | None | 84.8 | 84.9 | 84.8 |
| mBERT + mFlair | Char | 85.3 | 85.5 | 85.4 |
| mBERT + mFlair | OHE | 85.0 | 85.0 | 85.0 |
| mBERT + mFlair | BPEmb | 85.6 | 85.7 | 85.6 |
| mBERT + mFlair | All | 86.2 | 86.3 | 86.2 |

Table A.4: Overview of all results for the multilingual (En + Nl + Fr) NER task. The reported scores are the result of a single training run.

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 61.6 | 60.0 | 60.4 |
| None | OHE | 68.3 | 55.2 | 60.8 |
| **English embeddings** | | | | |
| None | BPEmb | 79.7 | 65.8 | 71.5 |
| None | fastT | 80.8 | 73.2 | 76.3 |
| None | All | 84.4 | 83.8 | 84.0 |
| BERT | None | 82.0 | 81.0 | 81.4 |
| BERT | Char | 82.8 | 82.1 | 82.4 |
| BERT | OHE | 82.2 | 81.0 | 81.5 |
| BERT | BPEmb | 83.5 | 82.5 | 83.0 |
| BERT | fastT | 84.3 | 83.3 | 83.8 |
| BERT | All | 85.0 | 84.2 | 84.6 |
| Flair | None | 79.5 | 79.2 | 79.2 |
| Flair | Char | 80.2 | 80.1 | 80.1 |
| Flair | OHE | 80.8 | 80.8 | 80.7 |
| Flair | BPEmb | 83.3 | 83.0 | 83.1 |
| Flair | All | 85.2 | 85.1 | 85.1 |
| BERT + Flair | None | 83.5 | 83.4 | 83.4 |
| BERT + Flair | Char | 84.2 | 83.6 | 83.8 |
| BERT + Flair | OHE | 83.7 | 82.8 | 83.2 |
| BERT + Flair | BPEmb | 84.2 | 83.7 | 83.9 |
| BERT + Flair | All | 85.3 | 85.1 | 85.2 |
| **Dutch embeddings** | | | | |
| None | BPEmb | 78.7 | 67.2 | 71.9 |
| None | fastT | 83.1 | 72.8 | 77.1 |
| None | All | 83.5 | 82.9 | 83.1 |
| BERTje | None | 80.1 | 77.4 | 78.6 |
| BERTje | Char | 80.2 | 79.2 | 79.7 |
| BERTje | OHE | 80.6 | 77.5 | 79.0 |
| BERTje | BPEmb | 82.9 | 81.7 | 82.3 |
| BERTje | fastT | 84.3 | 83.5 | 83.9 |
| BERTje | All | 85.0 | 84.8 | 84.9 |
| Flair Nl | None | 79.1 | 79.7 | 79.3 |
| Flair Nl | Char | 79.4 | 79.9 | 79.5 |
| Flair Nl | OHE | 80.3 | 80.7 | 80.4 |
| Flair Nl | BPEmb | 81.6 | 82.3 | 81.9 |
| Flair Nl | fastT | 84.0 | 84.2 | 84.0 |
| Flair Nl | All | 84.1 | 84.2 | 84.1 |
| BERTje + Flair Nl | None | 80.7 | 80.6 | 80.6 |
| BERTje + Flair Nl | Char | 80.5 | 80.6 | 80.5 |
| BERTje + Flair Nl | OHE | 81.5 | 81.1 | 81.2 |
| BERTje + Flair Nl | BPEmb | 83.0 | 83.4 | 83.1 |
| BERTje + Flair Nl | fastT | 83.7 | 83.9 | 83.7 |
| BERTje + Flair Nl | All | 85.2 | 85.8 | 85.5 |
| **French embeddings** | | | | |
| None | BPEmb | 77.9 | 64.6 | 69.9 |
| None | fastT | 82.5 | 71.8 | 76.2 |
| None | All | 83.8 | 83.0 | 83.3 |
| CamemBERT | None | 81.0 | 79.6 | 80.2 |
| CamemBERT | Char | 80.8 | 80.2 | 80.5 |

Table A.4: Overview of all results for the multilingual (En + Nl + Fr) NER task. The reported scores are the result of a single training run. *(continued)*

