ComboNER: A Lightweight All-In-One POS Tagger, Dependency Parser and NER

Aleksander Wawer
Institute of Computer Science, Polish Academy of Sciences
Jana Kazimierza 5
01-248 Warszawa, Poland
axw@ipipan.waw.pl
orcid.org/0000-0002-7081-9797

Abstract

The current natural language processing is strongly focused on raising accuracy. The progress comes at a cost of super-heavy models with hundreds of millions or even billions of parameters. However, simple syntactic tasks such as part-of-speech (POS) tagging, dependency parsing or named entity recognition (NER) do not require the largest models to achieve acceptable results. In line with this assumption we try to minimize the size of the model that jointly performs all three tasks. We introduce ComboNER: a lightweight tool, orders of magnitude smaller than state-of-the-art transformers. It is based on pre-trained subword embeddings and recurrent neural network architecture. ComboNER operates on Polish language data. The model has outputs for POS tagging, dependency parsing and NER. Our paper contains some insights from fine-tuning of the model and reports its overall results.

1 Introduction

The paradigm for automated language processing was originally based on building dedicated tools for every task. We illustrate it on the example of the Polish language, but it is very likely the case also for other languages. The dedicated tools include morphological taggers and disambiguators (a Polish language approximation of POS tagging), dedicated dependency parsers and named entity recognizers. For convenience, the tools sometimes were made available as web services and possibly composed into chains. One such example is Multiservice (Ogrodniczuk and Lenart, 2012), a linguistic Web service for Polish, combining several mature offline linguistic tools in a common online platform. Multiservice offers a number of pre-defined chains. For example, the dependency parser requires morphological tagging and disambiguation as the initial step. The two tools are put together in appropriate order in one chain.

Among universal tools (not dedicated to the Polish language) it is worth to mention:

- UDPipe, a pipeline (tool) for tokenization, tagging, lemmatization and dependency parsing (Straka et al., 2016).
- NLP-Cube which in addition to the capabilities mentioned for UDPipe offers sentence segmentation and lemmatization. The reported results cover many languages but not Polish (Boroș et al., 2018)
- Stanza, a tool for tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) and morphological features tagging, dependency parsing, and named entity recognition. Named entity recognition models are separate and Polish is not available (Qi et al., 2020).

Recently, with the advances of neural networks in natural language processing, new tools were proposed that simplify the chains by doing several tasks at once. One example is Combo (Rybak and Wróblewska, 2018), a bi-directional recurrent neural network (RNN) using word2vec embeddings with multiple outputs for morphological and dependency analysis. The model was submitted to the CoNLL 2018 Universal Dependency shared task.

The Combo tool, while doing certain things right, has its own issues. The biggest problem is the fact that it requires word-level word2vec embeddings for every token that it encounters. Unfortunately, the number of unique tokens in the Polish language is rather large. For example, in the National Corpus of Polish and Wikipedia, the number of unique case-sensitive tokens exceeds 2 millions. This is by far too many to fit into GPU memory. This

Proceedings of Recent Advances in Natural Language Processing, pages 1508–1514
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_169

leads to a situation where only word2vec vectors that are used in input texts are loaded into GPU memory (Embedding layer) and an additional preprocessing step of data-driven word vector selection and loading. Depending on linguistic productivity, this operation has to be repeated once in a while when processing large amounts of text. To solve this problem, one can apply techniques of vocabulary selection such as those described in (Chen et al., 2019). Hopefully, another simple solution exists for this issue: subword level embeddings (Sennrich et al., 2016). This idea, successfully applied in machine translation, became popular in multiple other scenarios and architectures, including transformer neural networks such as BERT (Devlin et al., 2019). It allows to represent an unknown word (not present in the dictionary) as a sequence of shorter tokens or subwords. Using subword-level tokenization and embeddings the problem of vocabulary selection is non-existent.

The goal of ComboNER is to (1) avoid the above issues of the initial Combo (Rybak and Wróblewska, 2018) by introducing subword tokenization and (2) add named entity recognition as another output. The overall design goal is to reduce the size of the model where possible.

