Named Entity Recognition

Jakob Božič
University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113, SI-1000 Ljubljana
jakob.bozic@gmail.com

Abstract

Named entity recognition (NER) is used to extract the named entities from the text into the pre-defined categories. Most commonly these categories are at least names, locations, organizations and other, however in various branches of NER different, branch-specific categories are added. State-of-the-art (SOTA) results are, similarly as in other fields of Machine Learning, given by deep (recurrent) neural networks, which have in last years completed the transition from hand-crafted to deep features. In our work we first give a brief overview of recent development of the field with more focus on Slovene language and then use of the current SOTA (RNN) approaches on ssj500k (Krek et al., 2019) and SentiCoref (Žitnik, 2019) datasets. We also investigate how well the knowledge is transferred between the both datasets

1 Related work

In our work we focus on more recent, deep learning based approaches, using Recurrent Neural Networks (RNN). A brief overview of history of NER is given in (Yadav and Bethard, 2019), which also contains results of different NER approaches for four different languages and shows that deep learning approaches outperform traditional methods across the board. Extensive study of NER for slovene language has been presented in (Štajner et al., 2013). Authors have used supervised learning and Conditional Random Fields to propose then state-of-the-art system. They evaluated how introduction of lexicons, part-of-speech tags and conjuctions of neighbouring properties effect the system's capability to correctly classify named entities. In their work they showed that using part-of-speech tags can further improve model's capabilites. When enabling all of their improvements, their model achieves 74% precision and 72% recall, however this drops to 63% precision and 59% recall, when using basic model with no part-of-speech tags and no lexicons.

Deep Recurrent neural networks have played the most significant role in recent advances in natural language processing. One of the key ingredients which has been around for quite a while is Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Only recently a similar, but simpler gating mechanism called Gated recurrent unit (GRU) (Cho et al., 2014) has started gaining popularity.

ELMO embeddings (Peters et al., 2018) are deep contextualized word representations which model word usage and how the usage varies in different contexts. They are easily adaptable to existing NLP solutions and usually introduce noticeable performance improvements.

Currently, SOTA results on many NLP tasks are achieved by approaches that incorporate Bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018).

2 Data

We train and evaluate our approach on both ssj500k and SentiCoref datasets.

2.1 ssj500k

ssj500k (Krek et al., 2019) contains approximately 500.000 words, manually annotated on different levels. The data comes from FidaPLUS corpus, which contains very diverse set of text from various daily newspapers, books, parliament speeches, adverts, etc. Roughly half of the dataset is also annotated with named entities, on which we focus our attention. There are in total 4.398 named entities in total, separated in five classes: loc(ation), org(anization), per(son), misc(ellaneous) and der(ivative)-per(son), however we decided to merge person and derivative-

person in per, since derivative-person contained very few samples. Due to only half of the corpus being labeled for named entities we only used those sentences with at least one named entity present, since we could not otherwise tell whether the sentence was labeled for named entities or not. In order to avoid or at least minimise the problems associated with class-imbalance (Buda et al., 2017) we used over-sampling for all categories. In particular, we first create one collection of sentences for each tag, a sentence is put in a collection if it contains at least one named entity for that tag. The last 20% of each collection is put in the test set. For each tag, we then over-sample the remaining part, by repeating each sentence as many times, as the quotient of the length of the collection for that and the length of collection for most common tag. This ensures that all tags are represented approximately equally. We could presumably further improve the data by introducing sentences without any named entities, however we were not able to determine which sentences were labeled and which not.

2.2 SentiCoref

SentiCoref (Žitnik, 2019) dataset contains 31.419 named entities, separated in three categories, there are 15.285 org(naizations), 8.606 per(sons) and 7.528 loc(ations). The data consists of news articles on various Slovenian online news portals. Unlikely in the ssj500k dataset, there is no misc(ellaneous) category. Again, we used over-sampling to avoid or minimise class-imbalance related drawbacks, the process is described above. We again used 80% of data for training and 20% for testing. The dataset is split into documents and not into sentences, which meant that we had to split it ourselves. Since we could not come up with efficient way to take advantage of existing text segmentation tools, we decided for very simple and not very good split. We removed all non alphanumeric characters except for dot, question mark and exclamation mark, which we used as splitting points. We then discarded sentences which were only single word, since those were mainly there because of the wrong split. While this is definitely not a optimal sentence splitting approach, we were still able to achieve very good results on both datasets, which shows that our method is indifferent to a minor number of wrongly split sentences.

3 Methodology

Our model consists of three main parts, (i) an embedder, (ii) an encoder, and (iii) a projector. Embedder transforms characters and words to embeddings and then encoder transforms them in representation which projector uses to make final predictions. Different embedders and different encoders are used and compared against each other. We use either pretrained ELMO embeddings (Ulčar and Robnik-Šikonja, 2019) or we train an embedder with embedding dimension of 64. For encoder we use either two layer LSTM (Hochreiter and Schmidhuber, 1997) or two layer GRU (Cho et al., 2014) network, with hidden dimension of 64, both in standard or bi-directional configuration. We tried using greater dimensions, however we observed either overfitting or the training was simply too long to perform all the experiments we wanted to. Projector was the only constant component, a simple fully-connected layer which maps outputs from the encoder in final tags.

Together we have three variable parts, for eight experiments on each dataset. In order to quantify, how well the knowledge is transferred from one dataset to the other, we evaluated the best performing model from each dataset on the other. Some minor adaptations were necessary however, since the datasets do not have the same labels.

4 Results

We extensively evaluated previously described variations of main components of our model. Results for both datasets are given in Table 1. The most noticeable conclusion we can draw is that quality of word representations plays a key role in the model's performance, since the performance boosts introduced by ELMO are overwhelming. Besides for the overall better word representations, one additional explanation why ELMO brings such significant performance boost is, that it effectively introduces much more training data, since it was pretrained on a very large corpus. We did not observe significant effects on performance when using either LSTM or GRUs, regardless whether we use them in bi-directional configuration or not. We report the average F-measure, which is calculated as simple mean of per-category F-measures, not the weighted average. We decided for that, since we believe that normal average better represents the model's ca-

Dataset	Embedder	Encoder	Bi-directional	Avg. F-measure	Per F-measure	Loc F-measure	Org- F-measure	Misc F-measure
ssj500k	Classical	LSTM	✓	48.30	58.02	60.14	52.57	22.48
	Classical	LSTM		51.26	60.67	60.91	54.38	29.07
	Classical	GRU	✓	49.37	55.40	63.49	56.70	21.90
	Classical	GRU		49.22	61.74	58.84	49.15	27.16
	ELMO	LSTM	✓	86.43	95.63	93.09	83.40	73.62
	ELMO	LSTM		85.74	96.02	92.14	82.66	72.15
	ELMO	GRU	✓	**87.74**	96.02	93.14	85.65	76.14
	ELMO	GRU		86.37	95.78	91.60	84.59	73.52
SentiCoref	Classical	LSTM	✓	75.11	81.15	66.16	78.91	/
	Classical	LSTM		76.07	83.72	66.59	77.89	/
	Classical	GRU	✓	75.56	82.26	66.20	78.22	/
	Classical	GRU		75.30	83.24	65.00	77.46	/
	ELMO	LSTM	✓	90.35	97.41	84.10	89.53	/
	ELMO	LSTM		90.53	97.21	84.28	90.11	/
	ELMO	GRU	✓	90.20	97.06	84.04	89.48	/
	ELMO	GRU		**90.59**	97.30	84.76	89.70	/

Table 1: Evaluation of different combinations of embedders and encoders on ssj500k and SentiCoref datasets. Highlighted in bold are the best average results on both datasets.

pabilities, instead of the properties of the data on which it was evaluated.

Hyperparameters In all of the experiments we used identical hyperparameters, we trained the model for 50 epochs, with early stopping if there were no improvements for 15 epochs, using Adam (Kingma and Ba, 2015) optimizer, with learning rate of 0.001, and weight decay 10^{-4}. Batch size was 4, since we were limited by the memory of our GPU. Both LSTM and GRUs had 2 layers, with hidden dimension of 64. When we used the classical embedder, the embedding dimension was also 64.

ssj500k comparison On the ssj500k dataset, our model performs significantly better as the one from (Štajner et al., 2013), however since the authors did not report how the train and test data was formed and also how they calculated the final average F1-measure, comparison may not be completely accurate. Their best reported model achieved 72% average F1 measure, whereas ours achieves 88% average F1 measure, which represents more than two-fold reduction of error. It is also worth noting, that our approach did not use any additional information, such as part-of-speech tags or dictionaries. In the same configuration, their average F1 measure was 61%.

4.1 Knowledge transfer

In order to evaluate, how general and data independent our approach is, we cross-evaluated the best models from each dataset on the other dataset. In particular, we evaluated the model which used ELMO as embedder with bi-directional GRUs and was trained on ssj500k dataset (7th row in Table 1), on the SentiCoref dataset, and the model which used ELMO as embedder with classical GRUs and was trained on ssj500k dataset (last row in Table 1) on the SentiCoref dataset. Since SentiCoref does not have misc(ellaneous) label, we marked all the words which were labeled as misc(ellaneous) in ssj500k dataset as other (not a named entity). For both datasets, the evaluation was performed on full dataset, not on the subset that was used to evaluate models when training on data from the same dataset.

Dataset	Avg. F-measure	Per	Org	Loc
ssj500k ->SentiCoref	86.10 (64.57)	96.68	84.52	77.09
SentiCoref ->ssj500k	82.78	93.51	73.10	81.74

Table 2: Results of cross-evaluation of best-performing models. When evaluating model that was trained on ssj500k on SentiCoref (first row), the model predicts some words as misc(ellaneous), however those are (at least considered) all wrong, since there is not misc(ellaneous) category in SentiCoref dataset. This means that the model gets 0.00 F-score for that label, and the value in parenthesis in the first row represents the average F-measure we get if we also consider the absent category when calculating average.

