How Is Context Important in Named Entity Recognition? A Comparison of Non-contextual and Contextual Word Embeddings

Dawid Smalcuga[1], **Piotr Andruszkiewicz**[1,2]

[1]Warsaw University of Technology, [2]IDEAS Research Institute
dawid.smalcuga.stud@pw.edu.pl, piotr.andruszkiewicz@pw.edu.pl

Abstract

The paper compares non-contextual and contextual word embeddings in Named Entity Recognition (NER) task. Word embeddings created by the GloVe, ELMo, BERT and RoBERTa models and the named entities predicted by the LUKE model have been tested. Models based on LSTM recursive neural network and convolutional neural network (CNN) have been created. They use vector representations and are being trained to recognize named entities in text. Using these models, the impact of word embeddings in the Named Entity Recognition task has been examined. The datasets used in the experiments are the Annotated Corpus for Named Entity Recognition and CoNLL-2003. We have investigated the importance of the size of context and which vector representation performs best in Named Entity Recognition. The experiments, we have conducted, prove that contextual word embeddings show their advantage if the context is longer than one sentence. Moreover, BERT and especially RoBERTa perform significantly worse than other models for entity types with small number of instances. Another finding is that cased BERT model achieves better results than its uncased counterpart.

1 Introduction

In this paper, Named Entity Recognition (NER), one of the tasks of Natural Language Processing (NLP), is examined. The NER tak is important in many applications of NLP, for instance, in analysis of user's utterances/commands passed to an intelligent agent. Such utterances are transformed with Automatic Speech Recognition (ASR) and then processed with NER to improve quality of tasks performed according to user's requests, e.g., Question Answering, making reservations, etc.

To investigate the impact of the size of the context on the NER task, we used two well-known benchmark datasets; namely, the Annotated Corpus for Named Entity Recognition (Walia) and CoNLL-2003 (Sang and Meulder, 2003). The main difference between these sets is the length of the context. The former dataset consists of single sentences, so that the context of each word is limited only to the sentence in which it occurs. The latter dataset consists of long documents where the meaning of each word depends on the broad context. The models that create the embedding vectors are: non-contextual GloVe (Pennington et al., 2014), contextual ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and RoBERTa (Liu et al., 2019), which create word representations, and contextual LUKE (Yamada et al., 2020), which creates entity representations. These models differ in construction, in particular GloVe uses statistics. ELMo uses recursive neural networks, while the other three models, BERT, RoBERTa and LUKE, are based on transformers.

The aim of this work is to examine the impact of the choice of a model that creates representations of words or entities in the NER task regarding the available context. Thus, models creating non-contextual and contextual word embedding vectors are to be examined. We use the word embedding vectors created by GloVe, ELMo, BERT and RoBERTa (words) and LUKE (entities), and then train a model based on neural networks in the task of Name Entity Recognition. We focus on differences between non-contextual and contextual embeddings in regards to the available context in the data. Furthermore, we investigate the difference between cased and uncased models that produce word embedding vectors.

The source code and the data are available at the following website `https://staff.elka.pw.edu.pl/~pandrusz/data/contexte/`.

2 Related Work

Named Entity Recognition (NER) task has been recently studied in the context of Deep Neural Net-

Model	F1
LSTM-CRF (Lample et al., 2016)	91.0
ELMo (Peters et al., 2018)	92.2
BERT (Devlin et al., 2019)	92.8
Akbik, Blythe, Vollgraf (2018) (Akbik et al., 2018)	93.1
Baevski, Edunov, Liu, Zettlemoyer, Auli (2019) (Baevski et al., 2019)	93.5
RoBERTa (Liu et al., 2019)	92.4
LUKE (Yamada et al., 2020)	94.3

Table 1: Named Entity Recognition results from (Yamada et al., 2020).

works (Li et al., 2022). As different network architectures emerge, they are applied in NER task. Beneath the architectures, there are also word representations – embedding vectors – which are important in recognition of different entities types.

The first well-known representation using embedding vectors is Word2Vec (Mikolov et al., 2013). This solution creates similar vectors for semantically close words and uses two standard approaches: skip-gram and CBOW. After popularization of Word2Vec, more solutions appeared quickly, including very popular GloVe (Pennington et al., 2014) or FastText (Bojanowski et al., 2017).