2 Datasets

This section describes the datasets used to train the ComboNER model. It is only an overview with basic information about data sizes, please refer the cited papers for more details such as label descriptions and annotation principles.

2.1 POS and Dependency

We trained the POS and dependency outputs of the ComboNER model using the Polish language subset of the Universal Dependencies (UD) treebank (Nivre et al., 2020). It contains 17723 sentences in the train set and 2215 sentences in the test and dev sets. Sentences are annotated for part-of-speech, morphological information and dependencies (heads and labels).

2.2 Named Entity

To train the named entity output of ComboNER, we used the relevant subset of the National Corpus of Polish (Przepiórkowski et al., 2012). For label descriptions and annotation principles for named entities refer to the chapter 9.

Overall, the dataset contains 39534 texts and 85816 sentences. Since ComboNER requires exactly one sentence as its input (a limit imposed by the dependency structure defined for a single sentence only), we splitted the named entity data into single sentences using the Polish language model of the spacy.io[1]. We then randomly divided the data into train and test part. This resulted in 81470 sentences (95%) in the train set and 4346 (5%) in the test set[2].

3 Hyperparameter Tuning

To tune the parameters we used only POS accuracy and one measure from the dependency parsing: UAS score (unlabeled attachment score). This was in order to simplify the analysis and check the balance between types of outputs. Therefore, the hyperparameter tuning was carried out only on the Universal Dependencies dataset.

Hyper-parameter tuning was done using the HParams API in TensorFlow[3] and grid-search approach.

Tuning was subjected to the following hyperparameters:

- Length of the subword embedding vector (emb_dim). Values tested: 25, 100, 200.
- The number of subwords for which we have vectors, i.e. the size of the vocabulary (vocab): 10,000 (10k), 100,000 (100k).
- The size of the Bi-LSTM cell output assuming a single layer model (rnn). Tested values: 100, 200.

ComboNER uses pre-trained Polish language subword embedding vectors from the BPEmb library (Heinzerling and Strube, 2018). Therefore, vocabulary and vector sizes are determined by availability in BPEmb.

For initial parameter selection only the results of the first training epoch were taken into account.

The Table 1 shows tuning results of the selected output as the accuracy of the POS tagger and dependency parser - where the UAS measure was selected. It also shows the mean of both measures in the AVG column.

[1] https://github.com/ipipan/spacy-pl

[2] We did not use a validation split because hyperparameter selection was focused on UD part of the data.

[3] https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams

Settings			Results		
emb_dim	**vocab**	**RNN**	**POS Acc**	**UAS**	**AVG**
25	10k	100	0.704	0.53	0.617
25	10k	200	0.706	**0.572**	**0.639**
25	100k	100	0.706	0.545	0.6255
25	100k	200	**0.713**	0.547	0.36
100	10k	100	0.719	0.55	0.6345
100	10k	200	0.725	**0.562**	0.6435
100	100k	100	0.725	0.474	0.5995
100	100k	200	**0.735**	0.558	**0.6465**
200	10k	100	0.726	0.556	0.641
200	10k	200	0.737	**0.567**	**0.652**
200	100k	100	0.732	0.529	0.6305
200	100k	200	**0.749**	0.555	**0.652**

Table 1: Hyperparameter optimization. Settings: embedding vector size (emb_dim), size of the vocabulary (vocab), dimensionality of the output space from Bi-LSTM (RNN). Results: POS tagger accuracy (POS Acc), unlabeled attachment score (UAS) and mean value of both former scores (AVG). Results are measured after training each model for one epoch only.

The largest embeddings (emb_dim equal to 200) work best in combination with size-corresponding Bi-LSTM cells (rnn equal to 200). This observation concerns both the accuracy of POS tagging and the UAS measure of the dependency parser.

Interestingly, the number of subwords, i.e. vocabulary size (vocab), turned out to be a non-obvious measure. The POS tagger performed better with a large number of subwords (100k vocab), while a dependency parser preferred a smaller number (10k vocab). This effect was present regardless of RNN cell size.

