Named Entity Recognition performance on Out of Vocabulary words

Ivan Korostelev
University of Alberta
korostel@ualberta.ca

Kiarash Aghakasiri
University of Alberta
aghakasi@ualberta.ca

1 Introduction

The field of Natural Language Processing has several well-established practical tasks among which are Part-of-Speech (POS) tagging and Named Entity Recognition (NER). Both of them ask the algorithm to assign one of the labels from a set of fixed size. In the first problem, the labels are typically the eight commonly accepted parts of speech [1] and the second has labels for organizations, people's names, locations, etc.

In this paper, we will focus on using the character features of the text to improve the model performance on the NER problem which restricts every word to have exactly one label. The label conventions are different across different data sets but the algorithm's structure does not require prior knowledge of the set of labels so we can test the architecture in slightly distinct settings.

Human languages are by nature dynamic, i.e. new words and morphological patterns are constantly emerging making it impossible to list an existing vocabulary and run a search algorithm within that word list. In the case of tagging problems, for every new test word the system will try to find the word within the vocabulary with a similar context window, which gives wrong predictions on practice. For example, given the term "Australian" and the knowledge of English morphology a human can think of "Aussie" as "a person from Australia" or "a person from Austria". A few examples of the term "Aussie" in context would further clarify its meaning but the pure letter structure of the word is already sufficient for knowledge building.

This intuition helps understand the difference between word-level and character-level word representations. As for the first one, each word is stored in computer memory as a reduced dimensionality vector based on the other words appearing within its context in all contexts in the training data. However, this idea does not allow to construct the meaning for a new word without providing sample context.

As human languages obey the Zipf's law, it becomes crucial to be able to grasp the morphological structure of the word from its spelling. When building models for language tagging problems, it is often the case that the main cause for prediction accuracy drop is the so-called Out-of-Vocabulary (OOV) words. Those are the words present in the training set but not among the validation samples. To address this issue, subword models should be exploited.

2 Previous work

The problem of dealing with OOV words arises in many NLP settings. It can be used to tell when someone is making a grammatical mistake, is using the wrong word, or has inappropriately conjugated verbs. In order to keep track of all those language forms, the model would require a dedicated OOV record storage system to remember every occurrence of a word. But it was shown empirically (Pinter et al., 2017) that some OOV words can be resolved with character-level modelling.

Three recent state-of-the art models for the NER problem and their results on CoNLL 2003 English are given in table 1:

In the paper (Straková et al., 2019), the authors propose two neural network architectures and test them on various NER data sets. Although the main goal of their research was to solve the nested NER problem (allows every word to have several labels), the authors provide model performance on the flat NER as well. They use a standard of the past LSTM+CRF (Long short-term memory and Conditional Random Field) and con-

[1] http://partofspeech.org

Model	F1	Paper
CNN Large + fine-tune (Baevski et al., 2019)	93.5	Close-driven Pretraining of Self-attention
RNN-CRF + Flair	93.47	Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition
LSTM-CRF + ELMo + BERT + Flair	93.38	Neural Architectures for Nested NER through Linearization

Table 1: Three recent state-of-the-arts

struct a sequence-to-sequence (seq2seq) models. Their success inspired us to use the first approach and Bidirectional Encoder Representations from Transformers (BERT).

In (Meemulla Kandi, 2018) the author investigates methods for OOV words semantic reconstruction within a context and provides sample predicted sentence tags. The main idea of the paper uses the fact that the word embedding should be adequately modeled by the linear combination of the embeddings of the context words. All the architectures built employs Recurrent Neural Networks (RNN) and the best-performer is a bidirectional LSTM. Real sentence examples show that OOV words can be adequately estimated given a context so we will build a bi-LSTM model in our work as well.

In (Ling et al., 2015), they mentioned two problems with word-level models. First, they cannot learn the structures of the words for instance -ing or -s because the look at words independently. Secondly, if there is huge amount of data available it's impossible to save a word embedding vector for all the words. Thus, they introduced character embedding. They created a character lookup table and then passed the character sequences to a bidirectional LSTM to find the embedding of the word.

These ideas have lead us to the conclusion that a character-level model in a combination with a word-level model would act most versatile for the proposed problem. Word vectors work great for the most common words whereas the addition of the subword information improves overall performance for OOV words. This intuition is implemented In the work (Garneau et al., 2019) where the authors build a simple yet competitive model for POS tagging problem. The model extends that described in (Pinter et al., 2017) so that it becomes a stack of left and right context word-level bi-LSTMs and a character level encoder, all enhanced by the Attention mechanism (Bahdanau et al., 2014).