In GloVe, learning is based on the probability of occurrence of a given word. To estimate this probability a table of mutual words occurrence is calculated. Then, we count how many times every word has appeared in the context of a given word. The context in the case is a "frame" with a width of 3 - the word preceding, a given word, and the following word.

The above models, produce the same vectors for a word no matter the context a word appears in. Thus, contextual vector embeddings were proposed. Contextual vectors are the extension of the described solutions. These models generate different vectors for the same word depending on the context of the word, i.e. words appearing next to the analyzed word.

ELMo (Peters et al., 2018) model uses recursive neural networks – LSTMs – to generate vectors. It process proceeding and following neighborhood by applying bidirectional LSTMs. A different type of contextual word embeddings models is based on transformers. The well known representatives of this type of solution are: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and LUKE (Yamada et al., 2020).

BERT drops recurrent mechanism and uses self attention. This model was later improved by, e.g., different pre-training procedure in RoBERTa and further improvements were incorporated in LUKE model.

LUKE treats words and entities in the text as independent tokens and generates their representations. It was tested on the CONLL-2003 dataset. This model achieved the highest F1 result at the time of its publication (please refer to Table 1). The model finds all possible ranges of entities in every sentence and then classifies them as one of the defined types of named entities or no entity. The representation of each entity is created by the alignment of the representation of the first and last word in the span and representation of the entity corresponding to the span. The maximum length of the entity span is 16 words, while the context is 512 tokens.

As there are many types of embeddings, a question arises – which type is the best in a given task. This aspect has been studied in the context of emotion detection (Polignano et al., 2019) and biomedical Natural Language Processing (Wang et al., 2018). We investigate the types of embedding vectors in Named Entity Recognition task for English.

3 Model for Word Embeddings Comparision

In this section, we present a description of the model we built for examining the impact of the method used for creating representations of words and entities in the task of NER. It uses the word embedding vectors created by the models described in previous sections, and then trains neural networks to recognize named entity. The number of entity types depends on the dataset we use and amounts to 8 for the Annotated Corpus for Named Entity Recognition called Kaggle (Walia) and 4 for CoNLL-2003 (Sang and Meulder, 2003).

Modern models, after appropriate training,

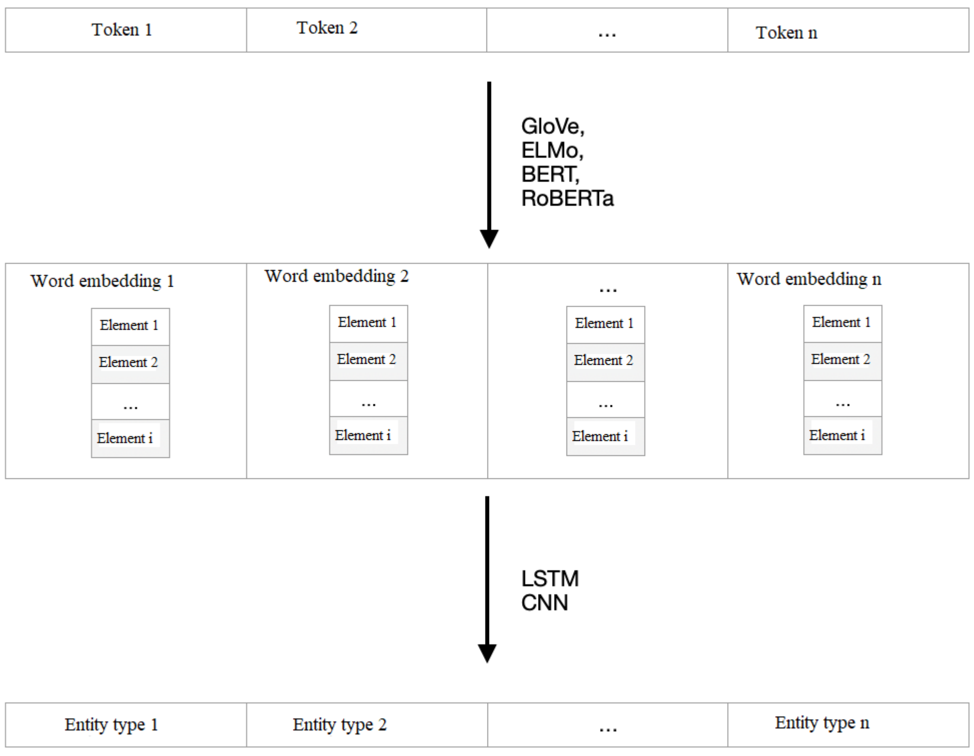

Figure 1: Model architecture using GloVe, ELMo, BERT or RoBERTa.

achieve very good results in the CoNLL-2003 ranking. For example, the LUKE achieves F1 score of 94.3, placing it at the top of the ranking. The idea in our research is to compare only the word embedding vectors, which are not adapted to NER task, not the whole models trained for NER task.

