Composing Byte-Pair Encodings for Morphological Sequence Classification

Anonymous COLING submission

Abstract

This document contains the instructions for preparing a paper submitted to COLING-2020 or accepted for publication in its proceedings. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used for both papers submitted for review and for final versions of accepted papers. Authors are asked to conform to all the directions reported in this document.

1 Introduction

Things added to model:

- Adaptive learning rate
- Fine-tuning
- Dropouts
- Trained character representations (?)

Things to add:

- (results) More detailed accuracy for len(BPE)> 2
- (results) With fine-tuning/without fine-tuning
- (data) Data statistics: number of MSD-tags
- (data) Pick other datasets, try to match dataset sizes better (100k-150k train examples, turkish very small atm)

bla bla bla bullshit about transformer as an introduction. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long. Roughly this long.

The transformer models have diverged from the tokenization of classical natural language processing systems, favoring a statistical approach splitting a string of characters into (statistically) meaningful units. Rather than identifying complete words it aims to find sub-word units occurring significantly often. Typical approaches to composing tokens from sub-token units have focused on combining character n-grams [cite bojanowski], while other approaches have looked at splitting words into *roots* and *morphs* then combining them. Byte-pair take another approach by not specifically looking for either character n-grams or morphs, rather it aims to split a corpus $\mathcal{C}$ into N tokens, where N is user defined. [EXPAND]

In this paper we explore how to combine bpe-pair encodings in a transformer model to perform sequence classification. The goal in sequence classification is to assign a label to every token in a sentence. When we are using byte-pair encoding (or similar sub-token representations) we must then find some way of combining the units that build the word whose receive some label.

Previous work have only brushed the surface of this problem by reporting that few differences are found between different methods. In this paper we wish to explore this in further detail and try to identify what effect different methods have for the final performance of a model.

Task: In this paper we explore morphological sequence classification. Morphological tagging involves identifying a set of morphological features that a word posses, such as number, person, case and so on. Morphological features primarily depend on the affixes of words, but additional information is also provided by the words context.

Data: For the task we will use the Universal Dependencies dataset [6] annotated with the UniMorph schema [4]. We are interested both in the general effects of using different composition methods, and whether some methods favor languages with certain typology. To explore this, we take a sample of eight languages from the Universal Dependencies dataset, four languages that use *fusional* morphology, an affix may be indicative of one or more morphological features, and four languages that use *agglutinative* morphology, each affix is mapped to one and only one morphological feature.

The fusional languages we look at are Arabic, Czech, Polish and Spanish, and the agglutinative languages we look at are Finnish, Basque, Turkish and Estonian. The size of the dataset and the average number of BPE tokens per word are shown in Table 1.

Language	Typology	$\frac{BPE}{word}$	Train	Validation	Test
Arabic-PADT	Fusional	1.39	225494	28089	28801
Czech-CAC	Fusional	1.77	395043	50087	49253
Polish-LFG	Fusional	1.75	104730	13161	13076
Spanish-AnCora	Fusional	1.34	439925	55196	54449
Finnish-TDT	Agglutinative	1.98	161791	19876	20541
Basque-BDT	Agglutinative	1.79	97336	12206	11901
Turkish-IMST	Agglutinative	1.73	46417	5708	5734
Estonian-EDT	Agglutinative	1.86	346986	43434	43825

Table 1: Treebank statistics.

The fusional languages where chosen such that two of them (Czech and Polish) have a higher BPE per token ratio than the other two (Arabic and Spanish). We do this because one factor that impact composition method may be BPE per token ratio, and we want to exclude this from our analysis. Thus, by having both fusional and agglutinative languages with similar BPE per token ratio we can discard this as an explanatory factor.

2 Method

In this section we present the model used for sequence classification, the methods we use to compose BPE embeddings, and how we trained the model.