The results of cross-dataset evaluation are given in Table 4.1. We can see that the performance of both models drops, which was expected, since the datasets contain data from different kinds of text. Rather unexpectedly, the model trained on ssj500k dataset works better on SentiCoref, than the model trained on SentiCoref works on ssj500k.

We would expect that to not be the case, since the SentiCoref datasets is bigger. One possible explanation is, that the more diversity in texts in ssj500k translates in better overall performance and better knowledge transfer. SentiCoref contains only data from online news portals, which may not translate very well to e.g. literary works contained in ssj500k.

Example In Table 4.1 we can see how the best models from both datasets perform on a piece of news article from RTVSlo. While the example is not very hard, we can see that both models do not make a single mistake.

5 Discussion

We evaluated various combinations of embedders and encoders for a NER model and demonstrated that using an embedder pretrained on larger corpus brings significant performance improvements. Our best model trained on ssj500k dataset achieves more than two-fold reduction in error compared to the one from (Štajner et al., 2013). We also extensively evaluated our models on SentiCoref dataset, and demonstrated that findings from ssj500k dataset are not dataset-specific. We cross-evaluated some of our models, and shown that they are capable of performing well on the kind of data they did not necessarily see during the training. In future, we plan to incorporate BERT in our model and search for potentially better hyperparameter values.

References

Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. 2017. A systematic study of the class imbalance problem in convolutional neural networks. *CoRR*, abs/1710.05381.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Simon Krek, Kaja Dobrovoljc, Tomaž Erjavec, Sara Može, Nina Ledinek, Nanika Holz, Katja Zupan, Polona Gantar, Taja Kuzman, Jaka Čibej, Špela Arhar Holdt, Teja Kavčič, Iza Škrjanec, Dafne Marko, Lucija Jezeršek, and Anja Zajc. 2019. Training corpus ssj500k 2.2. Slovenian language resource repository CLARIN.SI.

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

Matej Ulčar and Marko Robnik-Šikonja. 2019. High quality elmo embeddings for seven less-resourced languages.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models.

Slavko Žitnik. 2019. Slovene corpus for aspect-based sentiment analysis - SentiCoref 1.0. Slovenian language resource repository CLARIN.SI.

Tadej Štajner, Tomaž Erjavec, and Simon Krek. 2013. Named entity recognition in slovene text. *Slovenščina 2.0: empirical, applied and interdisciplinary research*, 1(2):58–81.

Model	Po	različnih	delih	Slovenije	zaznavamo	od	10	do	30	odstotkov
ssj500k	-	-	-	loc	-	-	-	-	-	-
SentiCoref	-	-	-	loc	-	-	-	-	-	-
Model	manj	rakavih	diagnoz	Kako	bo	to	vplivalo	na	dolgi	rok
ssj500k	-	-	-	-	-	-	-	-	-	-
SentiCoref	-	-	-	-	-	-	-	-	-	-
Model	ta	trenutek	še	ne	vemo	je	za	Televizijo	Slovenija	povedala
ssj500k	-	-	-	-	-	-	-	org	org	-
SentiCoref	-	-	-	-	-	-	-	org	org	-
Model	Vesna	Zadnjik	vodja	epidemiologije	in	registra	raka	na	Onkološkem	inštitutu
ssj500k	per	per	-	-	-	-	-	-	org	org
SentiCoref	per	per	-	-	-	-	-	-	org	org

Table 3: Example of evaluation of best models from both datasets on a piece of news article which was not included in either training set.

Named Entity Recognition

NLP - Final defense

Jure Bevc, Anže Dežman, Robert Modic

Abstract

In this report we worked on solving named entity recognition (NER) problem in Slovenian language. For this purpose we built solutions based on existing two models: *Stan-ford Named Entity Recognizer model* and *DeepPavlov BERT based model*. The models were trained using the ssj500k dataset and with different training parameters. In the beginning we used only *word* and *NER tag*, to which we later added *morphosyntactic tag*.

1 Introduction

Named entity recognition (NER) is a sub-field of natural language processing, where we seek to process text so that we locate and classify named entities into predefined categories. This has been done through hand crafted grammar-based techniques and through machine learning models. While state-of-the-art solutions for English are reaching near-human performance, we will attempt to implement a NER system for Slovenian language using the ssj500k dataset [5]. To do this, we have researched existing approaches in terms of how they are implemented and evaluated. This report summarizes our findings and approaches in the field of NER.

2 Related work

Since many approaches have been developed to solve the problem of named entity recognition, we started looking at surveys that were done on NER systems. One such survey covers fifteen years of research in the NER field [6], where the authors describe the developed machine learning techniques and also review features and model evaluation.
Another survey was done on deep learning models, which have only recently been studied and developed [9]. In this survey they present architectures for NER and compare them to previous approaches.

From these surveys we found out that most of the popular and recent approaches use some variant of recurrent neural networks (RNN). An efficient and lightweight architecture is presented in the paper [7], where they have drastically reduced the amount of training data, while still achieving close to state-of-the-art results.

Besides neural networks we have also looked at other papers, which describe conditional random fields or Hidden Markov Models as described in [10], where they also achieved great performance. We also reviewed an article published on NER in Slovene [11], where they used conditional random fields and the same ssj500k dataset as we were planning to use.

3 Implementations

For the task of Named entity recognition of the Slovene language we decided to evaluate the *Stanford Named Entity Recognizer model* [4] and the *DeepPavlov BERT based model* [3]. We trained them using the ssj500k [5] dataset.

3.1 Dataset preprocessing

We used the *minidom* [8] Python library to write an *XML-Parser* class, with functions to allow extraction of all words with their named entity sub-type.

In ssj500k dataset the named entities are marked with the <*seg*> tag, sub-type provided in the "sub-type" argument, "PERSON" or "LOCATION" for example. This tag can contain one or more <*w*> tags, denoting words, that are part of a named entity. However, not every named entity in the dataset was marked with the <*seg*> tag. We only used the part of the dataset that had the named entities marked.

Each <*w*> node tag also had a morphosyntactic tag written in its *ana* attribute. When parsing these, every hyphen inside its value was removed, to not cause problems during learning.

We wrote two named entity extraction methods, one for the *Stanford model* and one for *DeepPavlov BERT based model*. They both return the same elements, just in different format.

3.2 DeepPavlov BERT based model

For the base BERT model we used the *BERT-base, multilingual, cased, 12-layer, 768-hidden, 12-heads, 180M parameters* model provided by DeepPavlov[1]. The configuration we used is the default *bert_config.json* configuration, that comes with the base model. It is written bellow:


```
{
  "attention_probs_dropout_prob": 0.1,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pooler_fc_size": 768,
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2,
  "vocab_size": 119547
}
```


The *ssj500k* dataset was first preprocessed with our parser described in section 3.1, output of new-line separated sentences was first randomly split into a training set, validation set and test set with the ratios of 0.8, 0.1 and 0.1, as instructed in the documentation[2]. These sets were written to files *test.txt*, *train.txt* and *valid.txt* in CoNLL-2003 [2] like format, as shown below.

```
# Training with NER tags only:
<word> <NER tag>
# Training with NER and morphosyntactic tags:
<word> <morphosyntactic  tag> <NER tag>
```

The base model was then trained on preprocessed data. First it was used as a base for training with *NER* tags only and then for the training with both the *NER* and the *morphosyntactic tags*, producing two trained models. The training for both models lasted about 12 hours, or 5 epochs. Both models were then evaluated on their *test set*, results are shown in section 4. Because of the time difference between model training, their sets were different, so the results may not represent the true state of the models.

[1] http://files.deeppavlov.ai/deeppavlov_data/bert/multi_cased_L-12_H-768_A-12.zip

[2] http://docs.deeppavlov.ai/en/master/features/models/ner.html#training–data

3.3 Stanford model

The testing and training datasets that were used for the DeepPavlov BERT model were also used to evaluate the Stanford Named Named Entity Recognizer model [4]. This model is based on a Conditional Random Field (CRF) sequence model and it comes with many feature extractors for Named Entity Recognition, which can be configured through the NER properties file.

All available properties are documented in the NERFeatureFactory class [1] and since these can heavily impact the model performance, we present our properties in the following section:

```
useClassFeature=true
useWord=true
useNGrams=true
noMidNGrams=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
maxLeft=1
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
wordShape=chris2useLC
useDisjunctive=true
```

4 Results and discussion

4.1 BERT models' results

Tables 1 and 2 show the results of the DeepPavlov BERT models, the former trained only with *NER* tags and the latter trained with both *morphosyntactic tags* and *NER* tags.

NER tag	precision	recall	F1
LOC	0.90	0.92	0.91
MISC	0.56	0.50	0.52
ORG	0.79	0.76	0.77
PERSON	0.87	0.93	0.90
TOTAL	0.85	0.86	0.85

Table 1: DeepPavlov BERT NER tags only results.

NER tag	precision	recall	F1
LOC	0.84	0.90	0.87
MISC	0.62	0.50	0.55
ORG	0.64	0.68	0.66
PERSON	0.90	0.90	0.91
TOTAL	0.80	0.82	0.81

Table 2: DeepPavlov BERT NER and morphosyntactic tags results.

4.2 Stanford models' results

NER tag	precision	recall	F1
LOC	0.84	0.81	0.83
MISC	0.60	0.32	0.42
ORG	0.76	0.57	0.65
PERSON	0.76	0.88	0.82
TOTAL	0.77	0.73	0.75

Table 3: Results of the Stanford model trained on ssj500k dataset using only NER tags.

NER tag	precision	recall	F1
LOC	0.80	0.69	0.74
MISC	0.72	0.17	0.28
ORG	0.70	0.44	0.54
PERSON	0.73	0.84	0.78
TOTAL	0.74	0.63	0.68

Table 4: Results of the Stanford model trained on ssj500k dataset using morphosyntactic tags.