The architecture of the model we created is shown in Figure 1. LUKE, on the other hand, creates entity representations that are not available. For this reason, we use a version of the LUKE model that has been trained on the CoNLL-2003 dataset and its output, which is an entity type, instead of multidimensional vectors that are used to predict an entity type. The LUKE's output is processed in the same way as the model treats word embeddings – they are the input of the neural network.

3.1 Word Embedding Vectors

The first part of the model, we created, is a vector representation provided by one of the models:

GloVe trained on Wikipedia from 2014 and Gigaword 5 with the largest, 300-dimensional vector size.

ELMo and the following models generate contextual vectors. We used ELMo Medium, whose parameters are: total 28 million trainable parameters, 512-dimensional embeding vectors.

BERT is the most popular model in NLP. Thus, we decided to investigate the effect of two different versions of this model available in the Transformers library[1]: bert-base-cased and bert-base-uncased. The parameters of these models are as follows: 12 layers of transformers, 768 hidden units, a total of 110 million trainable parameters, 768-dimensional vectors. The *cased* model was trained on text consisting of lowercase and uppercase letters, while the text used to train the *uncased* model consisted only of lowercase letters. When not explicitly specified, BERT-cased has been used.

The RoBERT model is also from the Transformers library. The version we are using is roberta-base, which has the same number of parameters as bert-base.

The last model, LUKE, is also available in the Transformers library. It uses the *studio-ousia/luke-large-finetuned-conll-2003* version that was fine-tuned on the CoNLL-2003 dataset. It is not possible to use this model to classify the Kaggle dataset because the named entity types are already defined. There are no embedding vectors in the output, because the model is adapted to predict en-

[1]https://huggingface.co/docs/transformers/index

Word embeddings	Network type	Prec. macro	Recall macro	F1 macro	Prec. micro	Recall micro	F1 micro	Acc.
GloVe	LSTM	**68.32**	**60.79**	**62.83**	75.30	78.96	77.09	96.24
ELMo	LSTM	58.63	57.38	57.72	**78.26**	**80.67**	**79.45**	**96.60**
BERT	LSTM	56.49	52.15	52.88	75.42	77.15	76.27	96.00
RoBERTa	LSTM	47.00	41.99	42.95	63.73	65.93	64.81	94.25
GloVe	CNN	55.23	52.21	52.75	68.32	72.18	70.20	94.59
ELMo	CNN	56.84	53.03	53.90	73.90	76.63	75.24	95.74
BERT	CNN	46.70	46.02	45.48	71.03	74.11	72.54	95.28
RoBERTa	CNN	41.85	36.09	36.55	56.68	58.50	57.57	92.93

Table 2: Results for Kaggle set (the best results marked in bold).

Word embed.	Network type	art	eve	geo	gpe	nat	org	per	tim
GloVe	LSTM	**18.92**	38.71	82.55	**94.21**	**59.65**	56.86	72.86	78.85
ELMo	LSTM	2.44	25.32	**84.15**	92.54	37.21	**62.45**	**73.45**	**84.21**
BERT	LSTM	0.00	28.57	82.71	87.38	15.38	56.46	70.16	82.36
RoBERTa	LSTM	0.00	28.12	72.93	67.01	0.00	39.81	58.77	76.95
GloVe	CNN	0.00	**40.00**	78.90	93.20	34.38	44.69	60.37	70.50
ELMo	CNN	2.47	23.68	80.46	91.30	30.00	54.55	72.17	77.53
BERT	CNN	0.00	0.00	79.33	86.03	5.41	47.85	66.20	79.04
RoBERTa	CNN	0.00	11.54	66.12	65.21	0.00	30.57	48.29	70.66

Table 3: Results for entity types in Kaggle set.

tity types. The number at the model output, corresponding to the expected entity type, is treated as a 1-dimensional vector and is the input of the neural network. The idea behind this solution is to use all models in the same way, i.e., to train LSTM or CNN networks based on the created representations. Due to the lack of word embedding representation for the LUKE model, the result predicted by it was used.