We also tested the configuration of the model with two Bi-LSTM layers. We tested two values of LSTM cell sizes for the first layer (100, 200) and two (25, 50) for the second layer. When measuring performance for the first training epoch, a model with a two-layer Bi-LSTM performs generally worse than a single-layer model with similar settings. As a result, the addition of a second Bi-LSTM layer does not seem to be advisable.

For the named entity recognition, two variants of the model architecture were tried. In the Multiservice (Ogrodniczuk and Lenart, 2012) (and in many older generation NLP tools), named entity recognition model uses POS tags as an input feature and is therefore located after POS tagging. This observation leads to testing two design choices for the named entity recognition part of ComboNER. The first one is two independent outputs for NER and POS / dependency: the outputs share only the word embedding layer. The second assumes dependence: named entity branch takes as an input not only embeddings but also hidden states from the LSTM in POS / dependency branch. This setting allows named entity recognition to take into account information from layers trained on syntactic data, in manner somewhat similar to using POS features in older NLP models.

The results in terms of F1 scores of both variants are presented in Table 2. They clearly indicate that syntactic information helps in named entity recognition as the overall gain computed as an average for all entity categories is 12,8%.

	Dep.	**Indep.**	**Gain**
Date	0.797	0.702	+0.095
OrgName	0.427	0.213	+0.214
PersName	0.683	0.634	+0.049
PlaceName	0.564	0.410	+0.154
average			+0.128

Table 2: F1 score of two variants of the NER output: (Inep.) independent outputs for NER and POS / dependency parsing, (Dep.) named entity branch takes as an input not only embeddings but also hidden states from the LSTM in POS / dependency.

4 ComboNER Architecture

Figure 1 illustrates the design of ComboNER. Data processing starts with subword tokenization and embedding layer. Here, pre-trained subword

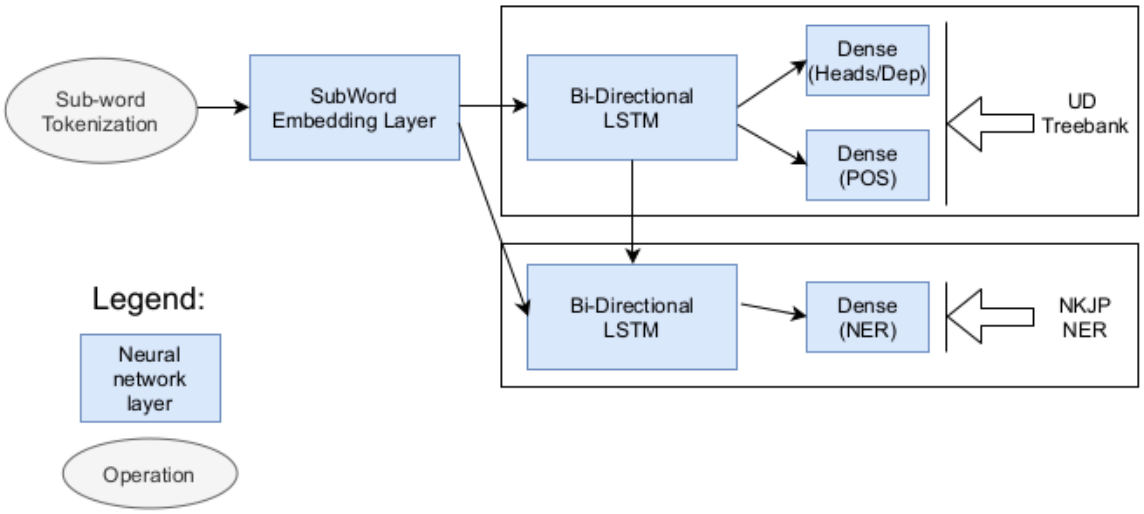

Figure 1: Diagram of ComboNER. UD Treebank and NKJP (The National Corpus of Polish) NER are datasets used to train each of the outputs. Heads/Dep are the two dependency parser outputs, one for heads and one for dependency labels.