Although such concatenation of the embeddings turned out to adjust to OOV words better on practice, the prediction power of the character-level component should be refined. The paper (Bharadwaj et al., 2016) contains ideas that address this issue. They take advantage of Attention and CRF

3 Experiments

We used CoNLL 2003 [2] and WikiNER [3] datasets to test our models, the first one consists of 203,621 tokens with POS tag 9 different NE tags, the latter consists of 2,831,601 words along with their POS tag and NE tags in 115,143 sentences with 17 different NE tags. The distribution of train, test and development data for both data sets is about 70%, 20% and 10%, respectively. For each of the methods we report two different accuracies, one is for words which were also seen in the training data set and the other one is for new words which were not in the training data set (OOV words).

There are a total of nine tags in CoNLL 2003 data set. The data contains the following entities: person (PER), organization (ORG), location (LOC) and miscellaneous names (MISC). As soon as two entities of the same tag type "***" appear immediately next to each other, the first word of each entity gets the tag "B-***" to mark the beginning of a new entity. Other words within the entity get tag "I-***". In addition to these eight tags ["B", "I"] x ["PER", "ORG", "LOC", "MISC"] there is a tag "O" for all other words, which takes 84% of all tokens in the development set 3.

3.1 IV/OOV proportion

The first and easiest method to start with for NER task is memorizing each and every word in the training data along with their tag. Surprisingly, it results in an acceptable accuracy of 0.93 but only for the words which is in the vocabulary. So, for OOV words the accuracy is 0. This shows that most words appear both in the training and test data and there is a need for manual OOV extrac-

[2] https://www.clips.uantwerpen.be/conll2003/ner/

[3] https://github.com/dice-group/FOX/tree/master/input/Wikiner

tion to provide an accurate score on the model performance.

3.2 Random Forest with feature engineering

Next naive method is Random Forest algorithm. Random Forest needs hand-crafted features, so we used two different feature sets. The first feature set includes features only related to the shape of the word (not meaning) such as length, uppercase, lowercase and etc. As POS tags are also available in the data set we used POS tags for the second set of features.

3.3 CRF

The next approach uses Conditional Random Field (CRF) which is usually chosen for sequential data. For observation we used 9 different features including POS tags and the hidden state is the NE tag of the word. The results at this point have progress in comparison with the previous algorithms, especially for OOV words. Using POS tags as CRF's observation and Random Forest's features, we reach to the fact that POS tag is very important feature in NER task and can profoundly improve the performance of methods.

3.4 LSTM

Another popular approach for sequential data task is Long Short Term Memory (LSTM). Our goal with this approach was to see some meaningful results for words with typos and to reproduce the results of (Meemulla Kandi, 2018). We did not use pretrained character or word embeddings but rather trained the model from scratch for ten epochs (until saturation). We observed that the examples from the paper (”fidditch” is recognized as a game/sports, ”lndl” as a city) hold true can be corrected in simple and common words (”liike” is treated as ”like”).

3.5 LSTM+CRF

As we mentioned earlier, the combination of LSTM and CRF was shown to provide best results in the NER task. For this approach, we built a bidirectional LSTM cell but the difference is that after the LSTM cell we have a Dense layer and the output of that layer goes to a CRF as observations and the hidden state of the CRF is the NE Tag. Again we let our model run for 10 epochs and chose 32 as batch size. As you can see in the table below there is no progress from LSTM to LSTM + CRF. The implementation of LSTM+CRF models was taken from the blog of Tobias Sterbak https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/.

3.6 Character level

In all the previous approaches, words was our smallest unit and that can be problematic when facing new words (OOV words) during test. As a much better approach especially for OOV words we used char-embedding along with word embeddings to get better results for unseen words. Char-level embedding might perform better on OOV words as it can capture the structure of words such as suffixes, affixes, stem morphology, etc; thus facing new words they can perform well in comparison with word-level models.

3.7 Proposed model (char-level attention + word-level)

After running all these algorithms, we tried to change them a little bit to be more suitable for OOV words. The idea was to add char-level attention to the last model here (char-level + word-level embedding). We used the Attention mechanism that considers the context for each timestamp [4].

After training the model for 10 epochs we achieved competitive results with 3 % increase for OOV words and 1 % increase for IV words compared to the previous method.