3.2 Trainable Neural Network Part of Model

The word embedding vectors are prepared just before entering the neural network. They are then processed by recursive neural network LSTM or convolutional neural network (CNN). In the next step, the result goes through the dropout layer with a value of 0.3. Finally, the values are linearly transformed and the logarithm of the softmax function is returned.

Batch sizes were 4 for training and 2 for validation and test. The hidden layer size of LSTM and CNN was 100. The learning rate was 10^{-3}, while the number of iterations was 5. The Adam optimizer was used.

3.3 Differences Between the Proposed Solution and the Existing Models

It is worth emphasizing the differences between the proposed solution and the models trained to detect named entities, which are mentioned in Table 1. Figure 1 shows the architecture of the model, where we can see that the embedding vectors are used by the LSTM and CNN networks. The goal of such a solution is to use the same mechanism for all vector representations. It seems that such a procedure may significantly reduce the efficiency of Named Entity Recognition, because single layers of neural networks did not achieve good results in the NER task, which indicates the level of complexity of the models described in Table 1. Models presented in this table were finetuned on the CoNLL-2003 dataset. Nevertheless, the purpose of this work is to compare the effects of word embedding vectors only, not to achieve the best result.

4 Experiments

This section presents the results of experiments we conducted in our study. We present the results over two NER datasets: Annotated Corpus for Named Entity Recognition called Kaggle (Walia) and CoNLL-2003 (Sang and Meulder, 2003).

Word embeddings	Network type	Prec. macro	Recall macro	F1 macro	Prec. micro	Recall micro	F1 micro	Acc.
GloVe	LSTM	58.18	67.79	61.47	56.84	67.79	61.83	92.56
ELMo	LSTM	82.32	86.35	84.21	84.46	88.24	86.31	97.46
BERT	LSTM	80.00	79.63	79.61	81.15	83.25	82.19	96.16
RoBERTa	LSTM	67.72	70.78	69.12	70.71	74.58	72.59	94.51
LUKE	LSTM	**92.64**	**92.78**	**92.71**	**94.05**	**94.02**	**94.03**	**98.66**
GloVe	CNN	48.71	57.03	49.37	36.50	56.29	44.28	89.08
ELMo	CNN	76.08	80.56	78.05	75.54	82.08	78.68	96.51
BERT	CNN	71.80	72.92	71.90	70.08	75.43	72.66	94.46
RoBERTa	CNN	61.04	62.80	61.12	61.41	67.23	64.19	92.84
LUKE	CNN	85.98	88.71	87.26	88.83	92.10	90.44	98.33

Table 4: Results for CoNLL-2003 set.

4.1 Description of Tests

On the Kaggle dataset, the following combinations of the vector representation models: GloVe, ELMo, BERT, RoBERTa and the type of neural network: LSTM, CNN were tested. On the other hand, on the CoNLL-2003 dataset, in addition to the above-mentioned algorithms, LUKE was also tested. In each case, the precision, recall and F1 score were calculated, divided into micro and macro, and accuracy. In addition, the F1 results of each category were calculated in order to examine which labels the model copes with the worst.

We divided the dataset into three parts: 70% - training, 15% - validation, 15% - testing.

4.2 Evaluation method

The word embedding vectors created by GloVe, ELMo, BERT and RoBERTa represent words, not entities. For this reason, we used the IOB2 format.

Assuming that we count the detection of entire entities, we can evaluate the operation of the model, i.e., the entity must be accurately predicted to be successful. According to this scheme we calculate the precision and recall, and thus - the F1 measures. This is a method published with the CoNLL-2003 dataset. In all experiments in the paper, averages of 5 runs are presented. The best results are marked with bold.

4.3 Research Results on the Kaggle Dataset

The results of the experiments on the Kaggle dataset are presented in Tables 2 and 3.

The first observation during the analysis of the results for the Kaggle set is high accuracy of all cases. There are no big differences between its values in subsequent rows of the tables. Therefore, comparing models based on accuracy seems not to be a good idea. The F1 micro results oscillate between 64.81 and 79.45 for the LSTM neural network and 57.57 – 75.24 for the CNN, while the macro F1 results between 42.95 – 62.83 and 36.55 – 53.90. The difference between micro and macro is greater than 20%, which shows that there are large discrepancies in the number of elements of different entity types. Definitely better results are achieved by a model using a recursive type of neural network. This is not a surprise, because recursion is perfect for language learning, because a given word depends on those that occurred earlier.