4.3 Total values of all models

model	precision	recall	F1
Štajner	0.63	0.59	0.61
Štajner + morph. tag	0.72	0.68	0.70
BERT	0.85	0.86	**0.85**
BERT + morph. tag	0.80	0.82	0.81
Stanford	0.77	0.73	0.75
Stanford + morph. tag	0.74	0.63	0.68

Table 5: total values for precision, recall and F1 score for all tested models summarised

4.4 Discussion

Both DeepPavlov BERT based models seem to be very good at distinguishing locations and persons. Low performance for MISC subtype can be attributed to low number of such samples in the ssj500k dataset. Overall the performance of the BERT model trained only on *NER* tags seems better than the one trained with *NER* and *morphosyntactic tags*, but as already stated in section 3.2 the evaluation sets were different, because of the time interval between model training, so the results may not be comparable.

We can observe that although the Stanford model achieved decent results in some categories, it generally performed worse than the DeepPavlov model. This model is closer to the results of the Štajner et. al. model with morphosyntactic tags. With added morphosyntactic tags the performance of the Stanford model seems to worsen, just like our BERT model.

We believe the reason for worse results in our models added morphosyntactic tags are due to added variability of possible true selections and the fact that there isn't enough data in the dataset to train on that variability.

5 Conclusion

As already stated in 3.1, the used ssk500j[5] dataset wasn't completely segmented, so we only had to use a portion of it. The dispersion of the named entity types was also very far from uniform, and of those, only "PERSON", "LOCATION" and "ORGANIZATION" major named entity types were present. If the rest of the dataset were to be completely segmented in the future, the models might be able to learn better how to distinguish certain named entities, perhaps other major entity types would be present in that portion. The portion of the dataset we didn't use, could also be used as an additional test set.

References

[1] Class nerfeaturefactory documentation.

[2] Language-independent named entity recognition (ii).

[3] M. Burtsev, A. Seliverstov, R. Airapetyan, M. Arkhipov, D. Baymurzina, N. Bushkov, O. Gureenkova, T. Khakhulin, Y. Kuratov, D. Kuznetsov, A. Litinsky, V. Logacheva, A. Lymar, V. Malykh, M. Petrov, V. Polulyakh, L. Pugachev, A. Sorokin, M. Vikhreva, and M. Zaynutdinov. Deeppavlov: Open-source library for dialogue systems. 07 2018.

[4] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[5] S. Krek, K. Dobrovoljc, T. Erjavec, S. Može, N. Ledinek, N. Holz, K. Zupan, P. Gantar, T. Kuzman, J. Čibej, Š. Arhar Holdt, T. Kavčič, I. Škrjanec, D. Marko, L. Jezeršek, and A. Zajc. Training corpus ssj500k 2.2, 2019. Slovenian language resource repository CLARIN.SI.

[6] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[7] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.

[8] xml.dom.minidom. 20.7. xml.dom.minidom — minimal dom implementation — python 3.6.10 documentation.

[9] V. Yadav and S. Bethard. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*, 2019.

[10] G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 473–480, USA, 2002. Association for Computational Linguistics.

[11] T. Štajner, T. Erjavec, and S. Krek. Named entity recognition in slovene text. *Slovenščina 2.0: empirical, applied and interdisciplinary research*, 1(2):58–81, Dec. 2013.

Razpoznavanje imenskih entitet na podatkovni množici ssj500k

Jakob Makovac
UL FRI, Večna pot 113
jm6531@student.uni-lj.si

Abstract

This document contains the formatting requirements for TACL final versions. These formatting rules take effect for all final versions received from September 2, 2018 onwards.

1 Uvod

Prepoznavanje imenskih entitet je pomembno področje obdelave naravnega jezika. Cilj je pridobivanje informacij iz teksta tako, da prepoznamo in ustrezno označimo različne imenske entitete, na primer: osebe, lokacije, organizacije ipd.

Za ocenjevanje rezultatov se navadno uporabljajo natančnost, priklic in ocena F.

Trenutno so najboljši rezultati doseženi s pomočjo modelov, ki se poslužujejo nevronskih mrež. Ocena F, ki jo dosegajo taki algoritmi so okrog 90% za angleščino in nekaj manj, okrog 85% za težje jezike, na primer španščino, nizozemščino (Yadav and Bethard, 2019).

Namen našega dela je, da najprej poskušamo implementirati razpoznavanje imenskih entitet s pomočjo pogojnih naključnih polj, nato dodati oziroma spremeniti značilke za izboljšanje rezultata, nato pa vključiti dodatne postopke za pridobivanje značilk, verjetno s pomočjo nevronskih mrež.

2 Sorodna dela

Najprej smo poskušali ponoviti rezultate članka Štajnerja (Štajner et al., 2013). V članku so uporabili podatke iz označenega korpusa ssj500k. Implementirali so postopek nadzorovanega učenja z algoritmom pogojnih naključnih polj. Najboljši rezultati, ki so jih dosegli dosegajo 74% natančnost in 72% priklic. Rezultati so se precej razlikovali po razredih entitet. Pri osebnih imenih na primer je bila končna ocena F 88%, pri stvarnih na primer pa samo 33%. Rezultate so poskušali izboljšati z vključevanjem dodatnih podatkov. Uporabili so na primer tako oblikoskladenjske oznake, kot tudi leksikone.

Model pogojnih naključnih polj se pogosto nadgrajuje z dodatnimi algoritmi, ki izboljšajo njegovo učinkvitost. Model LSTM-CRF na primer, izboljša napovedi modela CRF za okoli 5%, predvsem z izboljšanjem priklica (Habibi et al., 2017).

3 Podatki

Pri svojem delu smo uporabili slovenski korpus ssj500k, ki vsebuje okoli 500 tisoč besed. Te so ročno označene na nivoju tokenizacije, stavčne segmentacije, morfoloških oznak in lematizacije. Od tega je polovica besed označenih tudi na nivoju imenskih entitet, kar je pomembno za naše delo (Krek et al., 2018).

4 Metode

4.1 Pogojna naključna polja

Model pogojnih naključnih polj (conditional random fields ali krajše CRF) se pogosto uporablja pri procesiranju naravnega jezika ali določenih aplikacijah računalniškega vida. Uporaben je pri modeliranju sekvenčnih podatkov. Če stavek predstavimo kot zaporedje besed, je vsaka beseda označena z najbolj verjetnim stanjem. Označevanje je podobno Markovskim modelom odvisno le od lastnosti trenutne in predhodne besede.

Model CRF si lahko predstavljamo kot graf v katerem stanja predstavlja množica razredov entitet. Definiramo lahko dve naključni spremenljivki X in Y, kjer je X naključna premenljivka prek zaporedja podatkov, ki jih želimo označiti, Y pa spremenljivka preko razredov entit. (X, Y) je torej naključno polje, če ima Y markovsko lastnost glede na graf: $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$, kjer $w \sim v$ predstavlja da sta w in v soseda v grafu

(Lafferty et al., 2001).

5 Rezultati

6 Diskusija

7 Zaključek

References

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Simon Krek, Kaja Dobrovoljc, Tomaž Erjavec, Sara Može, Nina Ledinek, Nanika Holz, Katja Zupan, Polona Gantar, Taja Kuzman, Jaka Čibej, Špela Arhar Holdt, Teja Kavčič, Iza Škrjanec, Dafne Marko, Lucija Jezeršek, and Anja Zajc. 2018. Training corpus ssj500k 2.1. Slovenian language resource repository CLARIN.SI.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Vikas Yadav and Steven Bethard. 2019. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. *arXiv e-prints*, page arXiv:1910.11470.

Tadej Štajner, Tomaž Erjavec, and Simon Krek. 2013. Named entity recognition in slovene text. *Slovenščina 2.0: empirical, applied and interdisciplinary research*, 1(2):58–81.

Named entity recognition

Jan Jenko, Domen Požrl
University of Ljubljana
Faculty for computer and information science
Večna pot 113, SI-1000 Ljubljana
jj0549@student.uni-lj.si, *dp2576@student.uni-lj.si*

Abstract

We explore a supervised learning approach at classifying named entities, a widely studied problem in language processing. We followed (Štajner et al., 2013) and recreated their experiments using the CRF model. We went a step further in investigating the importance of different groups of attributes. Finally, we applied a neural network based BERT model to the same problem and found that it performs better, depending on the context of our problem.

1 Introduction

Named entities are proper nouns, words such as personal and geographic names, organisation and institution names, titles, etc. Named entity recognition plays an important role in text analysis, since named entities convey more information than simply the words that comprise them - they refer to fixed entities that remain constant throughout the text, and often even in between different texts. As such, they are an excellent means of linking texts. For example, the name of a public figure can be used to link together all newspaper articles in which the person in question is mentioned, helping us in constructing a narrative about that person through time. With this purpose in mind, there now exist indexes of named entities.

2 Related work

(Štajner et al., 2013) already worked on this exact problem which is why for now we implement their solution with minor changes in attribute construction. Their approach uses Conditional Random Fields for supervised learning on a set of attributes that are acquired from every word in the dataset. The first group of attributes comes from the patterns of the letters in the word. They check if there are any numbers in the word or if there are any capital letters, any punctuation and so on. The second group of attributes is a dictionary of known words related to some of the proper nouns we are trying to identify. Such words include but are not limited to world countries, capital cities of every country, all Slovenian cities, names of the days in the week, names of the months and so on. The third group are all morphosyntactic properties that are available for each word in the dataset. The fourth and final group of attributes is one attribute that describes the length of the word and another that is a logical conjunction of the attributes of the neighbouring words.

3 Data

Current systems for named entity recognition are trained using supervised learning, for which large corpora of labeled texts must exist. We will use the SSJ500k corpus (Krek et al., 2019) to train our models. It is important to note, that only a part of the corpus is tagged for named entities and so, we will only use this part of the corpus. In Table 1 we can see, alongside simple statistics, that the corpus is imbalanced in regards to number of proper nouns versus other words, the last of which represent about 98% of the entire corpus. This could potentially present a difficulty which one of the approaches looks to resolve by balancing the dataset.

