POS Tagging with Static and Contextual Word Embeddings

S2258327, S2160729

1 Introduction

During the last decade, there has been extensive research into the development of word embedding representations to solve various computational linguistics problems, such as intent detection, document classification, POS tag recognition, etc. Word embedding vectors could be broadly classified as static or contextual. Static word embeddings, such as the ones generated with an Skip-Gram model (Mikolov et al., 2013), assign a unique vector to each word. On the other hand, contextual word embeddings, such as the ones generated with BERT (Devlin et al., 2018), assign different vectors to the same word, depending on the context on which the word is found. Currently, BERT-based models are the most popular choice due to their state-of-the-art performance on multiple language modeling tasks.

In this paper we address the following research questions:

1. How does the performance of a POS tagger using GloVe (Pennington et al., 2014), a static word embedding model, and RoBERTa (Liu et al., 2019), a BERT-based contextual word embedding model, compare?

2. How does the choice of combination function affect the performance of POS tagging with RoBERTa?

3. What mistakes does the contextual POS tagger make? What are the potential causes of these mistakes?

The dataset used on this paper comes from the Universal Dependencies project. The training set contains 12,543 sentences and 204,608 tokens; the validation set contains 2,001 sentences and 25,151 tokens; the test set contains 2,077 sentences and 25,098 tokens.

2 Static vs. Contextual embedding POS tagging

In order to compare the results of the models that will be developed in this paper, we built a most common baseline classifier: the tag assigned to a word is the most common tag for that word in the training data.

For the first experiment, we trained two regularized (L2, λ=1) logistic regression classifiers, one using static word embeddings as input, and the second using contextual word embeddings (mean combination function). For both models the labels are the corresponding POS tags of each word vector. Figure 1 shows the classification metrics for some of the POS tags, when evaluated on the test set.

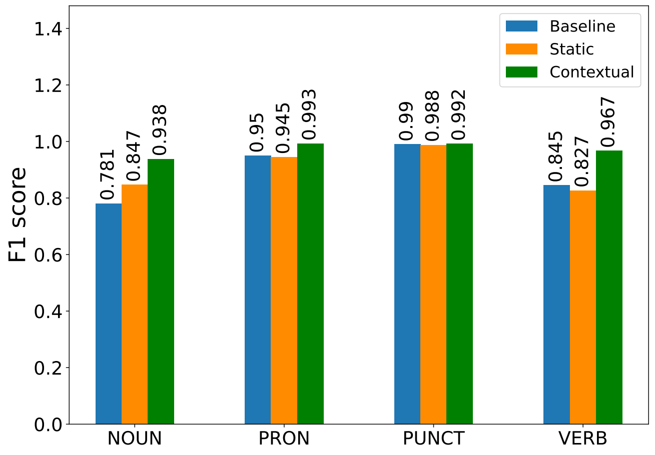

Figure 1: Static vs Contextual

In general, we see a significant improvement in performance when using contextual vectors for all tags, the weighted average F1-score is almost 10% higher when using contextual vectors (0.958) as opposed to using static vectors (0.866), as seen in Tables 3 and 4 (see A). As expected, the POS tagger using contextual word embeddings is far superior than the static and baseline classifiers. This is specially notable in tags such as NOUN and VERB where the context of a word is particularly important to determine the correct POS tag.

3 Choice of combination function for contextual embedding vectors

BERT-based models use a WordPiece tokenizer (Schuster and Nakajima (2012)). This means that words can be split into full forms (i.e. a single token) or into word pieces, where a word can be split into multiple tokens. To determine which words are split into word pieces and which are not, BERT-based models use a vocabulary; any word that is Out-Of-Vocabulary (OOV) is broken down into subwords. For example, using the auto-tokenizer for RoBERTa from HuggingFace (HuggingFace (2021)) (the tokenizer used in this paper), the word *cardiopulmonary* is

OOV and is split into the tokens ”card, iop, ul, monary” because these tokens are present in the vocabulary. The size of the vocabulary used by the auto-tokenizer is 50,265.

For words that are split into multiple tokens, the choice of combination function determines how the vector representations of all tokens are combined to form a single vector embedding. Table 1 shows the F1-score for the most common combination functions.