Model	OOV accuracy, %	IV accuracy, %
CRF	80	96
LSTM	46	**98**
LSTM+CRF	31	**98**
Char-lvl	71	95
Char-lvl+Attention	74	96
BERT	**89**	93

Table 2: Results on WikiNER data set

Another promising method for the NER problem is the Transformer model. This architecture has set state of the art in several sequence processing NLP problems (Devlin et al., 2018). We used HuggingFace's pre-trained Transformers [5] for the task of Named Entity Recognition. The

[4] implemented in keras_self_attention: https://github.com/CyberZHG/keras-self-attention/tree/master/keras_self_attention

[5] https://github.com/huggingface/transformers

model consists of 24 layers of Transformer blocks (Vaswani et al., 2017):

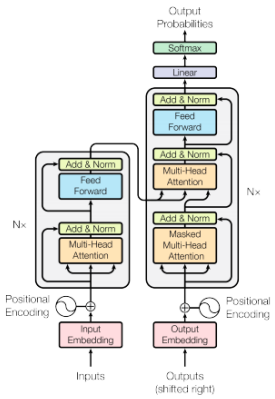

Figure 1: One transformer block

The model was pre-trained on a large cased English textual data (not disclosed by HuggingFace) and has a total of 340M trainable parameters. We retrained it for the CoNLL 2003 NER task. Even though the architecture operates on the word level, the method is good at capturing word tags from the context and gives unbeatable results in our setting.

4 Proposed model description

4.1 char-level + word-level

This implementation, consists of two parts. The first part is word level embedding which we used an embedding layer implemented in Keras. Each word consists of many characters, so we have a time distributed embedding layer of characters for each word. Then we concatenate these two embeddings and feed it to a bidirectional LSTM cell which is our decoder. We used 0.3 drop out for this layer to avoid over-fitting.

4.2 char-level + word-level + attention

We used the previous char-level + word-level embedding as the baseline and add attention to that to improve the model so we have same layers until the concatenation part. After concatenation we we want to feed this to our bidirectional LSTM cell we added an attention layer for characters and concatenate the output of the attention layer with char+word-level and feed it to the decoder which is again a bidirectional LSTM cell.

	Precision	Recall	F1-score	#
B-LOC	87.9	92.5	90.2	678
B-MISC	87.9	85.7	86.8	363
B-ORG	92.3	83.9	87.9	675
B-PER	**94.7**	91.8	93.2	661
I-LOC	64.9	88.1	74.7	109
I-MISC	86.9	73.2	79.5	127
I-ORG	91.2	77.6	83.8	401
I-PER	92.1	**95.2**	**93.6**	456
O	99.1	99.6	99.3	18770

Table 3: Metrics and counts per tag

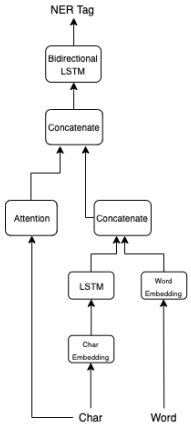

Figure 2: Character Level + Attention

You can see a brief overview of the model in figure 4.2.

5 Conclusion

As a result of this work, we have compiled a set of viable methods from the recent achievements in the field to solve the problem of Named Entity Recognition. The experiments show that the models can reasonably well predict tags for common English words or words with typos with enough context. Based on our intuition, we picked a composite model with competitive results existing methods. Besides showing better results than other models, it could not beat the remarkable performance of pretrained BERT (91.2% F1). The latter gives high recall metrics with a slightly poorer prediction quality for OOV words.

	Precision	Recall	F1	Accuracy	OOV accuracy	IV accuracy
CRF	**91.0**	**89.0**	**90.0**	98.0	**89.9**	98.8
LSTM	83.7	87.8	85.4	98.8	60.0	99.5
LSTM+CRF	80.1	86.6	82.8	**99.1**	45.8	**99.7**
Char-lvl	80.1	73.3	75.1	96.1	82.5	98.7
Char-lvl+Attention	88.6	87.5	87.7	97.7	82.3	98.6

Table 4: Model comparison

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1462–1472, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Nicolas Garneau, Jean-Samuel Leboeuf, and Luc Lamontagne. 2019. Predicting and interpreting embeddings for out of vocabulary words in downstream tasks. *CoRR*, abs/1903.00724.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Shabeel Meemulla Kandi. 2018. Language modelling for handling out-of-vocabulary words in natural language processing.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.