The version of the SSJ500k corpus available to us had 5 classes of named entities: personal, derived personal, organizational, location and miscellaneous. However, the older version that (Štajner et al., 2013) used did not include the derived personal named entity. These are similar to personal named entities, but signify possession. To be able to compare our results to results in (Štajner et al., 2013), we decided to merge the personal and derived personal classes.

	#
Sentences	7742
at least 1 named entity	1846
exactly 1 named entity	1294
exactly 2 named entities	346
3 or more named entities	206
Words	147165
Other words	144337
Proper nouns	2828
Person proper nouns	1295
Location proper nouns	972
Organisation proper nouns	334
Person derived proper nouns	128
Miscellaneous proper nouns	99

Table 1: This table shows a few simple statistics about the SSJ500k corpus.

4 Attributes construction

As described in the Related works section, we used a very similar method of attribute construction. The first group of the attributes, as seen in Table 2, tells us whether the word has any of the stated properties.

The first group of attributes
First letter capital
Only capital letters
Capital letters inside of the word
Numbers in the word
Only numbers in the word
Numbers and mathematical operations in the word
Only numbers and letters in the word (alphanumeric)
Roman numerals
Includes hyphen or dash
Only capital letters seperated by a dot
Single capital letter followed by a dot
Single letter
Single capital letter
Punctuation
Quotation marks
Only lowercase letters

Table 2: This table shows the attributes of the first group

The second group of attributes tells us whether the word is present in lists of words from Table 3.

The third group of attributes is the morphosyntactic properties available in the corpus. All of the

The second group of attributes
Slovene female names
Slovene male names
Slovene surenames
Slovene area names
Slovene cities
Capital cities of every country in the world
Every country in the world
Slovene names of days in the week
Slovene month names

Table 3: This table shows the attributes of the second group

described groups of attributes so far are very similar to those in (Štajner et al., 2013). The last group of attributes is where we introduce a different approach. We still take into account the length of the word but instead of logically combining the neighbouring words we use the entire set of attributes of the neighbouring words. That means that the final attributes vector for an ith word are all of the attributes vectors of words from $i-2$ to $i+2$. This way we give more importance to bigger parts of sentences and not just a single word.

5 Methods

In the first part, we used the CRF model to try reproduce the methods and results from (Štajner et al., 2013). We then further investigated the importance of groups of attributes by running multiple experiments. First, we used all attributes, as described in section 4. Notice that the attributes do not only contain properties of the target word (the word we are trying to classify), but also of its neighbourhood, that is of words that are at most two places away from the target word. In the second experiment we strip away the attributes with properties of words that are two places away from the target word, and in the third experiment we also strip away attributes with properties of words that are one place away from the target word. Surprisingly, we got the best results in the last experiment so we used the stripped data (attributes with properties of the target word only) in the subsequent experiments with the CRF model. Next, we eliminated a single attribute that carries the morphosyntactic properties of the word. The rationale here is that the data is unlikely to contain this property and also not contain the named entity classification we are after. This had a significant neg-

ative effect on the classification accuracy. In the last CRF experiment, we tried to somewhat balance the classes. The dataset is highly imbalanced, with vast majority of words not being a named entity. To counter this imbalance, we removed from our training data the sentences that did not contain any named entities.

For each experiment we first tuned the hyper-parameters using a grid search with data split into train (70%) and test (30%) subsets. We then used the models with optimal hyper-parameters and evaluated them using a five fold cross validation.

In the second part, we tested a BERT model (Devlin et al., 2018) pretrained for slovene, croatian and english (Ulčar and Robnik-Šikonja; Žitnik, 2020) on a modified version of the constructed attributes. We used individual words and their morphosyntactic properties to update the whole model weights for named entity recognition with 3 epochs. The data was split into train (80%) and test (20%) subsets and then the test data was split in half for validation. Later the performance was evaluated in terms of precision, recall and the F1-score.

6 Results

Table 4 shows the best results achieved in (Štajner et al., 2013). The combined figure is the weighted average score, where weights are the supports of each class.

Depending on the application, this table can be considered incomplete, as it does not show the classification accuracy for words that are not a named entity. We believe this is an important aspect of a classifier, so we will also include this figure.

	Precision	Recall	F1	#
per	0.85	0.91	0.88	1922
org	0.63	0.58	0.60	785
misc	0.39	0.30	0.33	406
loc	0.82	0.80	0.81	1284
combined	0.74	0.72	**0.73**	4397

Table 4: Best results achieved in (Štajner et al., 2013) using the CRF model.

6.1 CRF model

The results of the first experiment where we used original data is shown in table 5. The overall performance is comparable to results in table 4, but there are significant differences in individual classes. Organizational named entities and miscellaneous classes have significantly worse results in our experiment, but are also very underrepresented. This might contribute to the poor performance (as the model didn't have many significant learning samples), but it also means the overall score is not severely impacted.

	Precision	Recall	F1	#
per	0.79	0.82	0.81	1423
org	0.43	0.38	0.39	334
misc	0.20	0.09	0.12	99
loc	0.84	0.82	0.84	972
combined	0.74	0.74	**0.75**	2828
not-propn	0.998	0.999	0.999	144337
combined	0.993	0.994	**0.994**	147165

Table 5: Classification accuracy using original data.

As seen in table 6, the classification accuracy did not change much in the second experiment, where we did not consider properties of words that were two places away from the target word.

	Precision	Recall	F1	#
per	0.79	0.84	0.82	1423
org	0.47	0.35	0.39	334
misc	0.19	0.09	0.12	99
loc	0.84	0.83	0.84	972
combined	0.75	0.75	**0.75**	2828
not-propn	0.998	0.999	0.999	144337
combined	0.993	0.999	**0.994**	147165

Table 6: Classification accuracy of the CRF model working on data containing properties of the target word and its direct neighbors.

We then went further and disregarded properties of all words except the target word. The performance improved marginally, as seen in table 7

Next, we removed the attribute containing the morphosyntactic properties of the words. The performance worsened significantly, as seen in table 8.

Finally, we attempted to improve the performance by balancing the training data. As the model performs on whole sentences it was impossible to balance the classes completely (nor would we want to, given how few words are actually a named entity). We removed the sentences with no named entities and cut the original 7742 sentences

	Precision	Recall	F1	#
per	0.78	0.90	0.84	1423
org	0.76	0.27	0.40	334
misc	0	0	0	99
loc	0.87	0.84	0.85	972
combined	0.78	0.79	**0.76**	2828
not-propn	0.998	0.999	0.999	144337
combined	0.994	0.995	**0.994**	147165

Table 7: Classification accuracy of the CRF model working on data containing only properties of the target word.

	Precision	Recall	F1	#
per	0.81	0.40	0.46	1423
org	0	0.01	0	334
misc	0	0	0	99
loc	0.98	0.56	0.71	972
combined	0.75	0.39	**0.48**	2828
not-propn	0.989	0.996	0.992	144337
combined	0.984	0.984	**0.983**	147165

Table 8: Classification accuracy of the CRF model working with data without the morphosyntactic properties.

down to 1846 significant ones. In the original data, only 2% of all words are named entities. In the balanced training data, this figure is 7%.

As seen in table 9, the balancing did not have a significant impact on the classification performance.

	Precision	Recall	F1	#
per	0.79	0.84	0.82	1423
org	0.47	0.35	0.39	334
misc	0.19	0.09	0.12	99
loc	0.84	0.83	0.84	972
combined	0.75	0.75	**0.75**	2828
not-propn	0.998	0.999	0.999	144337
combined	0.993	0.994	**0.994**	147165

Table 9: Classification accuracy of the CRF model learning on a more balanced dataset.

6.2 BERT model

In Table 10 we can see that the BERT model performs quite differently compared to the CRF model. The individual named entity classes were predicted significantly better, but it also performed significantly worse in predicting words that are not named entities. All CRF models predicted the 'non-propn' class with F1 score of at least 0.99, while BERT only achieved F1 of 0.80. Because the 'non-propn' class carries such weight, the overall performance of the model is worse. Usually this kind of method uses a lot of data (Walia) so perhaps using more data could improve the performance even further, but we only had the current corpus at our disposal. In Figure 1 we can see how validation and training loss change with respect to the number of epochs. It is worth noticing that the validation loss does not decrease drastically therefore there is no reason to think that increasing the number of epochs would improve the performance of the model significantly.

	Precision	Recall	F1	#
per	0.81	0.89	0.85	142
org	0.83	0.71	0.77	28
misc	0.58	0.93	0.72	15
loc	0.98	0.93	0.95	132
combined	0.86	0.88	**0.87**	317
not-propn	0.78	0.81	0.80	317
combined	0.78	0.81	**0.80**	634

Table 10: Classification accuracy of the BERT model.

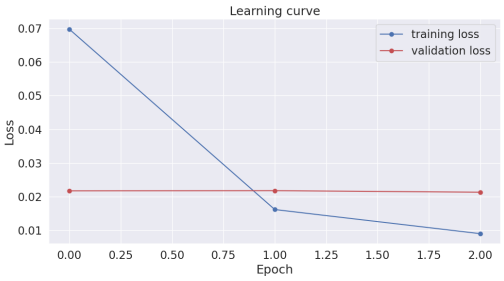

Figure 1: Training loss and validation loss with respect to number of epochs.

7 Conclusion

We managed to reproduce the results from (Štajner et al., 2013) using the CRF model and further analysed the importance of certain groups of attributes. We found that including attributes with properties of neighboring words does not significantly improve the performance, in fact, the model performed better overall if we did not include those attributes. The CRF model relies heavily on the attribute containing the morphosyntactic properties of words. If we ommit this attribute, the overall F1 score (weighted average) accross all four classes of named entities drops from the usual 0.75 to 0.48. Balancing the training dataset does not

improve the result - CRF itself seems to account for data imbalance well.