In both cases – LSTM and CNN, the best micro results are achieved by the model using the embedding vectors created by ELMo. It is ranked first in the ranking of all the measures used, achieving F1 micro scores of 79.45 and 75.24, respectively.

However, in the case of macro average, the best F1 result is achieved by GloVe with LSTM equal to 62.83, and with CNN it is only 0.85 pp lower than the best result achieved by ELMo. It is worth noting the high value of GloVe macro precision with LSTM, which is equal to 68.32. The results of GloVe average macro show that the word embedding vectors have a greater impact on the ability to detect named entities of different types with more similar effectiveness, regardless of the number of elements in a given entity type. It may be surprising that the model using GloVe word embedding vectors seems to work so well. However, keeping in mind that the context of words in the Kaggle datasset is very limited, for this reason, context models do not have significant advantage over non-contextual models.

As for the BERT model, it ranks third for LSTM

Word emb.	Net.	loc	misc	org	per
GloVe	LSTM	67.62	52.48	55.07	70.72
ELMo	LSTM	89.05	70.05	81.89	95.86
BERT	LSTM	85.66	64.81	78.78	89.20
RoBERTa	LSTM	76.77	50.07	68.77	80.87
LUKE	LSTM	94.96	**84.60**	**93.50**	**97.77**
GloVe	CNN	74.96	62.18	32.49	27.85
ELMo	CNN	84.08	70.82	66.91	90.37
BERT	CNN	79.27	65.50	65.51	77.31
RoBERTa	CNN	71.29	45.08	58.03	70.08
LUKE	CNN	**95.12**	67.61	88.57	97.74

Table 5: Results for CoNLL-2003 set per entity type.

and second for CNN in micro results. It achieves the largest difference between the average micro and macro F1 results, which for the CNN case is as much as 27.06 pp. This shows how much it relies on a similar distribution of elements in entity types.

The worst results in all cases were achieved by the RoBERTa model. It is worth recalling that according to Table 1 the RoBERTa model, despite being an improved BERT model, performs worse in the task of NER compared to BERT. However, the difference is only 0.4 pp. Nevertheless, the results, we obtained, of the word embedding vectors created by RoBERTa are surprisingly low with a significant margin to BERT (around 9-13 pp.).

Table 3 shows the F1 results of each label for each case tested. For the entity type "art", the results of the most test cases are equal to 0. GloVe with LSTM achieved an F1 score significantly higher than other models, but still very low – 18.92%. The number of elements of the "art" category is only 59, which to some extent explains the difficulties in learning how to recognize this type of entity by the model. Two other categories with a small number of elements are "nat" and "eve", respectively 30 and 56. In these cases, the F1 scores of all models are definitely lower than for more numerous categories, leading to the same conclusion. In addition, it can be seen that the BERT and RoBERTa models achieve significantly lower results than other models in detecting "nat" entities.

In summary, the best embedding vector model for the Kaggle dataset is ELMo, because it achieves the best average micro scores. Nevertheless, the GloVe model is best at predicting different entity types, on average. The worst results were achieved by the RoBERTa model and, similarly to the BERT model, it performs significantly worse for entity types with small number of instances.

4.4 Research Results on the CoNLL-2003 Dataset

The first noticeable difference between the research results on the CoNLL-2003 dataset (Tables 4 and 5) and Kaggle is that the GloVe model achieves the lowest results in all categories. This is because the CoNLL-2003 dataset is divided into documents, so the context of each word is much broader than Kaggle's. Small context is the only advantage of non-contextual embedding vectors over contextual ones.

An additional model tested on the CoNLL-2003 dataset is LUKE. It is a model created for tasks dealing with entities, which achieved by far the best results in our study. The F1 result of the micro combination of LUKE with CNN is better than the second result for this type of network by as much as 11.76 pp. ELMo is in the second place and has a few percent advantage over the BERT model. The last but one place is occupied by the RoBERTa model, with much better results than the last GloVe. It is worth paying attention to the small discrepancy in the average micro and macro results. The difference between the extreme cases of F1 measure is 3.47 pp. for LSTM and 5.09 pp. for CNN. This is due to the similar number of elements in all categories, where the only category with a different number of elements is "MISC" (around twice smaller than other categories).