Transformer model For the task we use the XLMR [1] model[1]. XLMR is a masked language model based on the transformer (specifically, RoBERTa [3]) trained on data from 100 different languages, using a shared vocabulary. In this experiment we use XLMR_{base} model, it has 12 encoder layers, 16 (?) attention heads and use 768 dimensions for its hidden size.

2.1 Byte-pair combination methods

2.2 Model

For morphological tagging we use the XLMR base model with a classification module on top. The classification module is a LSTM followed by a two layer linear transformation.

The model we use to predict morphological features is simple. For each sentence we extract n BPE embeddings from $XLMR_{base}$ then align them to words. We then run all words who consist of more

[1] We use the huggingface implementation [CITE/LINK]

than one BPE embedding through a function f which combine the BPE embeddings. This produce one embedding per word which we concatenate with character features generated by a character LSTM. We then pass the BPE features concatenated with the character features to a LSTM to extracxt contextual features. We pass the LSTM outputs to a linear transformation layer that compute scores for each class in the output. We then apply softmax and compute the loss.

An outline of the model is presented in Figure 1, in the outline f represent the different methods we use to combine BPE embeddings.

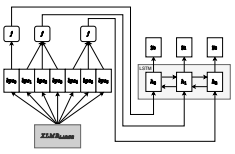


Figure 1: Procedure outline

2.2.1 BPE features

We obtain BPE features for a *token* by combining the BPE tokens that compose the token by combining their features using three different methods:

Sum: For the sum method, we take the sum for each dimension of all BPE embeddings. Thus, for dimension i we add all the values in all BPE embeddings $x_i^0+, ..., +x_i^n$:

$$x'_i = \sum_{j=1}^{n} x_i^j \tag{1}$$

Mean: In the mean method we calculate the sum and divide by the number of BPE embeddings in the word. Thus, for dimension i we calculate $\frac{x_i^0+,...,+x_i^n}{n}$:

$$x'_i = \frac{1}{n} \sum_{j=1}^{n} x_i^j \tag{2}$$

RNN: In this method we employ a bidirectional RNN to learn how to compose the BPE embeddings. For each multi-BPE token, we pass the sequence of BPE embeddings through an RNN and use the final output as the word representation.

2.2.2 XLMR features

When extracting features from XLMR we compute a weighted sum of the layer representations. This is inspired by previous work showing that the different layers in language models encode different kinds of information. That is, we create a random vector w with one element per layer, indicating the importance of the layer. We calculate the layerwise weighted sum for each bpe representation as follows:

$$x_i = \sum_{1}^{J} softmax(c)_j l_{ji} \tag{3}$$

2.2.3 Character features

In addition to layer attention we use a character LSTM to extract a word representation based on characters. The final representation we pass to the word-LSTM is the concatenation of the word representation based on BPE compositions and characters, $w_i = \text{concat}(f(bpe_0, ..., bpe_K), LSTM(c_0, ..., c_M))$

2.2.4 Label smoothing

Given that many of the languages have a large number of morphological tags we want to prevent the model from growing overconfident for certain classes. To address this we introduce label smoothing, that is, instead of the incorrect classes having 0 probability and the correct class 100% probability we let each of the incorrect classes have a probability of α. We assign probabilities to a target t by $(1-\alpha)t + \alpha/|C|$ where $|C|$ is the number of classes.

2.3 Training

When fine-tuning the model we freeze the XLMR parameters for the first epoch. When training the model we use cosine annealing learning rate with restarts every epoch, that is, the learning rate starts high then incrementally decrease to $1e-12$ during N steps, where N is the number of batches in an epoch.

Parameter	Value
Epochs	15
Batch size	4 / 32
Character representation size	128
Optimizer	Adam
Learning rate	0.001
Learning rate$_{xlmr}$	1e-06
Label smoothing	0.03

Table 2: Hyperparameters used for training the model. Slashed indicate the value of a parameter when we finetune or extract features.

