Composing Byte-Pair Encodings for Morphological Sequence Classification

Adam Ek,
Centre for Linguistic Theory and Studies in Probability,
Department of Philosophy, Linguistics and Theory of Science,
University of Gothenburg,
adam.ek@gu.se

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)

In this project we explore the transformer model applied to sequence classification. In sequence classification a dataset is split into words (or tokens) and each word is assigned a label. Neural model can predict the label of a word by assigning it a representation and passing it through the network. However, the transformer model use byte-pair encoding (BPE) [7], or some variation of it [2], to tokenize text. BPE tokens are sub-strings that occur significantly often in the corpus, and does not correspond directly to words. For example, the word ”scientifically” may be composed of the the BPE tokens: [”scient”, ”ifical”, ”ly”]. In sequence classification, we need to assign a label to the word ”scientifically”. But the word is now composed of three BPE embeddings, so to predict a label for the word we need to compose all of the BPE embeddings into one representation.

In this project is to explore six different methods of creating word representations from multiple BPE embeddings.

Task: We explore combination of BPE embeddings with the task of Morphological-Tagging-in-Context [5]. The task is simply to predict the morphological features (such as case, number person, ...) of each word (as defined in the dataset) given the sentence which the word occur in.

Data: For the task we will use the Universal Dependencies dataset [6] annotated with the UniMorph schema [4]. We look at 6 different languages: Arabic, Basque, Finnish, Czech, Spanish and Turkish. The language typology, ratio of BPE-tokens per word and training/development and test data can be found 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.

Base model For the task we will 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_{LARGE}, the model have 12 encoder layers, 16 attention heads and use 1024 dimensions for its hidden size.

2 Method

In this section we present different methods of composing BPE embeddings into word representations and the structure of our neural network. We align BPE tokens to words by a simple string matching algorithm.

2.1 Byte-pair combination methods

Sum: For the sum method, we simply 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 $\frac{x_i^0+,...,+x_i^n}{n}$:

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

Max: In the max method we take the most activated value of dimension i over all BPE embeddings $x^0...x^n$ in the word.

$$x'_i = argmax(x_i^0, ...x_i^n) \tag{3}$$

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.

[1]We use the fairseq implementation: `https://github.com/pytorch/fairseq/tree/master/examples/xlmr`

Att1: In the first attention method we use self-attention [8] with one linear transformation. Like in the transformer we use three copies of the BPE sequence as inputs and transform one of them using a linear layer. We then calculate pairwise scores and apply softmax. We then use the softmax to create a weighted sum for the word representation.

$$x = softmax(\frac{q(kW_k)^\intercal}{\sqrt{d}}) \cdot V \tag{4}$$

Att3: Like in the first attention method we use self-attentions. However, in the second attention approach we use three linear transformations, one for each copy of the input.

$$x = softmax(\frac{qW_q kW_k^\intercal}{\sqrt{d}}) \cdot vW_v \tag{5}$$

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.

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 the BPE representation as follows:

$$x = \sum_{1}^{N} c_n l_n \tag{6}$$

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.

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.

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

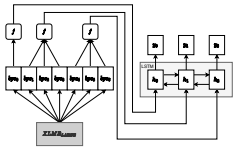

Figure 1: Procedure outline

The model we use to predict morphological features is simple. For each sentence we extract n BPE embeddings from $XLMR_{LARGE}$ then align them to words. We then run all words who consist of more than one BPE embedding through a function f which combine the BPE embeddings. This produce one embedding per word which we pass to a bidirectional LSTM to extract 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.

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							
Spanish-AnCora	.842	.979	.979	**.980**	.968	.967	.971
Arabic-PADT	.770	.952	.953	**.954**	.941	.939	**.948**
Czech-CAC	.771						
Polish-LFG							

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	.927	.928	.939
Basque-BDT	.864	.863	.886
Turkish-IMST	.853	.846	.863
Estonian-EDT			
Spanish-AnCora	.967	.966	.970
Arabic-PADT	.937	.935	.945
Czech-CAC			
Polish-LFG	.906	.906	.927

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

4 Discussion

We can observe an interesting trend for the treebank samples. For the languages with agglutinative morphology[2] the average increase using RNN compared to Mean is 4.3%. For the fusional[3] languages, the difference between RNN and Mean is only 1.1%.

Interestingly, this observation seem task-dependant rather than BPE dependant. We can see in Table 1 that Czech is a fusional language, but with a BPE-to-word ratio (1.77) comparable to the agglutinative languages. But, Czech also exhibit the pattern of benefiting less from RNN than from Mean, indicating that fusional languages[4] regardless of BPE-to-word ratio is less affected by the composition method than agglutinative languages.

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.

[2] 1-to-1 correspondence between morphemes and grammatical/syntactic features.

[3] 1-to-many correspondence between morphemes and grammatical/syntactic features.

[4] In our language/treebank sample.