We then tried to classify named entities based on the original data, but with a neural network based BERT model. The new model performed better in classifying the words that were indeed named entities, but did worse overall since it did not perform well in classifying words that are not named entities as such.

Which model to use depends on the context of the task. If we know that a word is a named entity and want to classify it, BERT performs better. If we need overall performance on all words, we would choose CRF.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Simon Krek, Kaja Dobrovoljc, Tomaž Erjavec, Sara Može, Nina Ledinek, Nanika Holz, Katja Zupan, Polona Gantar, Taja Kuzman, Jaka Čibej, Špela Arhar Holdt, Teja Kavčič, Iza Škrjanec, Dafne Marko, Lucija Jezeršek, and Anja Zajc. 2019. Training corpus ssj500k 2.2. Slovenian language resource repository CLARIN.SI.

Ulčar and Robnik-Šikonja. Pretrained model.

Abhinav Walia. [link].

Slavko Žitnik. 2020. Neural sequence tagging (pytorch).

Tadej Štajner, Tomaž Erjavec, and Simon Krek. 2013. Razpoznavanje imenskih entitet v slovenskem besedilu. *Jezikovne tehnologije*, 2(1):58–81.

Named Entity Recognition

Lodi Dodevksa, Viktor Petreski
University of Ljubljana
Faculty for Computer and Information Science
Večna pot 113, SI-1000 Ljubljana
lodi.dodevska@gmail.com, vpetreski96@gmail.com

Abstract

For this assignment we implemented a couple of Named entity recognition models on two datasets: CoNLL-2003 and GMB. We started with CRF, which was our baseline for future work. Then we continued with Bi-LSTM + CRF. Finally, we managed to implement ELMo embeddings on top of the Bi-LSTM network and fine-tune pre-trained BERT model. We achieved fairly good results with each of them.

1 Introduction

After we did some research on classic and state-of-the-art Named entity recognition models, we chose to implement Conditional Random Fields (CRFs), which will represent the baseline for our work. Next, we decided to continue with a model of Bi-LSTM with a CRF layer. We chose to use rather simple embedding for the data at the beginning, because first we wanted to get familiar with the Bi-LSTM models, explore the influence of the parameters on the performance and only then use some more complicated state-of-the-art embedding. For the final part of the assignment we added ELMo embeddings on top of the Bi-LSTM model and we tried to fine-tune the pre-trained BERT model for our task. We are using CoNLL-2003 and GMB datasets, which are popular for experimenting in NER-related tasks. We did our experiments on both of the obtained datasets at the same time.

2 Related work

Named entity recognition is a very developed branch of natural language processing. It can be split in two groups: generic (tries to localize names of people, organizations, locations etc.) and domain-specific (mostly used in the pharmaceutical, biological or medical research area, for extracting proteins, genes etc.). In this assignment we will focus on the first category. We checked a few papers in order to familiarize with the existing methodologies. The algorithms that are used can be roughly divided in four groups:

- Rule-based approach: follows a set of predefined rules to extract the named entities from a text. The disadvantage of these methods is that the rules must be created manually, based on the domain we are researching (Kim and Woodland, 2000).

- Supervised learning models: such as Hidden Markov Models, Support Vector Machines (Isozaki and Kazawa, 2002), Decision Trees or Conditional Random Fields. An interesting example of CRF is (Finkel et al., 2005) which explains the baseline for Stanford Named Entity Recognizer. These models are trained on an annotated corpus and they obtain fairly good results when used in the right domain, but their drawback is that they are corpus-dependent and there may not be available corpus for each domain we would like to explore.

- Unsupervised learning models: do not depend on an annotated corpus. They aim to find hidden principles or patterns that represent the data appropriately. An example of an unsupervised model is Google's BERT, which was pretrained on a large text corpus from Wikipedia.

- Deep learning approach: most of the successful state-of-the-art NER models are focused on this area. (Yadav and Bethard, 2019) gives a nice overview of deep neural network architectures and highlights some recent improvements.

3 Methods

3.1 Dataset

We are using two datasets for this project because we had some troubles with obtaining the CoNLL-2003 dataset at the beginning. By the point we managed to get CoNLL dataset, we already did some work on GMB and that is why we decided to keep working on both of them.

For initial analysis we chose Groningen Meaning Bank (GMB) dataset, which is a freely available annotated corpus of texts mostly used for named entity recognition POS tagging tasks.

The dataset is given in csv format. We used Pandas DataFrame from python for reading and analyzing it. The dataset contains 25 columns, but we need only *sentence_idx*, *pos*, *word* and *tag* for further usage. One row in the dataset corresponds to one word from the text.

First we had to deal with NaN values, or at least we thought so. Since the dataset is preprocessed to some extent, there was only one erroneous line which contained only NaN values, which was a surprise to us. After we removed the line, there were 1050794 rows left. Then we removed the duplicate sentences. After that there are 768956 rows (sentences) left and 30172 total unique words.

Next, we explored the tag column, which contains the appropriate named entity tag for each token. The following types of entities are covered:

1. *geo* = Geographical Entity
2. *org* = Organization
3. *per* = Person
4. *gpe* = Geopolitical Entity
5. *tim* = Time indicator
6. *art* = Artifact
7. *eve* = Event
8. *nat* = Natural Phenomenon

The entities are tagged using IOB format, which means that there are prefixes I and B before a tag. B indicates that the tag is at the beginning of a chunk, I denotes that the tag is inside of a chunk. If the token does not belong to any of the aforementioned entities, its corresponding tag is O. The total values from each type of entity are shown in Table 1.

Entity	Total
geo	32969
org	27136
per	24991
tim	19517
gpe	11989
art	478
eve	433
nat	205

Table 1: Number of appearances of each entity type

We can see that the tags are not uniformly distributed. We will have to take that into account when we will analyze the performance of the NER model that we will use in the future and identify the number of false positives carefully.

The CoNLL-2003 dataset is the second dataset that we are going to use. It is already completely preprocessed, so there is not much work to do in this part. There is a German and an English version, but we are using only the English one. The dataset contains 4 columns: word, POS tag, chunk tag and NER tag. We are not going to use chunk tag for this task.

Something that is important to mention is that this dataset contains less tags than GMB. There are *per* = Person, *org* = Organization, *loc* = Location and *misc* = Miscellaneous. The tags are annotated in the same way as in the GMB dataset, meaning IOB format is used. The only difference in CoNLL 2003 is that the *PER* tag does not have *B* (before) annotation.

3.2 Data preparation

Before we started with training the models we had to prepare appropriate train and test sets. CoNLL-2003 dataset already contains three subsets: the largest set is the training data and the two smaller sets (*testA* and *testB*) are for testing. On the other hand, the GMB dataset did not come split into subsets for training and testing, so we split the dataset into training and test subset with ratio 70:30. In the Bi-LSTM models, we used 10% of each training set as a validation set.

The input sequences for CRF and Bi-LSTM-CRF are different. CRF uses the words and the corresponding POS tags for creating the input features. Bi-LSTM-CRF uses only the tokens of the sentences. Bi-LSTM-CRF expects a numerical input, so we convert each token and label (tag) of

the vocabulary to a corresponding integer index. The good side of this representation is that it does not require a lot of time to train. Another thing that we must take care of is the length of the sequences. In order to hand over the data as an input to Keras model, all the sequences need to have the same length. So, we found the longest sentence and set the maximum length of the input sequence to be equal to its length. The sentences that are shorter than this are padded to get to the required length. This padding approach is repeated in all of the following models too.

When using ELMo, we actually pass the ELMo embeddings to the Bi-LSTM network we have previously created, but this time without the CRF layer. The input of the ELMo layer consists of padded string sequences, which are transformed into an appropriate embedding.

Finally, when fine-tuning BERT we use BertTokenizer to tokenize the sentences from the datasets and create appropriate signatures/indices for them. Then the input is padded with the previously described procedure. We add a new "PAD" tag here, which will be the assigned label for the padding inputs. The tags in both BERT/ELMO cases are converted to their corresponding numerical indices.

3.3 Models

As mentioned before, we are using a couple of different models for Named entity recognition. The first model (CRF) is supposed to represent the baseline for our work. The second model is Bi-LSTM-CRF. We are going to finish our project with implementing state-of-the-art embeddings: ELMO and BERT.

3.3.1 CRF

Conditional Random Field is a statistical model which is used very often in the NER field. Up to some time ago the most successful NER models were based on CRF. We provide a list of features for each token in the sentence as an input to this model. Based on these features, the model tries to predict a label for each word in the sentence, or, better said, an optimal sequence of labels for the entire sentence. The hard part here is finding the types of features that are useful and can improve the predictions. Luckily, CRFs are widely popular and there is a lot of research done. We will use the feature dictionary suggested by *sklearn-crfsuite*. Some of the token's properties that are taken into account are whether it starts with uppercase or lowercase, what is its POS tag, is it a digit or not, etc.

3.3.2 Bi-LSTM with CRF layer

Bi-LSTM networks are used when we need to process sequences where the past and future features are equally important, just like in our case. The first layer in our network is the Input layer, which carries only the shape of the input data. Next comes the Embedding layer. It takes three parameters as an input: size of the vocabulary of the dataset, dimensions of the embedding (size of the vector space in which the words will be embedded) and the length of the input. Since all our sentences are padded, this length is actually the length of the longest sentence.