embedding vectors are used (along with the tokenizer), namely monolingual Polish language BPEmb (Heinzerling and Strube, 2018) trained on Wikipedia. This part is common to all outputs. Then, computation is split into two branches: one for POS and dependency, the other for named entities. Each of the two branches begins with a bi-directional LSTM layer. Named entity branch takes as input not only embeddings but also hidden states from the LSTM in POS / dependency branch. Architecture of this type was selected on the basis of experiments described in Table 2. Intuitively, part-of-speech information is often useful for named entity recognition; in certain natural language processing pipelines such as Multiservice (Ogrodniczuk and Lenart, 2012) named entity recognition (NER) uses part-of-speech (POS) features and therefore is executed downstream (after) POS.

Activations inside the network are tanh and softmax is used in the final layers. For details of implementation see (Rybak and Wróblewska, 2018); in particular, dependency parser outputs with the dot product solution follow closely the initial model.

5 Final Model

5.1 Model Parameters

For the final model, we selected following hyperparameters:

- 100 output units in the POS and dependency LSTM,
- 50 output units in the named entity LSTM,
- dropout rate of 0.2,
- learning rate of 3e-4,
- 100 output units of the Dense layer for POS output,
- 100 output units of the Dense layer for dependency output,
- vocabulary size of 50.000 (50k),
- embedding size of 100.

The selection of embedding size 100 and 50k vocabulary is a compromise between the quality of POS and dependency outputs.

The number of parameters of the ComboNER is 5.3 millions. The size of the embedding layer alone is 5 millions (embedding size times vocabulary) and we make it trainable for both training tasks as it improves the overall results marginally. The remaining parts of the model are rather modest with named entity part contributing over 100k parameters. The disk size of the model slightly exceeds 200 MB.

The model handles sentences up to 67 subword tokens long as this was the maximum sentence length encountered in the UD dataset. It assumes that the input consists of a single sentence. No

special tokens are needed to indicate beginning or end of sentence.

5.2 Evaluations

We trained the model 45 epochs on the Polish language UD dataset (POS and dependency part) and 20 epochs on the subset NKJP (the National Corpus of Polish, named entity labels) as described in Section 2. The execution time of a single training epoch was 3 minutes on the UD treebank (POS and dependency) and about 10 minutes on the named entity data on a Tesla V100-SXM2-16GB GPU card.

Table 3 contains the POS results of the final fully trained output of ComboNER. The overall accuracy (not reported in the Table) is 0.933. The performance is satisfactory for most parts-of-speech. The worst performing one is PROPN (proper names), due to high lexical variation and relatively low frequency in the UD corpus.

In the case of dependency parser outputs, the two relevant measures are unlabelled attachment score (UAS) and labelled attachment score (UAS). The values measured are 0.71 for UAS and 0.858 for LAS score.

Table 4 contains the results of named entity evaluation. Two categories, present in the data, did not provide reasonable results: GeogName and Time. This is due to low counts, as these categories were the least frequent in the dataset. In the case of categories: Date, OrgName, PersName and PlaceName ComboNER performed with reasonable precision and recall.

6 Conclusions

This paper describes ComboNER: a new version of the Combo tool (Rybak and Wróblewska, 2018), developed with following assumptions in mind: (1) use subwords to avoid out-of-vocabulary issues (words in model's input not present in memory / the embedding layer), (2) add named entity recognition output, (3) lightweight in terms of memory footprint, as of today's standards. The tool was implemented in TensorFlow 2.

ComboNER follows many design choices of the original Combo (Rybak and Wróblewska, 2018), but introduces some new solutions.

Training multi-output models on multiple datasets is a challenging task. Technically, it was solved by freezing appropriate part (layers) of the model while training another part, using a dedicated TensorFlow data loader class for each corpus.

The paper contains the results of hyperparameter optimization, which dictate the final hyperparameter choice for the ComboNER model. There are several interesting findings. First, the number of subwords (vocabulary size): the POS tagger performed better with a large number of subwords (100k vocab), while dependency parser preferred a smaller number (10k vocab). Second, named entity layers greatly benefit from accessing layers pretrained for POS and dependency. This architecture improves the F1 score of named entity recognition by 12,8% on average compared to the variant with fully independent training of both tasks.