Table 1: Comparison of combination functions[1]

	F1 score					Support
	Mean	Max	Sum	First	Last	Support
NOUN	0.938	0.937	0.938	0.933	0.938	4135
PRON	0.993	0.993	0.993	0.992	0.992	2156
PUNCT	0.992	0.992	0.992	0.991	0.992	3096
VERB	0.967	0.965	0.966	0.965	0.965	2638
Macro Avg.	0.934	0.931	0.932	0.927	0.932	25082
W. avg.[2]	0.958	0.957	0.956	0.953	0.957	25082

[1] Complete table in appendix A, [2] weighted average

Before analysing the results, we explain the difference between each combination function. The “mean” combination function takes the element-wise average of all the tokens that conform the subword. Similarly, the “sum” and “max” take the sum and maximum of all the tokens that constitute the subword, respectively. Lastly, the “first” and “last” functions only consider either the first or last token of the subword. It is worth mentioning that in the English language, the first token of a subword is not generally helpful to determine the POS tag of that word, e.g. the token “word” could be follow by “ify”, “ification”, “iness”, etc.

As observed in Table 1, based on the average F1-score of the combination functions, there is not a significant performance difference between them (except for “first” combination function). To further analyze why the performance of different combination functions are similar, we delved into the specifics of the tokenization process performed on our dataset. First, we analyzed the training set and observed that in total, RoBERTa obtained 228,969 non-unique tokens and that only 14,433 of those correspond to subword tokens; this means that approximately 6% of the tokens are subword tokens, a small subset of the total training set. Furthermore, when words are split, they rarely contain more than 3 subwords, so there is not much difference between the resulting embedding vectors for different combination functions. Hence, due to the aforementioned observations, even though the performance difference between the combination functions is small, the mean combination function is the preferred option due to its higher weighted average F1-score and will be used for subsequent analysis.

4 Analysis of mistakes made by the “mean” contextual POS tagger

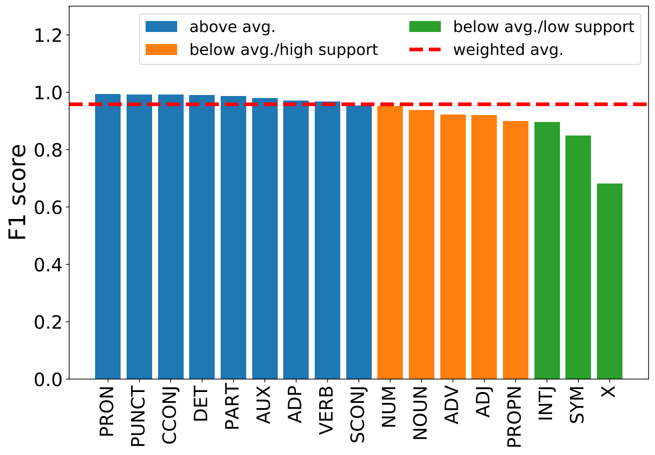

Figure 2: F1-score vs POS tags

Overall, the model’s performance is excellent (i.e. F1-score close to 1). There are some tags with a particularly low performance, but this is because they have a low support or are not common (e.g. tag X). Interestingly, the performance is below average for certain tags with high support. This can be observed in Figure 2. We explored the hypothesis that F1-score and support are highly correlated, but it is not the case (see A). This decrease in performance seems to be explained by syntactic ambiguity. Even powerful models such as RoBERTa can get confused by ambiguity in the context of the analyzed sentences. For instance, the model incorrectly predicts the tag for roman numbers (e.g. *“VI”*) as nouns. Similarly, the model seems to confound nouns with PROPN when the word starts with a capital letter, i.e.: “*Name* of specific Hibachi restaurant in Chicago?”, the word *Name* is incorrectly predicted as PROPN. Similar examples of misclassification for other tags can be seen in B.

5 Conclusions

As discussed, contextual POS taggers offer superior performance than static POS taggers and the baseline model. Additionally, the choice of combination function does not substantially affect the performance of the POS tagger, since the proportion of subwords to full-words is small, and each tokenized word is rarely comprised by more than three tokens. Furthermore, we observed that for several tags, below-average performance is usually caused by a low support. We also found that the F1-score for some POS tags with high support can be below average and in these cases, the problem’s root cause is syntactic ambiguity rather than low support. Future research ideas could be exploring new combination functions, such as a weighted average of the tokens.

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.

HuggingFace. 2021. Roberta.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

A Results tables for static and contextual word embeddings

Preliminary exercise

Table 2 shows the performance of the baseline model when evaluated on the test set.

Table 2: Baseline model

	Precision	Recall	F1-score	Support
NOUN	0.674	0.929	0.781	4136
PRON	0.968	0.933	0.950	2158
PUNCT	0.994	0.985	0.990	3098
VERB	0.884	0.809	0.845	2640
Macro Avg.	0.841	0.780	0.792	25098
W. avg.[1]	0.872	0.858	0.855	25098

[1] weighted average

Table 3 shows the performance of all the POS tags when using static word embeddings.