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| CamemBERT | OHE | 82.1 | 80.5 | 81.2 |
| CamemBERT | BPEmb | 83.1 | 82.4 | 82.7 |
| CamemBERT | fastT | 85.5 | 85.1 | 85.3 |
| CamemBERT | All | 86.1 | 85.6 | 85.8 |
| Flair Fr | None | 76.5 | 75.8 | 76.1 |
| Flair Fr | Char | 77.5 | 77.4 | 77.3 |
| Flair Fr | OHE | 78.8 | 77.5 | 78.1 |
| Flair Fr | BPEmb | 81.9 | 81.2 | 81.5 |
| Flair Fr | fastT | 85.0 | 84.5 | 84.7 |
| Flair Fr | All | 84.4 | 83.3 | 83.8 |
| CamemBERT + Flair Fr | None | 82.0 | 81.2 | 81.6 |
| CamemBERT + Flair Fr | Char | 82.3 | 81.9 | 82.0 |
| CamemBERT + Flair Fr | OHE | 82.9 | 82.1 | 82.5 |
| CamemBERT + Flair Fr | BPEmb | 84.1 | 83.6 | 83.8 |
| CamemBERT + Flair Fr | fastT | 85.2 | 85.8 | 85.4 |
| CamemBERT + Flair Fr | All | 86.0 | 85.5 | 85.7 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 79.4 | 70.7 | 74.5 |
| None | All | 80.2 | 79.1 | 79.6 |
| mBERT | None | 86.6 | 86.4 | 86.5 |
| mBERT | Char | 87.1 | 86.9 | 87.0 |
| mBERT | OHE | 87.3 | 86.8 | 87.1 |
| mBERT | BPEmb | 87.2 | 86.7 | 86.9 |
| mBERT | All | 87.6 | 87.4 | 87.5 |
| mFlair | None | 76.8 | 77.0 | 76.7 |
| mFlair | Char | 77.3 | 77.7 | 77.3 |
| mFlair | OHE | 78.0 | 78.1 | 78.0 |
| mFlair | BPEmb | 80.7 | 81.4 | 81.0 |
| mFlair | All | 81.2 | 81.5 | 81.3 |
| mBERT + mFlair | None | 86.6 | 86.9 | 86.7 |
| mBERT + mFlair | Char | 86.2 | 85.9 | 86.0 |
| mBERT + mFlair | OHE | 86.4 | 86.6 | 86.5 |
| mBERT + mFlair | BPEmb | 87.5 | 87.9 | 87.6 |
| mBERT + mFlair | All | 86.9 | 86.9 | 86.9 |

# Appendix B

# Faktion results

In this appendix, we provide an overview of the entity-level test-set precision, recall and F1-scores for all NER experiments on the Dutch (Table B.1), French (Table B.2) and multilingual (Table B.3) Faktion datasets, as described in chapter 3. Large stackings of static and task-specific embeddings are referred to as ‘All’, indicating - for monolingual embeddings - the concatenation of character (Char) embeddings, one hot word type embeddings (OHE), BytePair embeddings (BPEmb) and fastText (fastT) embeddings for the respective languages. For multilingual embeddings, “All” refers to the concatenation of multilingual BytePair embeddings (mBPEmb), together with OHE and character embeddings. Details are given in the text.

Table B.1: Overview of all results obtained on the Dutch Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5).

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 10.5 ± 7.8 | 2.6 ± 1.7 | 4.1 ± 2.7 |
| None | OHE | 53.6 ± 13.6 | 30.8 ± 9.3 | 38.5 ± 10.7 |
| **Monolingual embeddings** | | | | |
| None | BPEmb | 74.1 ± 6.6 | 40.3 ± 4.1 | 50.9 ± 3.5 |
| None | fastT | 72.2 ± 3.8 | 43.8 ± 5.5 | 53.4 ± 3.8 |
| None | fastT + BPEmb | 77.2 ± 8.1 | 53.8 ± 6.4 | 62.2 ± 5.3 |
| None | fastT + BPEmb + Char | 78.0 ± 5.7 | 57.0 ± 6.4 | 65.0 ± 4.7 |
| None | All | 76.4 ± 7.5 | 50.2 ± 6.1 | 59.8 ± 5.4 |
| BERTje | None | 81.5 ± 3.8 | 51.5 ± 4.8 | 62.3 ± 3.7 |
| BERTje | Char | 84.0 ± 2.8 | 48.6 ± 2.7 | 61.5 ± 2.7 |
| BERTje | OHE | 80.0 ± 3.2 | 53.8 ± 4.4 | 63.6 ± 2.8 |
| BERTje | BPEmb | 79.5 ± 5.3 | 52.9 ± 2.4 | 63.0 ± 2.5 |
| BERTje | fastT | 74.5 ± 5.6 | 49.8 ± 5.2 | 59.3 ± 4.9 |
| BERTje | fastT + BPEmb | 87.5 ± 4.5 | 58.3 ± 3.4 | 69.4 ± 2.5 |
| BERTje | fastT + BPEmb + Char | 85.9 ± 5.3 | 63.2 ± 3.1 | 72.0 ± 1.4 |
| BERTje | All | 79.1 ± 2.9 | 56.2 ± 4.6 | 65.2 ± 3.1 |
| Flair Nl | None | 71.6 ± 5.2 | 59.9 ± 7.1 | 64.8 ± 6.1 |
| Flair Nl | Char | 77.6 ± 7.0 | 68.8 ± 6.5 | 71.9 ± 5.4 |
| Flair Nl | OHE | 81.9 ± 6.0 | 62.8 ± 6.4 | 70.3 ± 4.8 |
| Flair Nl | BPEmb | 74.6 ± 5.7 | 64.9 ± 6.9 | 68.0 ± 4.5 |
| Flair Nl | fastT | 76.0 ± 9.5 | 62.7 ± 7.5 | 67.6 ± 6.9 |
| Flair Nl | fastT + BPEmb | 81.5 ± 4.7 | 69.3 ± 5.4 | 74.0 ± 2.5 |