Table 5 presents F1 results for individual labels. The distribution for GloVe with LSTM is significantly different from that for GloVe with CNN, especially for "PER" type, where the difference is 42.87 pp. Other models achieve significantly better results for this type of entity, and the LUKE model

Word embeddings	Network type	Prec. macro	Recall macro	F1 macro	Prec. micro	Recall micro	F1 micro	Acc.
BERT-cased	LSTM	**56.49**	**52.15**	**52.88**	**75.42**	**77.15**	**76.27**	**96.00**
BERT-uncased	LSTM	50.89	47.64	48.38	73.02	73.56	73.29	95.07
BERT-cased	CNN	46.70	46.02	45.48	71.03	74.11	72.54	95.28
BERT-uncased	CNN	46.11	43.08	43.41	68.80	68.25	68.52	94.28

Table 6: Results of BERT-cased and BERT-uncased for Kaggle set.

Word embeddings	Network type	Prec. macro	Recall macro	F1 macro	Prec. micro	Recall micro	F1 micro	Acc.
BERT-cased	LSTM	**80.00**	**79.63**	**79.61**	**81.15**	**83.25**	**82.19**	**96.16**
BERT-uncased	LSTM	76.82	75.20	75.91	78.75	78.33	78.54	95.23
BERT-cased	CNN	71.80	72.92	71.90	70.08	75.43	72.66	94.46
BERT-uncased	CNN	70.26	71.51	71.12	70.86	71.51	71.18	94.01

Table 7: Results of BERT-cased and BERT-uncased for CoNLL-2003 set.

yields very high scores: 97.77 for LSTM and 97.74 for CNN. The worst scores can be observed for the "MISC" category. It is around twice as numerous as the others, and this is the reason for the worse results.

In conclusion, the LUKE model is by far the best choice of all the models tested. However, the system uses its ability to create entity representations. For word representations, the ELMo model achieves the best results, similarly to the Kaggle dataset. This model is followed by BERT, RoBERTa and GLoVe. The latter model yields significantly worse results as it is the only one model that creates non-contextual embeddings and the CoNLL-2003 dataset consists of long documents, thus long context is available.

Another finding is that RoBERTa underperforms compared to BERT. Despite being an optimized version of BERT, RoBERTa does not always outperform it in Named Entity Recognition (NER) tasks, please refer to Table 1. Several factors contribute to this. Firstly, there are differences in pretraining. RoBERTa removes the Next Sentence Prediction (NSP) task present in BERT. While studies have suggested that NSP is not critical for many NLP tasks, it might play a role in NER, where cross-sentence context can be important. Secondly, RoBERTa performs worse than BERT in NER tasks on both the CoNLL-2003 and Kaggle datasets. RoBERTa struggles particularly with less frequent entity types such as "nat" (nationalities) and "art" (artifacts). BERT, with NSP training, may be better at capturing global document-level

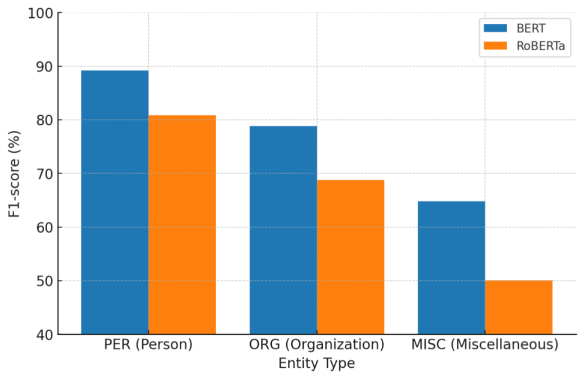

Figure 2: A bar chart comparing F1-scores for different entity categories with separate bars for BERT and RoBERTa on the CoNLL-2003 dataset.

context. Thirdly, RoBERTa employs an improved subword encoding mechanism – Byte-Pair Encoding at the byte level, which enhances generalization for rare words. However, in NER, this can lead to excessive fragmentation of named entities, making them harder to classify correctly.

An F1-score comparison, please refer to Figure 2, indicates that RoBERTa performs worse across all entity types on the CoNLL-2003 dataset.

RoBERTa's optimizations impact computational requirements in various ways. RoBERTa was trained on 160GB of raw text, whereas BERT used 13GB, requiring significantly more memory and compute power. Eliminating Next Sentence Prediction (NSP) should theoretically improve computational efficiency, but it does not necessarily enhance NER performance.