Table 3: POS tag performance for static word embedding

	Precision	Recall	F1-score	Support
ADJ	0.855	0.817	0.836	1782
ADP	0.867	0.883	0.875	2030
ADV	0.864	0.709	0.779	1147
AUX	0.900	0.885	0.893	1509
CCONJ	0.991	0.992	0.991	738
DET	0.955	0.975	0.965	1898
INTJ	0.933	0.692	0.794	120
NOUN	0.825	0.870	0.847	4136
NUM	0.887	0.723	0.796	541
PART	0.666	0.995	0.798	630
PRON	0.965	0.926	0.945	2158
PROPN	0.729	0.759	0.744	1985
PUNCT	0.994	0.983	0.988	3098
SCONJ	0.611	0.639	0.625	443
SYM	0.772	0.830	0.800	106
VERB	0.834	0.821	0.827	2640
X	0.231	0.022	0.040	137
Accuracy			0.867	25098
Macro Avg.	0.816	0.795	0.797	25082
W. avg.[1]	0.868	0.867	0.866	25082

[1] weighted average

Table 4 shows the performance comparison for all combination functions when using contextual word embeddings.

Table 4: Full comparison of combination functions

	F1 score					
	Mean	Max	Sum	First	Last	Support
ADJ	0.920	0.917	0.915	0.902	0.919	1782
ADP	0.971	0.973	0.972	0.972	0.972	2028
ADV	0.921	0.918	0.914	0.910	0.921	1147
AUX	0.980	0.980	0.979	0.978	0.979	1508
CCONJ	0.992	0.993	0.991	0.991	0.991	737
DET	0.989	0.989	0.988	0.987	0.989	1898
INTJ	0.896	0.887	0.886	0.895	0.894	118
NOUN	0.938	0.937	0.938	0.933	0.938	4135
NUM	0.952	0.950	0.947	0.946	0.949	540
PART	0.986	0.982	0.985	0.984	0.985	629
PRON	0.993	0.993	0.993	0.992	0.992	2156
PROPN	0.993	0.900	0.896	0.890	0.895	1984
PUNCT	0.992	0.992	0.992	0.991	0.992	3096
SCONJ	0.954	0.951	0.955	0.952	0.956	443
SYM	0.849	0.849	0.855	0.835	0.857	106
VERB	0.967	0.965	0.966	0.965	0.965	2638
X	0.681	0.654	0.664	0.638	0.647	137
Macro Avg.	0.934	0.931	0.932	0.927	0.932	25082
W. avg.[1]	0.958	0.957	0.956	0.953	0.957	25082

[1] weighted average

Figure 3 shows the F1-score and support for each tag (when evaluated in the test set). Additionally, the figure shows the best-fit linear regression. There seems to be no linear relationship between the variables, and the R2 value (0.174) suggests that the linear relationship between support and F1-score is weak.

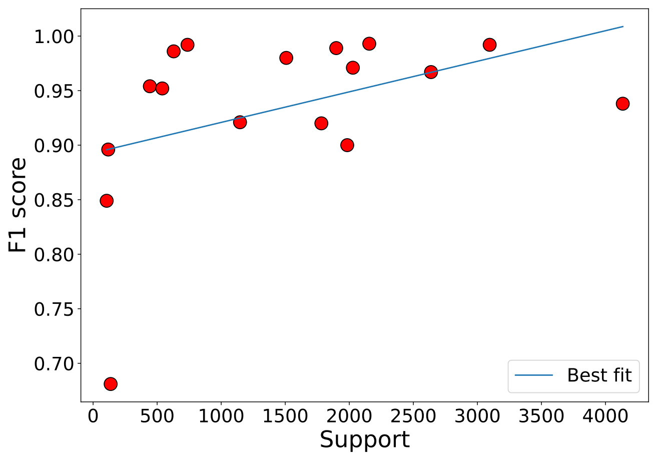

Figure 3: F1-score vs Support

B Prediction error examples

The tagged words are shown in bold font below.

- **ADV** classification error example:

  > ”that’s fine, i don’t want to see you **either**,
  > i just need to make a cd.”

  True tag: ADV, predicted: DET. The word **either** can be an adverb, a conjunction or a determiner. Although the model has enough context to classify it correctly in this example, it made a mistake here.

- **ADJ** classification error example:

  > ”Angry crowds chanted anti-American slogans in the western city of Falluja (pop. 256,000) as the security police killed in a **friendly** fire incident by US troops were buried on Saturday.”

  True tag: ADJ, predicted: ADV. The word **friendly** could classified as an adjective or as noun (e.g. ”England will play a friendly in France”), in this case the model mislabelled it as an adverb.

- **PROPN** classification error example:

  > ”They are certainly being nasty to the United Nations **Security** Council in connection with the anti-proliferation treaty.”

  True tag: PROPN, predicted: ADJ. The word **Security** *could* serve as an adjective, but in this particular context it is a proper noun and the model misclassifies it.