Table B.1: Overview of all results obtained on the Dutch Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5). *(continued)*

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| Flair Nl | fastT + BPEmb + Char | 82.3 ± 7.5 | 66.5 ± 7.4 | 72.6 ± 6.0 |
| Flair Nl | All | 78.7 ± 7.3 | 67.8 ± 6.1 | 72.0 ± 4.8 |
| BERTje + Flair Nl | None | 79.0 ± 4.0 | 50.9 ± 4.1 | 61.3 ± 2.9 |
| BERTje + Flair Nl | Char | 82.7 ± 3.6 | 53.9 ± 5.4 | 64.3 ± 3.8 |
| BERTje + Flair Nl | OHE | 83.0 ± 6.9 | 52.5 ± 1.4 | 64.1 ± 3.2 |
| BERTje + Flair Nl | BPEmb | 84.2 ± 4.7 | 58.9 ± 5.0 | 68.8 ± 3.7 |
| BERTje + Flair Nl | fastT | 78.5 ± 3.5 | 53.6 ± 4.3 | 63.4 ± 3.9 |
| BERTje + Flair Nl | fastT + BPEmb | 83.0 ± 4.5 | 59.6 ± 5.6 | 68.6 ± 3.8 |
| BERTje + Flair Nl | fastT + BPEmb + Char | 77.2 ± 5.2 | 53.7 ± 4.0 | 62.7 ± 3.4 |
| BERTje + Flair Nl | All | 84.6 ± 2.2 | 60.2 ± 5.4 | 69.6 ± 3.3 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 83.0 ± 2.6 | 57.7 ± 8.6 | 66.9 ± 6.3 |
| None | BPEmb + Char | 79.0 ± 8.8 | 55.5 ± 6.7 | 64.6 ± 6.8 |
| None | All | 84.7 ± 5.0 | 62.2 ± 7.3 | 70.8 ± 4.8 |
| mBERT | None | 80.5 ± 5.0 | 57.3 ± 5.4 | 66.5 ± 5.1 |
| mBERT | Char | 81.5 ± 4.0 | 58.8 ± 7.5 | 67.7 ± 6.4 |
| mBERT | OHE | 81.6 ± 3.0 | 57.4 ± 6.2 | 66.9 ± 5.2 |
| mBERT | BPEmb | 78.2 ± 2.3 | 59.2 ± 6.0 | 67.1 ± 4.7 |
| mBERT | BPEmb + Char | 76.2 ± 4.8 | 58.3 ± 6.3 | 65.7 ± 5.4 |
| mBERT | All | 81.6 ± 4.6 | 58.5 ± 5.7 | 67.7 ± 4.7 |
| mFlair | None | 81.4 ± 5.3 | 51.0 ± 4.1 | 62.4 ± 4.3 |
| mFlair | Char | 75.8 ± 6.4 | 40.9 ± 3.9 | 52.6 ± 4.1 |
| mFlair | OHE | 76.7 ± 6.3 | 51.9 ± 1.9 | 61.3 ± 2.6 |
| mFlair | BPEmb | 88.2 ± 6.4 | 60.6 ± 5.3 | 70.7 ± 4.0 |
| mFlair | BPEmb + Char | 77.5 ± 8.1 | 52.3 ± 3.9 | 61.9 ± 4.6 |
| mFlair | All | 82.2 ± 5.0 | 60.4 ± 7.8 | 68.3 ± 5.7 |
| mBERT + mFlair | None | 87.0 ± 3.3 | 53.8 ± 9.2 | 65.1 ± 8.0 |
| mBERT + mFlair | Char | 79.7 ± 8.8 | 50.9 ± 6.8 | 61.0 ± 6.3 |
| mBERT + mFlair | OHE | 76.9 ± 6.8 | 49.2 ± 12.5 | 57.3 ± 11.9 |
| mBERT + mFlair | BPEmb | 77.2 ± 7.9 | 57.4 ± 8.6 | 65.1 ± 7.8 |
| mBERT + mFlair | BPEmb + Char | 87.8 ± 3.9 | 62.6 ± 4.8 | 72.6 ± 3.6 |
| mBERT + mFlair | All | 80.4 ± 5.6 | 60.9 ± 6.5 | 69.0 ± 5.9 |