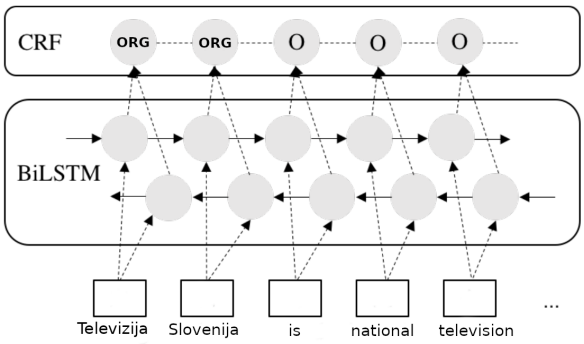

Figure 1: Bi-LSTM-CRF visual scheme

This layer is followed by Bi-LSTM layers. Bi-LSTM is comprised of two separate LSTM layers. One reads the sentence from the beginning to the end (left to right) and the other reads it in a reversed order. We are going to optimize this layer with respect to *units*, *dropout* and *recurrent_dropout*. The *return_sequence* parameter is set to True, which means that the layer will return the full sequence of the output, and not just a final value. That is why after the Bi-LSTM comes the TimeDistributed layer. This layer allows to apply the next layer to every element of the sequence independently. The outputs of the Bi-LSTM are actually scores of each label. These scores are handed over to the CRF layer and the label that has the highest score will be selected as a prediction for the token. The visual representation of the model can be seen on Fig. 1. We could easily create a model without the last CRF layer, but researchers have shown that using CRF can improve the performance of the network, because it allows to better capture the context of the sequence.

3.3.3 ELMo

ELMo (Embeddings from Language Models) is one of the most used approaches lately for data representation. The embedding is a function of the entire sentence that contains the embedded word, which means that one word can have different embeddings in different sentences, depending on the context. This makes ELMo superior to other embeddings like word2vec or GLoVe. We use ELMo on top of the previously defined Bi-LSTM network (Fig. 2). We will not include the CRF layer since there were a lot of complications due to different requirements for Keras, Tensorflow and Keras-contrib.

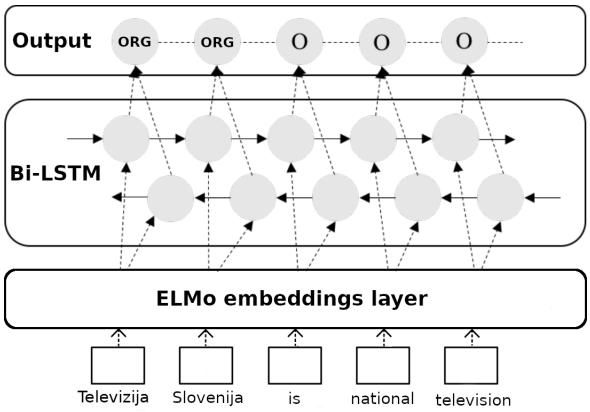

Figure 2: ELMo + Bi-LSTM visual scheme

3.3.4 BERT

BERT or Bidirectional Encoder Representations from Transformers is pre-trained on a very large unlabeled corpus, which includes texts from Wikipedia and Book Corpus. It uses a relatively novel technique called Masked Language Model, which means that it masks words in the sentence and then tries to predict them, while using the entire context of the sentence. The key difference between Bi-LSTM and BERT is that BERT can take the previous and the following token(s) in the *same time*. The real benefit of BERT is that it can be used for fine-tuning on a smaller dataset for some specific NLP task. We used the pre-trained BERT base cased model, since this is the suggested version for NER-related tags.

4 Results and discussion

In almost all of the cases we obtained about 97% - 99% accuracy. But, we could not use this measure to evaluate our models because it is not appropriate for this type of predictions. We are expecting that most of the tokens are assigned the label **O** since it is the most common label and it outnumbers the other labels which are much more important. That is why we are using F1-score for evaluation, which is a better metric in the case of imbalanced labels. The results of our experiments can be seen in Table 2 (CoNLL-2003 dataset) and Table 3 (GMB dataset). We added some performance measurements from other CoNLL-2003 papers as well. GMB is mostly used in tutorials and online courses and it was hard to find official results from conferences. We did not want to include irrelevant data, so we put only our results in the GMB table.

The final F1-score for the each model is calculated properly according to the weights of each tag.

Model	Parameters	F1-score
CRF	c1: 0.09 c2: 0.002	89%
Bi-LSTM-CRF	Embedding size: 45 Dropout: 0.1 Epochs: 12	85%
Tuned BERT	Epochs: 3 Adam optimizer	93%
ELMo + Bi-LSTM	Epochs: 5	94%
CRF (Florian et al., 2003)		88%
CRF (Ciaramita and Altun)		90%
BI-LSTM-CRF (Huang et al., 2015)		84%
Bi-LSTM (Athavale et al., 2016)		89%

Table 2: Our vs. others F1-Scores for CoNLL2003

4.1 Parameter tuning

For the CRF model we used *lbfgs (Gradient descent using the L-BFGS method)* algorithm for optimization, because it is supposed to achieve better solution than regular Gradient descent with less iterations. We optimized the regularization parameters *c1* and *c2*. We also combined the training dataset of CoNLL-2003 with one of its test sets, in order to see whether that will contribute to better performance.

For the Bi-LSTM model, we used relatively small dropout and recurrent dropout rates, somewhere between 0.1 and 0.2. The performance did not improve with increasing these values. We also tried different embedding sizes, but the best F1-score was achieved for embedding size 45-50. Even though we started with fairly small number of epochs, eventually we realized that with more epochs we can obtain better results, but, as always,

there is a point of diminishing return. After increasing the number of epochs over 15, the improvement was negligent.

Next, we changed the word embedding to ELMo embedding and trained the Bi-LSTM model once again. We used from 150 to 1024 LSTM units and the best F1-score was achieved with 512 units. We tried changing the dropout and recurrent dropout rates from 0.1 to 0.3, but we realized that the performance is the best if they are around 0.2.

We used the pytorch version for Bert, because we had a lot of problems with tensorflow. We chose to use the BertAdam optimizer and LayerNorm for improved generalization accuracy.

Model	Parameters	F1-score
CRF	c1: 0.5 c2: 0.1	85%
Bi-LSTM-CRF	Embedding size: 45 Dropout: 0.15 Epochs: 12	82%
Tuned BERT	Epochs: 3 AdamBert optimizer	82%
ELMo + Bi-LSTM	Epochs: 5	85%

Table 3: Results from our experiments on GMB dataset

4.2 Discussion

In the following two tables (Table 4 and 5) we can see the F1-scores for each tag separately. Since most of the tags (but not all of them) appear in IOB format, their F1-score is the average of the F1-scores of both I-tag and B-tag (this output comes from *sklearn_crfsuite*) because it is convenient to analyze the performance for the entire class at once. Small disadvantage of this approach is that the averaging the two scores can sometimes lead us to wrong conclusions. For example, the *B-misc* label in CoNLL-2003 had 0.90 F1-score for 1263 samples and *I-misc* had 0.0 F1-score, but only for 4 samples. If we take a look at Table 4, we can see that the F1-score for *misc* is 0.45. Unless we explore the results more thoroughly, this score will lead us to believe that the performance for *misc* class is not that great.

We have checked the other labels as well and this problem is not present there.

Next, we can see that some of the tags are generally easier to learn, like person (*per*) and location (*geo* / *loc*). These entities appear in a lot of samples, because they are frequently used in real life

CoNLL-2003 F1-scores		
Label	**F1-score (ELMo + Bi-LSTM)**	**F1-score (BERT)**
org	0.91	0.86
loc	0.95	0.95
per	0.98	0.96
misc	0.45	0.86

Table 4: Comparison of F1-score for ELMo + Bi-LSTM and BERT models on CoNLL-2003 dataset.

GMB F1-scores		
Label	**F1-score (ELMo + Bi-LSTM)**	**F1-score (BERT)**
geo	0.84	0.87
gpe	0.86	0.94
per	0.6	0.79
org	0.66	0.72
tim	0.84	0.83
art	0.36	0.03
nat	0.68	0.31
eve	0.2	0.34

Table 5: Comparison of F1-score for ELMo + Bi-LSTM and BERT models on GMB dataset.

communication. On the other hand, there exists a separate class label for artifacts (*art*), which is hard to identify even by humans, let alone an artificial model. We are not surprised that this type of labels are a lot harder to learn. Also, we would like to point out that *org* in CoNLL-2003 is slightly different from the label with the same name in GMB. It is more general in CoNLL, meaning that more subjects are included in this category, while in GMB the tag is more strict, hence the difference in the F1-scores.

Finally, since both of the datasets have different labels and total number of levels, it is expected that the models will learn differently when trained on each of them. Because of that, we cannot expect to have two identical outputs for the same input.

If we run the two ELMo + Bi-LSTM classifiers on the following examples: *"I just had breakfast in London, in Blue Cafe."* and *As Harry rides the Hogwarts Express on his journey back from school after his fourth year and the dire Triwizard Tournament, he dreads disembarking from the train.* we would obtain somewhat different predictions for the crucial entities.

We had omit the rest of the words in the table for the second sentence, due to legibility (all of them

were correctly classified as *O* by both models).

We can see that Blue Cafe is identified as a location by the model trained on GMB and as a organization/location by CoNLL. Which of these is more true than the other is debatable because it depends a lot on the entire context of the conversation and not only on the single sentence provided as an input.

Word	Pred. by ELMo+Bi-LSTM trained on GMB	Pred.by ELMo + Bi-LSTM trained on CoNLL
I	O	O
just	O	O
had	O	O
breakfast	O	O
in	O	O
London	I-loc	B-geo
in	O	O
Blue	I-loc	I-org
Cafe	I-loc	I-geo

Word	Pred. by ELMo+Bi-LSTM trained on GMB	Pred.by ELMo + Bi-LSTM trained on CoNLL
Harry	B-nat	I-per
Hogwarts	I-org	I-org
Express	I-geo	I-org
Triwizard	I-org	I-misc
Tournament	B-art	I-misc

In the Harry Potter example it is evident that the model trained on GMB is struggling to predict the correct labels for Triwizard Tournament. It marked the entities with different labels because it could probably not recognize that these two go together (they are, in fact, a name of a magical contest in the Harry Potter books). This is a lot simpler task for the CoNLL model, because it can assign miscellaneous tag to both entities, after it successfully throws away all the other possibilities, including the *O* tag.