The results of PolEval competition in 2018 are an interesting reference for comparisons regarding dependency parser and named entity recognition quality (Ogrodniczuk and Kobyliński, 2018)[4]. It appears that LAS score achieved by the ComboNER is competitive and easily on-par with participating systems. Unfortunately, the solutions for named entity recognition outperform the output of ComboNER in a significant manner. One explanation of this fact is context: effective named entity recognition requires context wider than just the input sentence. Most of the corpora and tools assume the supra-sentence level. Important clues for identyfing named entities are often a part of preceding sentences.

In order to evaluate the POS tagger, one can compare the results of the PoleEval competition in 2017(Kobyliński and Ogrodniczuk, 2017)[5]. In this context, the accuracy of 0.933 is very competitive and on-par with the best of the participating systems.

To recap, this paper describes a relatively straightforward yet important modifications over previous work (Rybak and Wróblewska, 2018). Our goals were to develop a small system and avoid large scale transformer, at the same time solving out-of-vocabulary problems posed by word-level embeddings. In the future it may be advisable to compare with small transformer models like DistilBERT (Sanh et al., 2020) (40% less parameters than the BERT base model).

[4] http://2018.poleval.pl/index.php/results/

[5] http://2017.poleval.pl/index.php/results/

	prec	recall	F1
ADJ	0.983	0.968	0.975
ADP	0.935	0.979	0.957
ADV	0.788	0.630	0.700
AUX	0.812	0.904	0.856
CCONJ	0.929	0.892	0.910
DET	0.882	0.772	0.823
NOUN	0.750	0.859	0.801
NUM	0.783	0.757	0.770
PART	0.857	0.742	0.795
PRON	0.905	0.948	0.926
PROPN	0.754	0.602	0.670
PUNCT	0.997	9.996	0.997
SCONJ	0.876	0.911	0.893
VERB	0.876	0.911	0.893

Table 3: Part-of-speech evaluation of the fully trained model. Precision (prec), recall and F1 measured on the test set.

	prec	recall	F1
Date	0.891	0.721	0.797
OrgName	0.640	0.320	0.427
PersName	0.747	0.629	0.683
PlaceName	0.653	0.496	0.564

Table 4: Named entity recognition evaluation of the fully trained model. Precision (prec), recall and F1 measured on the test set. GeogName was skipped due to low results.

6.1 Software and Data

Source codes and serialized models are available at https://github.com/CLARIN-PL/ComboNER/ and https://github.com/alexwz/ComboNER.

Acknowledgments

Work financed by the Polish Ministry of Science and Higher Education, a program in support of scientific units involved in the development of a European research infrastructure for the humanities and social sciences in the scope of the consortium CLARIN ERIC 2018-2021.

References

Tiberiu Boroș, Stefan Daniel Dumitrescu, and Ruxandra Burtica. 2018. NLP-cube: End-to-end raw text processing with neural networks. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 171–179, Brussels, Belgium. Association for Computational Linguistics.

Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Yang Wang. 2019. How large a vocabulary does text classification need? a variational approach to vocabulary selection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3487–3497, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Łukasz Kobyliński and Maciej Ogrodniczuk. 2017. Results of the PolEval 2017 competition: Part-of-speech tagging shared task. In *Proceedings of*

the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pages 362–366, Poznań, Poland. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Maciej Ogrodniczuk and Łukasz Kobyliński, editors. 2018. *Proceedings of the PolEval 2018 Workshop*. Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Maciej Ogrodniczuk and Michał Lenart. 2012. Web Service integration platform for Polish linguistic resources. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, pages 1164–1168, Istanbul, Turkey. European Language Resources Association (ELRA).

Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. 2012. *Narodowy Korpus Jezyka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Piotr Rybak and Alina Wróblewska. 2018. Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *LREC*, pages 4290–4297.