Table B.2: Overview of all results obtained on the French Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5).

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 22.9 ± 11.0 | 5.0 ± 3.1 | 7.4 ± 4.1 |
| None | OHE | 39.9 ± 4.4 | 31.0 ± 3.8 | 34.9 ± 4.0 |
| **Monolingual embeddings** | | | | |
| None | BPEmb | 46.5 ± 4.3 | 31.1 ± 1.7 | 37.0 ± 2.4 |
| None | fastT | 50.8 ± 4.2 | 40.3 ± 3.2 | 44.8 ± 3.4 |
| None | fastT + BPEmb | 55.5 ± 4.7 | 49.3 ± 4.2 | 51.9 ± 4.1 |
| None | fastT + BPEmb + Char | 55.7 ± 4.7 | 46.7 ± 5.1 | 50.5 ± 5.0 |
| None | All | 53.7 ± 3.6 | 44.9 ± 4.0 | 48.6 ± 3.5 |
| CamemBERT | None | 49.9 ± 5.4 | 26.9 ± 3.2 | 34.7 ± 3.8 |
| CamemBERT | Char | 47.3 ± 5.1 | 27.1 ± 2.3 | 34.3 ± 2.8 |
| CamemBERT | OHE | 51.6 ± 5.9 | 40.8 ± 5.1 | 45.2 ± 5.3 |
| CamemBERT | BPEmb | 53.4 ± 5.1 | 42.9 ± 3.9 | 47.4 ± 4.3 |
| CamemBERT | fastT | 51.0 ± 3.0 | 41.2 ± 2.4 | 45.3 ± 2.3 |
| CamemBERT | fastT + BPEmb | 54.7 ± 6.1 | 47.8 ± 4.0 | 50.6 ± 4.8 |
| CamemBERT | fastT + BPEmb + Char | 59.6 ± 6.5 | 50.7 ± 5.1 | 54.5 ± 5.6 |
| CamemBERT | All | 60.1 ± 6.3 | 50.5 ± 5.0 | 54.6 ± 5.4 |
| Flair Fr | None | 62.3 ± 6.6 | 32.2 ± 2.8 | 42.3 ± 3.8 |
| Flair Fr | Char | 55.1 ± 7.1 | 32.2 ± 6.2 | 40.3 ± 6.7 |
| Flair Fr | OHE | 48.9 ± 6.5 | 35.8 ± 5.8 | 41.2 ± 6.1 |
| Flair Fr | BPEmb | 56.1 ± 5.9 | 44.3 ± 5.1 | 49.3 ± 5.3 |
| Flair Fr | fastT | 54.1 ± 6.1 | 43.8 ± 5.2 | 48.3 ± 5.6 |
| Flair Fr | fastT + BPEmb | 53.7 ± 4.3 | 46.9 ± 5.6 | 49.8 ± 4.9 |
| Flair Fr | fastT + BPEmb + Char | 58.4 ± 4.3 | 51.2 ± 2.9 | 54.4 ± 3.3 |
| Flair Fr | All | 56.2 ± 5.2 | 47.7 ± 5.1 | 51.4 ± 5.1 |
| CamemBERT + Flair Fr | None | 60.2 ± 4.5 | 35.8 ± 4.9 | 44.6 ± 5.1 |
| CamemBERT + Flair Fr | Char | 49.0 ± 7.3 | 34.9 ± 5.8 | 40.4 ± 6.2 |
| CamemBERT + Flair Fr | OHE | 54.9 ± 6.7 | 37.3 ± 5.7 | 44.4 ± 6.2 |
| CamemBERT + Flair Fr | BPEmb | 57.7 ± 6.0 | 43.5 ± 3.4 | 49.4 ± 4.2 |
| CamemBERT + Flair Fr | fastT | 55.1 ± 7.2 | 46.4 ± 6.5 | 50.2 ± 6.7 |
| CamemBERT + Flair Fr | fastT + BPEmb | 53.4 ± 6.0 | 47.0 ± 5.8 | 49.8 ± 5.8 |
| CamemBERT + Flair Fr | fastT + BPEmb + Char | 58.5 ± 4.9 | 48.3 ± 3.9 | 52.6 ± 4.2 |
| CamemBERT + Flair Fr | All | 59.6 ± 5.4 | 50.9 ± 4.7 | 54.6 ± 4.8 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 49.5 ± 6.3 | 42.5 ± 6.8 | 45.3 ± 6.2 |
| None | BPEmb + Char | 58.0 ± 6.0 | 49.2 ± 3.6 | 52.8 ± 4.3 |
| None | All | 52.1 ± 5.4 | 44.6 ± 3.9 | 47.6 ± 4.1 |
| mBERT | None | 62.3 ± 7.3 | 39.7 ± 3.4 | 48.3 ± 4.7 |
| mBERT | Char | 57.4 ± 6.0 | 36.3 ± 4.3 | 44.3 ± 4.8 |
| mBERT | OHE | 58.0 ± 4.3 | 39.5 ± 3.5 | 46.9 ± 3.6 |
| mBERT | BPEmb | 59.9 ± 4.7 | 49.6 ± 4.6 | 53.9 ± 4.3 |
| mBERT | BPEmb + Char | 58.8 ± 5.2 | 47.9 ± 4.5 | 52.6 ± 4.5 |
| mBERT | All | 59.9 ± 4.4 | 49.0 ± 3.8 | 53.7 ± 3.8 |
| mFlair | None | 50.3 ± 5.8 | 28.6 ± 5.2 | 36.2 ± 5.8 |
| mFlair | Char | 52.3 ± 5.8 | 28.2 ± 4.9 | 36.1 ± 5.1 |
| mFlair | OHE | 42.1 ± 6.0 | 27.0 ± 3.2 | 32.7 ± 4.0 |
| mFlair | BPEmb | 59.7 ± 5.7 | 49.0 ± 5.1 | 53.7 ± 5.3 |
| mFlair | BPEmb + Char | 59.9 ± 4.7 | 48.0 ± 4.3 | 53.2 ± 4.3 |
| mFlair | All | 60.6 ± 6.6 | 49.6 ± 5.3 | 54.5 ± 5.8 |
| mBERT + mFlair | None | 62.4 ± 2.6 | 41.5 ± 2.7 | 49.6 ± 2.2 |
| mBERT + mFlair | Char | 61.3 ± 3.8 | 39.6 ± 3.4 | 47.8 ± 3.1 |

