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.

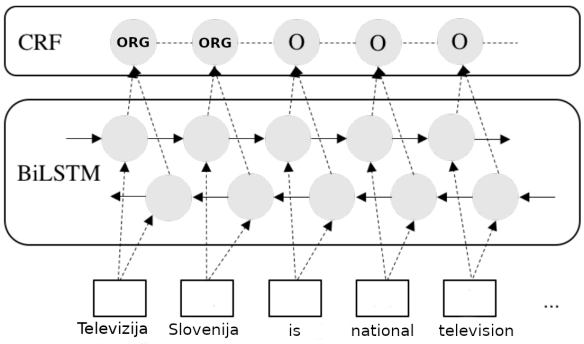

Figure 1: Bi-LSTM-CRF visual scheme

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.

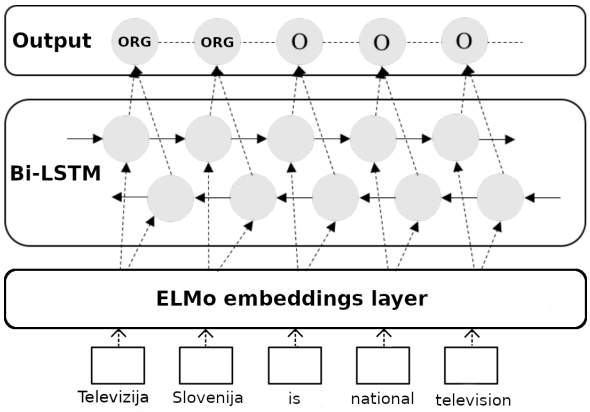

Figure 2: ELMo + Bi-LSTM visual scheme

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.