To conclude, while RoBERTa is designed as an improved version of BERT, its modifications do not always lead to better performance in NER tasks. It is more computationally demanding yet does not always leverage its increased training data and longer context effectively for entity recognition.

4.5 Meaning of Capital Letters. Comparison of BERT-cased and BERT-uncased Results

At the first glance, the task of Named Entity Recognition seems to consist in detecting proper names. This is not true in all cases, because there are entities that have a defined type and are not proper names, for example, the date in the Kaggle dataset is of type "tim". Nevertheless, many named entities begin with capital letters, for example, the first name and surname of a person, or the name of an organization. Thus, we investigate whether the BERT-cased model, pre-trained on a dataset containing uppercase and lowercase letters, produces embedding vectors that outperform those produced by BERT-uncased, a model that has been pre-trained on an all-lowercase dataset.

The test results for BERT-cased and BERT-uncased are presented in Tables 6 and 7. The effects of these models on the Kaggle and CoNLL-2003 datasets were examined in the same way as the effects of the models in the previous sections.

The results of research on both datasets show that the BERT-cased model creates embedding vectors that work better in the task of detecting named entities. The only case where the BERT-uncased model performed better is CNN's BERT for the CoNLL-2003 dataset, where the micro precision is 0.78 pp. higher than that achieved by BERT-cased. For the Kaggle dataset, the largest difference in F1 micro scores is 4.02 pp. while the biggest difference in F1 macro results is 4.50 pp. The analogous values for the CoNLL-2003 dataset are 3.65 pp. and 2.7 pp., respectively. The differences in the results are significant, considering that the only difference is the size of the letters. The BERT-cased model produces embedding vectors that work better in the NER task.

5 Conclusions

In this paper, the influence of the choice of a model creating word embedding vectors on the results of the Named Entity Recognition, one of the widely used Natural Language Processing tasks, was investigated.

Five models creating word embeddings were used for the research: non-contextual GloVe, contextual ELMo, BERT, and RoBERTa, which create word representations, and contextual LUKE, which creates entity representations.

Based on the experiments, it can be concluded that in the case of Named Entity Recognition in single sentences (short context), the ELMo's embeddings perform the best. GloVe achieves slightly worse results. The context of the word is significantly less important when we are limited to just a single sentence, so using GloVe's word embedding vectors seems like a good idea. The BERT model also performs well, while the RoBERTa model ranks in the last position. However, in the NER task with the long context, the best performer is LUKE, which achieved much higher results than other models. Among the models creating representations of words, the best results were achieved by ELMo, and the worst by GloVe. This model creates non-contextual embedding vectors, which is why it does not perform well with long texts, i.e., long context. BERT again performs better than RoBERTa but worse than ELMo.

Another finding is that, BERT and especially RoBERTa perform significantly worse than other models for entity types with small number of instances for both short and long context. GloVe yields the best results in such a case for short context, especially with LSTM.

Finally, the effect of keeping/removing upper letters was investigated in the Named Entity Recognition task. Two types of BERT models were tested: BERT-cased, which was pre-trained on a dataset consisting of uppercase and lowercase letters, and BERT-uncased, which was pre-trained on a dataset consisting of only lowercase letters. As it turned out, such a small change in the dataset led to significantly better results of the BERT-cased model.

An interesting extension of this paper seems to be the study of the influence of the method of creating embedding vectors in other Natural Language Processing tasks, as well as the study of other models. Another possible continuation is to use the entity and words representations created by LUKE and examine their influence on the results of the experiments. For this purpose, it would be necessary to implement the LUKE model, because the embedding vectors created by LUKE are not available.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5359–5368. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022. A survey on deep learning for named entity recognition. *IEEE Trans. Knowl. Data Eng.*, 34(1):50–70.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019. A comparison of word-embeddings in emotion detection from text using bilstm, CNN and self-attention. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, UMAP 2019, Larnaca, Cyprus, June 09-12, 2019*, pages 63–68. ACM.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

A. Walia. Annotated corpus for named entity recognition. *https://www.kaggle.com/datasets/abhinavwalia95/entity-annotated-corpus*.

Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul R. Kingsbury, and Hongfang Liu. 2018. A comparison of word embeddings for the biomedical natural language processing. *J. Biomed. Informatics*, 87:12–20.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6442–6454. Association for Computational Linguistics.