Table B.2: Overview of all results obtained on the French Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5). *(continued)*

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| mBERT + mFlair | OHE | 57.6 ± 7.2 | 37.1 ± 2.9 | 44.8 ± 4.0 |
| mBERT + mFlair | BPEmb | 59.4 ± 3.9 | 48.9 ± 4.1 | 53.3 ± 3.6 |
| mBERT + mFlair | BPEmb + Char | 61.1 ± 4.9 | 47.6 ± 3.3 | 53.2 ± 3.3 |
| mBERT + mFlair | All | 61.4 ± 3.7 | 47.0 ± 2.7 | 53.0 ± 2.7 |

Table B.3: Overview of all results obtained on the multilingual (Nl + Fr) Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5).

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| **Only task-specific embeddings** | | | | |
| None | Char | 22.9 ± 5.8 | 5.5 ± 1.5 | 8.8 ± 2.3 |
| None | OHE | 55.8 ± 2.4 | 31.7 ± 2.3 | 40.1 ± 2.0 |
| **Dutch embeddings** | | | | |
| None | BPEmb | 57.4 ± 1.4 | 39.1 ± 2.5 | 46.5 ± 2.1 |
| None | fastT | 60.4 ± 4.1 | 40.5 ± 4.2 | 48.3 ± 4.3 |
| None | fastT + BPEmb | 66.5 ± 3.8 | 50.7 ± 3.9 | 57.2 ± 3.8 |
| None | fastT + BPEmb + Char | 67.7 ± 2.7 | 51.2 ± 4.4 | 57.9 ± 3.7 |
| None | All | 67.3 ± 3.4 | 51.7 ± 4.3 | 58.3 ± 3.9 |
| BERTje | None | 67.8 ± 3.6 | 32.5 ± 1.8 | 43.7 ± 1.6 |
| BERTje | Char | 64.4 ± 5.5 | 29.7 ± 2.1 | 40.4 ± 2.7 |
| BERTje | OHE | 64.2 ± 2.2 | 29.1 ± 2.0 | 39.9 ± 2.0 |
| BERTje | BPEmb | 62.5 ± 2.2 | 37.0 ± 3.3 | 46.2 ± 3.1 |
| BERTje | fastT | 68.5 ± 4.2 | 38.9 ± 2.6 | 49.2 ± 2.6 |
| BERTje | fastT + BPEmb | 69.0 ± 3.2 | 43.7 ± 3.2 | 53.3 ± 3.1 |
| BERTje | fastT + BPEmb + Char | 68.7 ± 2.5 | 42.6 ± 2.0 | 52.4 ± 1.6 |
| BERTje | All | 65.9 ± 2.8 | 41.0 ± 3.3 | 50.2 ± 2.8 |
| Flair Nl | None | 67.1 ± 2.6 | 43.9 ± 3.6 | 52.7 ± 2.8 |
| Flair Nl | Char | 67.1 ± 1.6 | 44.7 ± 4.1 | 53.2 ± 3.0 |
| Flair Nl | OHE | 68.5 ± 3.2 | 46.7 ± 3.2 | 55.2 ± 2.5 |
| Flair Nl | BPEmb | 67.3 ± 2.9 | 49.7 ± 3.1 | 56.8 ± 2.4 |
| Flair Nl | fastT | 67.5 ± 3.9 | 49.9 ± 3.3 | 57.2 ± 3.3 |