5 Conclusion and future work

For the purpose of this assignment we have implemented a few models for named entity recognition: CRF, Bi-LSTM + CRF, ELMo + Bi-LSTM and fine-tuned BERT. Even though we did a lot of extra work because of using two datasets, it was really interesting to compare their predictions on various inputs in the end. There is, however, one drawback in our work - we only use English datasets. We would really like to get the chance to work on a dataset in Macedonian some day.

In the future, we believe that we could improve our results using more than one embedding type at once (ex. BERT + Flair, ELMo + BERT etc).

References

Vinayak Athavale, Shreenivas Bharadwaj, Monik Pamecha, Ameya Prabhu, and Manish Shrivastava. 2016. Towards deep learning in hindi NER: an approach to tackle the labelled data sparsity. *CoRR*, abs/1610.09756.

Massimiliano Ciaramita and Yasemin Altun. Named-entity recognition in novel domains with external lexical knowledge. in advances in structured learning for text and speech processing workshop, 2005. language specific issue and feature exploration. In *In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 363–370, USA. Association for Computational Linguistics.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 168–171.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, page 1–7, USA. Association for Computational Linguistics.

Ji-Hwan Kim and Philip C. Woodland. 2000. A rule-based named entity recognition system for speech input. In *INTERSPEECH*.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models.

Named Entity Recognition in Slovene Language: Intermediate Report

Tilen Kopač and **Jure Zajc**
Faculty of Computer and Information Science
University of Ljubljana
Večna pot 113, 1000 Ljubljana

Abstract

Named entity recognition is a problem which today's models solve very accurately. This is true especially for languages like English which have enormous corpora and are not so morphologically rich, but not so much for a lot of others. We present a transformer-based model, fine-tuned on a Slovene annotated corpus, which achieves an F-score similar to the one achieved by the original BERT model on data in English language.

1 Introduction

Named entity recognition (NER) aims to locate entities in unstructured text and classify them into groups such as names of people and organizations, locations, quantities and other. State-of-the art NER models achieve F-scores above 90 % for the English language, but systems for other languages fail to achieve such high scores. For a long time, state-of-the-art NER models were neural networks with recurrent architecture (RNNs) composed of long short-term memory (LSTM) and conditional random field (CRF) layers. In recent years, performance of this kind of models was surpassed by transformer-based models which use attention mechanisms to circumvent the weaknesses of RNNs.

A lot of work was also done on NER for Slovene as well as other Slavic languages. Compared to the English language, performing NER on these languages is harder because they are morphologically richer and have fewer learning resources like corpora. The situation for Slovene language is now changing, as more and more researchers are getting involved and a multitude of new language tools are being developed.

2 Related Work

Our work will be focused on deep learning models. These show a lot of potential in named entity recognition and are achieving very good results on challenging tasks (Yadav and Bethard, 2019). Models of different approaches mostly differ in their architectures.

Chiu and Nichols achieved an F-score of 91.62 % with their model using a hybrid bidirectional LSTM and convolutional neural network (CNN) architecture (Chiu and Nichols, 2016). This detects word-level as well as character-level features, diminishing the need to rely on large knowledge bases. They also propose an efficient method of encoding partial lexicon matches.

Huang et al. got a similar score using a similar architecture but only on word level (Huang et al., 2015). Their additions include a CRF layer for which they claim is used for the first time in combination with a bidirectional LSTM network in a natural language processing model.

For languages other than English, models working on word-, character- and affix-level seem to produce better results. This was shown by Yadav et al. who achieved very good scores on Spanish, Dutch and German benchmarks as well as on the English ones (Yadav et al., 2018). Their model is built of two LSTM layers and a CRF.

One approach for named entity recognition in Slovene language achieves an accuracy of 74 % and a recall of 72 % using supervised machine learning based on CRF (Štajner et al., 2013). Their model was trained on the ssj500k training corpus which presents a valuable corpus for Slovene language with about 500.000 annotated tokens (Krek et al., 2019).

Arkhipov et al. use the Bidirectional Encoder Representations from Transformers (BERT) model to achieve good results on four Slavic languages (Arkhipov et al., 2019). They pre-train a

general BERT model on these languages and extend it with a word-level CRF layer.

3 Methods

3.1 Data Pre-processing

In our experiment, we use the ssj500k training corpus created by (Štajner et al., 2013). It contains around 500.000 manually annotated tokens, about half of which are also annotated with named entity annotations. Many different types of Slovene texts such as news and magazine articles and excerpts from books are included in the data set.

Data is stored in a file formatted in Text Encoding Initiative (TEI) format. Sentences are contained inside `<s>` tokens and are made of words wrapped in `<w>` tokens and punctuation marks wrapped in `<pc>` tokens. If a word or a group of words represents a named entity, it is additionally wrapped in `<seg>` tokens, with its `type` and `subtype` attributes defining the type of entity. The type of all segments we are interested in is "name", as this means the segment contains a named entity. Different subtypes are "per" for person, "org" for organization, "loc" for location, "misc" for miscellaneous and "deriv-per" for person derivative. As outlined in (Fišer et al., 2018), all types except the latter are standard, while person derivative is introduced to mark personal possessive adjectives.

The part of the corpus with named entity annotations consists of 196,526 words. 11,593 of them are named entities. Their distribution is presented in table 1. More than 80 % of them are either persons, organizations or locations. Person derivatives only present a small fraction of all named entities.

Type	Count	Percentage
all	11.593	100 %
per	4494	38.76 %
org	2726	23.51 %
loc	2481	21.40 %
misc	1727	14.90 %
deriv-per	165	1.42 %

Table 1: Counts and percentages of different entity types in the corpus

We process the data sentence-wise. Some entities include words which contain punctuation marks but are not annotated as such, for example "Damijan R.". This segment is made of two tokens, "Damijan" and "R.", but our predictor will tokenize it as "Damijan", "R" and ".". For this reason, we perform tokenization using a basic version of the same tokenizer when parsing data. Labels of entities, extended this way, are extended as well. In our example, "Damijan R." will be labeled with three "per" labels instead of just two. Words which do not represent named entities are labeled with "othr" label.

Our data set wrapper is designed so it can be used for different purposes. A language for tokenizer must be specified so the wrapper knows whether to use Slovene or multilingual tokenizer. The same one should be used as in entity prediction. One of the parameters controls whether the data will be used for training or evaluation. For the latter, only each tenth sentence is parsed, so our test set contains 10 % of all sentences. All the other ones are used for training. We can also specify whether we want to keep all labels or only three main ones. In the latter case, "misc" labels are transformed into "othr" and "deriv-per" labels are transformed into "per".

3.2 Model Architecture and Fine-Tuning Process

The architecture of our model is similar to the one described by (Arkhipov et al., 2019). We use a pre-trained BERT model on top of which we add a layer for token classification, which classifies labels of given tokens.

A scheme of the fine-tuning process is shown in appendix A. Sentences get tokenized by a pre-trained tokenizer. Label of each word is multiplied by the number of tokens the word is split into. Tokens are transformed into IDs according to the tokenizer's vocabulary. Tokenized sentences and labels are padded so that all are of the same length. We create attention masks, which tell the model, which parts of each sentence are the original sentence and which are just padding, so that it doesn't take the padding into account when learning. Training data is then split into training and validation set with the latter consisting of 10 % of all sentences. We pack them into PyTorch tensors and create data loaders which will feed them to our model.

We opt for fine-tuning of all of model's weights instead of just the ones in the last layer, which serves as a classifier. In each epoch, labels are pre-

dicted and backward pass of the loss is performed. We also calculate loss on validation set, so we know when the model starts over-fitting training data. We use similar parameter values as the ones used in the original BERT paper (Devlin et al., 2018):

- **Embedding size:** 128
- **Batch size:** 32
- **Number of epochs:** 3
- **Learning rate:** 3e-5

4 Results

We evaluate performance of four different models: Slovene and multilingual simple and Slovene and multilingual full. Slovene models use a Slovene pre-trained BERT as the base for fine-tuning and multilingual ones use a multilingual pre-trained BERT. The words "simple" and "full" refer to the set of labels they are trained on and can predict. Full models are trained on all five labels, while we only use "per", "loc" and "org" labels when training simple models.

All models are evaluated on the same train set. We calculate their recall, precision and F-score, shown in table 2 and figure 1. This is a total score over all entity types, where all entities are classified as positive cases and other tokens are classified as negative ones.

Model	Recall	Precision	F-score
Sl. [F]	0.7473	0.9712	0.8447
Multi. [F]	0.8671	0.9128	**0.8894**
Sl. [S]	0.8415	0.9663	0.8996
Multi. [S]	0.9188	0.9117	**0.9152**

Sl. – Slovene model
Multi. – multilingual model
$[S]$ – simple model
$[F]$ – full model

Table 2: Recall, precision and F-score of four evaluated models. Multilingual models achieve better results

Full models achieve slightly lower F-scores, as they deal with a harder problem than simple ones. Contrary to expectations, models based on Slovene BERT perform worse than multilingual ones. The problem does not lie in classifying detected entities, but detecting them in the first place, as Slovene models' recall is much lower than that of multilingual models. Our best model overall is the multilingual simple model, which achieves an F-score of approximately **0.92**.

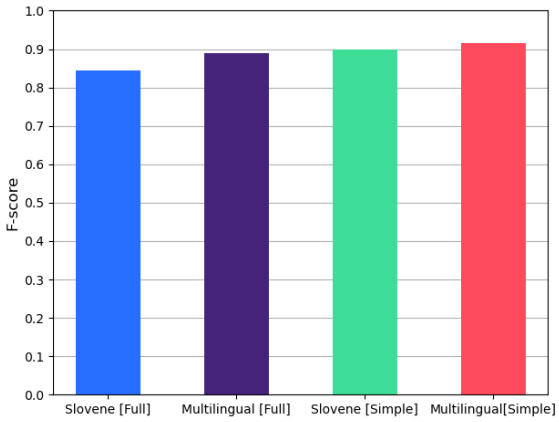

Figure 1: F-scores of models, calculated over all predictions

Evaluation of our models' performance for each individual label is shown in figure 2. Slovene models perform slightly better than multilingual ones when classifying person entities, but they are worse at classifying other entity types. Out of the basic three types, they struggle at predicting organization entities the most, which is clearly evident by their F-scores of about 0.8 and lower, compared to more than 0.96 for person entities. Interestingly, the full model predicts them better than the simple one.