3 Results

We present the accuracy of assigning morphological tags to all tokens in Table 3. Both when we finetune and only extract BPE weights we see that using an RNN to compose BPE tokens yields slightly better performance. In general, our results are lower than the reported State-of-the-Art.

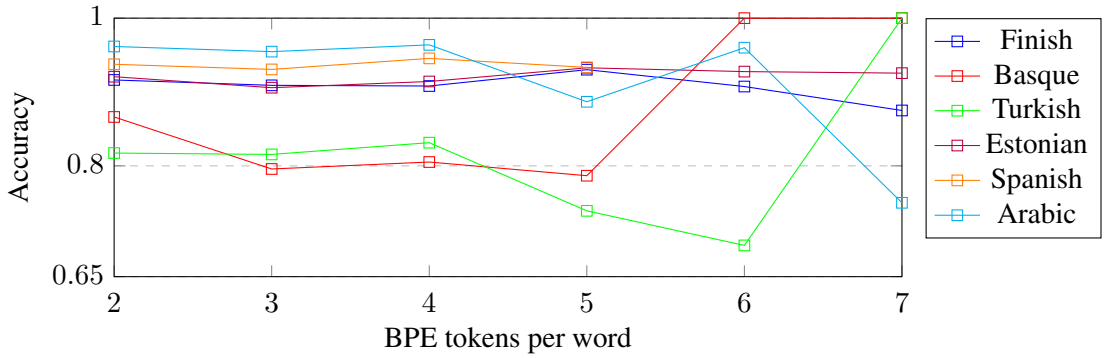
		Finetuning			Feature extraction		
Treebank	Baseline	Sum	Mean	RNN	Sum	Mean	RNN
Finnish-TDT	.751	.965	.963	**.967**	.930	.931	**.942**
Basque-BDT	.676	.910	.909	**.920**	.865	.865	**.888**
Turkish-IMST	.620	.898	.891	**.905**	.856	.849	**.866**
Estonian-EDT					.931	.934	**.939**
Spanish-AnCora	.842	.979	.979	**.980**	.968	.967	**.971**
Arabic-PADT	.770	.952	.953	**.954**	.941	.939	**.948**
Czech-CAC	.771				.944		
Polish-LFG					.907	.907	**.928**

Table 3: Accuracy for morphological tagging. We evaluate both when we finetune the XLMR model and when we only extract BPE embeddings weights.

However, the results in Table 3 also include tokens which are only composed of one BPE token. To better evaluate our composition methods we compute the accuracy for tokens which are composed of two or more BPE tokens. The results can be seen in Table 4.

	Method		
Treebank	Sum	Mean	RNN
Finnish-TDT	.893	.897	**.913**
Basque-BDT	.802	.803	**.840**
Turkish-IMST	.807	.796	**.816**
Estonian-EDT	.904	.908	**.916**
Spanish-AnCora	.952	.951	**.959**
Arabic-PADT	.927	.925	**.935**
Czech-CAC			
Polish-LFG	.834	.833	**.876**

Table 4: Accuracy for morphological tagging on all tokens that are composed of 2 or more BPE tokens.



4 Discussion

Analogously to character embeddings, the RNN method seem to provide stronger results than using either Sum or Mean for combining BPE tokens. We hypothesize that this is because Sum or Mean don't respect the ordering of elements within a word, that is, we don't relate the values of BPE_0 to $BPE_{1:N}$ with respect to the morphological tags.

5 Conclusions

As a general summary of our work, we highlight that language typology is not irrelevant when working with models that employ byte-pair encoding.

References

[1] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[4] Arya D McCarthy, Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. Marrying universal dependencies and universal morphology. *arXiv preprint arXiv:1810.06743*, 2018.

[5] Arya D McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian J Mielke, Jeffrey Heinz, et al. The sigmorphon 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. *arXiv preprint arXiv:1910.11493*, 2019.

[6] Joakim Nivre et al. Universal dependencies 2.2. 2018. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.