| Flair Nl | fastT + BPEmb | 66.7 ± 2.5 | 52.0 ± 3.2 | 58.2 ± 2.6 |
| Flair Nl | fastT + BPEmb + Char | 66.7 ± 1.8 | 51.5 ± 2.9 | 57.9 ± 2.1 |
| Flair Nl | All | 69.8 ± 2.1 | 54.5 ± 1.6 | 61.1 ± 1.4 |
| BERTje + Flair Nl | None | 67.1 ± 2.1 | 28.7 ± 1.8 | 40.1 ± 2.0 |
| BERTje + Flair Nl | Char | 64.6 ± 3.7 | 29.4 ± 2.5 | 40.3 ± 2.9 |
| BERTje + Flair Nl | OHE | 64.8 ± 2.3 | 26.2 ± 2.6 | 36.9 ± 2.4 |
| BERTje + Flair Nl | BPEmb | 65.7 ± 1.6 | 32.1 ± 1.6 | 43.1 ± 1.6 |
| BERTje + Flair Nl | fastT | 69.3 ± 3.4 | 34.5 ± 1.8 | 45.9 ± 1.8 |
| BERTje + Flair Nl | fastT + BPEmb | 67.6 ± 3.2 | 42.4 ± 3.6 | 51.8 ± 3.2 |
| BERTje + Flair Nl | fastT + BPEmb + Char | 66.4 ± 1.4 | 39.5 ± 2.6 | 49.3 ± 2.1 |
| BERTje + Flair Nl | All | 67.7 ± 2.6 | 39.5 ± 3.3 | 49.7 ± 3.0 |
| **French embeddings** | | | | |
| None | BPEmb | 55.4 ± 1.5 | 35.2 ± 1.3 | 43.0 ± 1.3 |
| None | fastT | 56.8 ± 3.2 | 40.1 ± 3.6 | 46.8 ± 3.5 |
| None | fastT + BPEmb | 66.3 ± 1.7 | 52.0 ± 3.4 | 58.2 ± 2.8 |
| None | fastT + BPEmb + Char | 65.6 ± 2.9 | 52.0 ± 4.0 | 58.0 ± 3.6 |
| None | All | 66.0 ± 2.6 | 51.5 ± 3.4 | 57.7 ± 3.0 |
| CamemBERT | None | 52.8 ± 5.2 | 22.2 ± 1.6 | 30.9 ± 1.8 |
| CamemBERT | Char | 62.4 ± 2.4 | 29.8 ± 2.8 | 40.0 ± 2.7 |
| CamemBERT | OHE | 62.6 ± 2.5 | 44.8 ± 2.4 | 52.0 ± 1.7 |
| CamemBERT | BPEmb | 64.3 ± 1.8 | 50.2 ± 2.6 | 56.2 ± 1.9 |
| CamemBERT | fastT | 68.2 ± 2.1 | 50.2 ± 3.2 | 57.6 ± 2.5 |
| CamemBERT | fastT + BPEmb | 67.8 ± 3.7 | 52.8 ± 3.8 | 59.3 ± 3.7 |
| CamemBERT | fastT + BPEmb + Char | 68.4 ± 3.1 | 53.0 ± 3.0 | 59.6 ± 2.9 |
| CamemBERT | All | 67.4 ± 3.6 | 53.4 ± 3.0 | 59.5 ± 2.8 |
| Flair Fr | None | 61.9 ± 4.1 | 28.0 ± 2.9 | 38.0 ± 2.8 |
| Flair Fr | Char | 63.3 ± 1.2 | 31.4 ± 3.0 | 41.6 ± 2.9 |
| Flair Fr | OHE | 67.9 ± 4.6 | 41.6 ± 4.0 | 51.1 ± 3.8 |
| Flair Fr | BPEmb | 67.5 ± 4.4 | 46.5 ± 2.4 | 54.9 ± 2.8 |