The difference in performance is even more obvious when comparing predictions for miscellaneous and person derivative entities. Slovene full model fails in predicting these types correctly, while the multilingual one seem to perform well even in these cases. Miscellaneous is the only entity type with whose prediction the latter struggles, as it achieves an F-score lower than 0.8. Difference in performances of our model for different entity types is comparable with that of the model of (Štajner et al., 2013), who point out that predicting miscellaneous entities is hard as this class is very diverse.

We plot probability distributions of real entity type distribution as well as the distribution of predictions of our models, shown in figure 3. The simple ones perform similar, but the multilingual model's prediction distribution is closer to the real one. The full models' predictions differ much more. The Slovene model predicts a token is a person in more than half of all entity predictions,

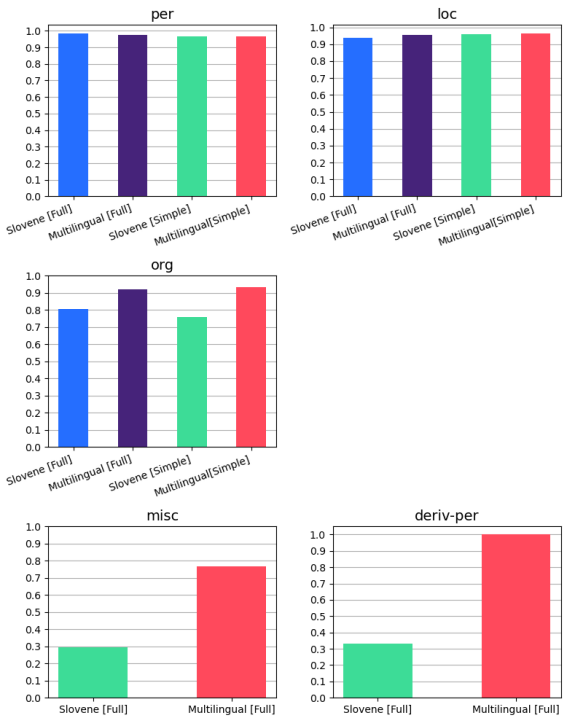

Figure 2: F-scores of models for each entity type

but very rarely predicts a miscellaneous or person derivative entity. The probability of predicting an organization entity is much closer to the real distribution, but knowing that its F-score for this type is low, we assume these predictions are wrong. In contrast, the multilingual model predicts organization entities less often and miscellaneous ones much more often.

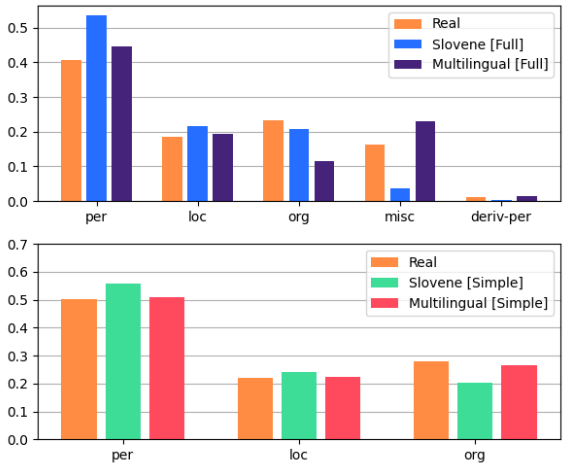

Figure 3: Probability distributions of models' predictions

5 Conclusion

We fine-tuned four different models for named entity recognition in Slovene language. Training was performed on pre-processed ssj500k corpus containing about 200,000 sentences with named entity annotations. Two models used a Slovene and two of them a multilingual pre-trained BERT as their base. For each base, a full model was trained on all different entity types and a simple one was trained only on the three basic ones.

Surprisingly, multilingual models achieve better F-scores in general as well as for all entity types except persons, where Slovene models perform better. The best performing model is multilingual simple model, which achieves an F-score of approximately 0.92, while a full multilingual model achieves a slightly lower score of 0.89. This is similar to the performance of the original BERT model on data in English language, described by (Devlin et al., 2018). The result also showcases the progress made in this field, as a CRF-based model by (Štajner et al., 2013) trained on the same corpus achieved a much lower F-score.

Our best model struggles the most when predicting miscellaneous entities. Of the three basic entity types, organizations appear to be a bit harder to predict than persons and locations. Person derivative entities seem to be predicted perfectly, but it is hard to say this for certain, as the test set only contains 15 such occurrences. Performance could certainly be improved by training our model on a corpus of additional sentences containing more miscellaneous and person derivative entities.

References

Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, and Alexey Sorokin. 2019. Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93, Florence, Italy. Association for Computational Linguistics.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-

training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Darja Fišer, Nikola Ljubešić, and Tomaž Erjavec. 2018. The janes project: language resources and tools for slovene user generated content. *Language Resources and Evaluation*, pages 1–24.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Simon Krek, Kaja Dobrovoljc, Tomaž Erjavec, Sara Može, Nina Ledinek, Nanika Holz, Katja Zupan, Polona Gantar, Taja Kuzman, Jaka Čibej, Špela Arhar Holdt, Teja Kavčič, Iza Škrjanec, Dafne Marko, Lucija Jezeršek, and Anja Zajc. 2019. Training corpus ssj500k 2.2. Slovenian language resource repository CLARIN.SI.

Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Vikas Yadav, Rebecca Sharp, and Steven Bethard. 2018. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172, New Orleans, Louisiana. Association for Computational Linguistics.

Tadej Štajner, Tomaž Erjavec, and Simon Krek. 2013. Named entity recognition in slovene text. *Slovenščina 2.0: empirical, applied and interdisciplinary research*, 1(2):58–81.

A Fine-Tuning Process Scheme

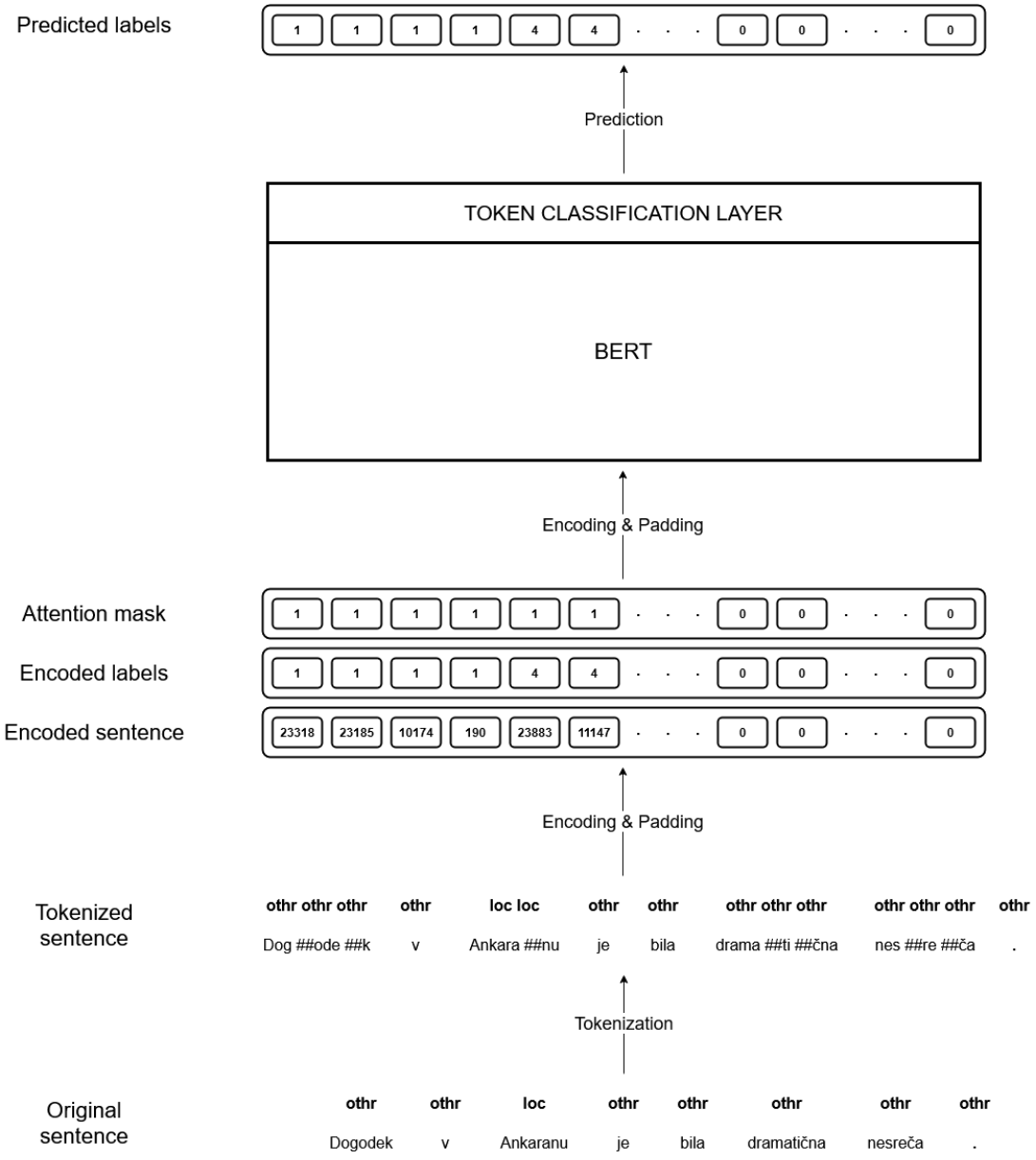

Figure 4: Scheme of the fine-tuning process of our model. When predicting, padding is not performed so attention masks are not needed and labels are not known.