Table B.3: Overview of all results obtained on the multilingual (Nl + Fr) Faktion dataset. All results are reported as mean ± standard error of the mean (k = 5). *(continued)*

| Contextualized embedding | Static/task-spec. embedding | Precision | Recall | F1-score |
|---|---|---|---|---|
| Flair Fr | fastT | 64.4 ± 1.5 | 44.8 ± 4.2 | 52.4 ± 2.9 |
| Flair Fr | fastT + BPEmb | 68.1 ± 3.0 | 51.7 ± 4.6 | 58.5 ± 4.1 |
| Flair Fr | fastT + BPEmb + Char | 66.4 ± 3.4 | 49.2 ± 4.2 | 56.3 ± 3.9 |
| Flair Fr | All | 68.4 ± 3.9 | 49.9 ± 3.9 | 57.3 ± 3.3 |
| CamemBERT + Flair Fr | None | 72.0 ± 4.4 | 40.6 ± 3.8 | 51.6 ± 3.7 |
| CamemBERT + Flair Fr | Char | 69.4 ± 1.0 | 40.0 ± 2.0 | 50.6 ± 1.6 |
| CamemBERT + Flair Fr | OHE | 70.1 ± 2.3 | 40.6 ± 3.0 | 51.1 ± 2.5 |
| CamemBERT + Flair Fr | BPEmb | 68.1 ± 3.6 | 50.1 ± 2.5 | 57.5 ± 2.4 |
| CamemBERT + Flair Fr | fastT | 69.0 ± 3.2 | 47.9 ± 3.0 | 56.3 ± 2.3 |
| CamemBERT + Flair Fr | fastT + BPEmb | 67.4 ± 3.9 | 49.3 ± 3.4 | 56.7 ± 3.2 |
| CamemBERT + Flair Fr | fastT + BPEmb + Char | 66.5 ± 3.1 | 49.1 ± 2.6 | 56.3 ± 2.4 |
| CamemBERT + Flair Fr | All | 68.3 ± 1.8 | 51.8 ± 3.0 | 58.6 ± 1.6 |
| **Multilingual embeddings** | | | | |
| None | BPEmb | 69.0 ± 3.4 | 52.7 ± 4.1 | 59.3 ± 3.5 |
| None | BPEmb + Char | 69.4 ± 3.6 | 51.1 ± 2.9 | 58.5 ± 2.4 |
| None | All | 73.7 ± 4.6 | 53.3 ± 3.4 | 61.6 ± 3.5 |
| mBERT | None | 68.1 ± 2.3 | 45.8 ± 2.6 | 54.6 ± 2.3 |
| mBERT | Char | 70.6 ± 2.5 | 46.7 ± 2.8 | 56.1 ± 2.6 |
| mBERT | OHE | 72.3 ± 2.9 | 45.5 ± 2.9 | 55.7 ± 2.7 |
| mBERT | BPEmb | 71.5 ± 2.2 | 54.3 ± 2.1 | 61.6 ± 1.8 |
| mBERT | BPEmb + Char | 74.6 ± 2.6 | 54.4 ± 2.1 | 62.8 ± 1.9 |
| mBERT | All | 72.3 ± 3.2 | 52.3 ± 2.9 | 60.2 ± 2.0 |
| mFlair | None | 68.2 ± 4.6 | 35.5 ± 3.1 | 46.3 ± 2.9 |
| mFlair | Char | 66.1 ± 5.1 | 34.2 ± 1.8 | 44.9 ± 2.5 |
| mFlair | OHE | 69.3 ± 4.3 | 37.7 ± 2.7 | 48.5 ± 2.7 |
| mFlair | BPEmb | 70.4 ± 3.7 | 51.1 ± 3.8 | 58.8 ± 3.2 |
| mFlair | BPEmb + Char | 69.6 ± 3.2 | 52.5 ± 4.6 | 59.5 ± 3.8 |
| mFlair | All | 69.0 ± 2.9 | 50.8 ± 3.9 | 58.3 ± 3.3 |
| mBERT + mFlair | None | 71.0 ± 2.6 | 41.8 ± 1.6 | 52.5 ± 1.5 |
| mBERT + mFlair | Char | 72.7 ± 2.6 | 41.8 ± 2.5 | 52.9 ± 2.3 |
| mBERT + mFlair | OHE | 72.6 ± 2.3 | 44.4 ± 2.3 | 54.9 ± 2.0 |
| mBERT + mFlair | BPEmb | 70.6 ± 3.7 | 51.4 ± 2.8 | 59.4 ± 3.0 |
| mBERT + mFlair | BPEmb + Char | 70.4 ± 4.1 | 50.9 ± 3.1 | 58.9 ± 3.2 |
| mBERT + mFlair | All | 71.1 ± 3.3 | 51.2 ± 3.2 | 59.4 ± 3.0 |

# Bibliography


Agić, Ž. and Vulić, I. (2019). JW300: A Wide-Coverage Parallel Corpus for Low-Resource Languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3204–3210, Florence, Italy. Association for Computational Linguistics.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., and Vollgraf, R. (2019). FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. (2018). Linear Algebraic Structure of Word Senses, with Applications to Polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

Artetxe, M., Labaka, G., and Agirre, E. (2018). A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Carreras, X., Màrquez, L., and Padró, L. (2002). Named Entity Extraction using AdaBoost. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Chinchor, N. and Sundheim, B. (1993). MUC-5 Evaluation Metrics. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*.

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

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Esuli, A. and Sebastiani, F. (2010). Evaluating Information Extraction. In Agosti, M., Ferro, N., Peters, C., de Rijke, M., and Smeaton, A., editors, *Multilingual and Multimodal Information Access Evaluation*, Lecture Notes in Computer Science, pages 100–111, Berlin, Heidelberg. Springer.

Ethayarajh, K. (2019). How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Firth, J. R. (1957). *A Synopsis of Linguistic Theory 1930-1955*. Philological Society, Oxford.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named Entity Recognition through Classifier Combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 168–171.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*.

Heinzerling, B. and Strube, M. (2017). BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. *arXiv:1710.02187 [cs]*.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing*. 3th ed. draft edition.

Lafferty, J., Mccallum, A., and Pereira, F. (2001). *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*.

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

Lample, G., Conneau, A., Ranzato, M., Denoyer, L., and Jégou, H. (2018). Word translation without parallel data.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics (Oxford, England)*, 36(4):1234–1240.

Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. (2019). Linguistic Knowledge and Transferability of Contextual Representations. *arXiv:1903.08855 [cs]*.

Ma, X. and Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv:1603.01354 [cs, stat]*.

Manning, C. (2019a). *CS224n: Natural Language Processing with Deep Learning*. Stanford NLP.

Manning, C. (2019b). Stanford CS 224N | Natural Language Processing with Deep Learning. http://web.stanford.edu/class/cs224n/.

Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D., and Sagot, B. (2019). CamemBERT: A Tasty French Language Model. *arXiv preprint arXiv:1911.03894*.

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

Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J. R. (2013). Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is Multilingual BERT? *arXiv preprint arXiv:1906.01502*.

Reimers, N. and Gurevych, I. (2017). Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. *arXiv:1707.09861 [cs, stat]*.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

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

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wang, H., Yu, D., Sun, K., Chen, J., and Yu, D. (2019). Improving Pre-Trained Multilingual Model with Vocabulary Expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China. Association for Computational Linguistics.

Wu, S. and Dredze, M. (2019). Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.