Composing Byte-Pair Encodings for Morphological Sequence Classification

Adam Ek and **Jean-Philippe Bernardy**
Centre for Linguistic Theory and Studies in Probability,
Department of Philosophy, Linguistics and Theory of Science,
University of Gothenburg,
{adam.ek, jean-philippe.bernardy}@gu.se

Abstract

Byte-pair encodings is a method for splitting a word into sub-word tokens, a language model then assigns contextual representations separately to each of these tokens. In this paper, we evaluate four different methods of composing such sub-word representations into word representations. We evaluate the methods on morphological sequence classification, the task of predicting grammatical features of a word. Our experiments reveal that using an RNN to compute word representations is consistently more effective than the other methods tested across a sample of eight languages with different typology and varying numbers of byte-pair tokens per word.

1 Introduction

After its introduction, the Transformer model (Vaswani et al., 2017) has emerged as the dominant architecture for statistical language models, displacing recurrent neural networks, in particular, the LSTM and its variants. The Transformer owes its success to several factors, including the availability of pretrained models, which effectively yield rich contextual word embeddings. Such embeddings can be used as is (for so-called *feature extraction*), or the pre-trained models can be finetuned to specific tasks.

At the same time as Transformer models became popular, the tokenization of natural language texts have shifted away from methods explicitly oriented towards words or morphemes. Rather, statistical approaches are favoured: strings of characters are split into units which are not necessarily meaningful linguistically, but rather have statistically balanced frequencies. For example, the word "scientifically" may be composed of the tokens: "scient", "ifical", "ly" — here the central token does not correspond to a morpheme. That is, rather than identifying complete words or morphemes, one aims to find relatively large sub-word units occurring significantly often, while maximizing the coverage of the corpus (the presence of the "out of vocabulary" token is minimized). Approaches for composing words from sub-word units have focused on combining character n-grams (Bojanowski et al., 2017), while other approaches have looked at splitting words into *roots* and *morphemes* (El Kholy and Habash, 2012; Chaudhary et al., 2018; Xu and Liu, 2017), and then combining them.

In this paper, we consider Byte-Pair Encodings (BPE) (Sennrich et al., 2015). BPE has been popularized by its usage in translation and the BERT Transformer model (Devlin et al., 2018). The BPE algorithm does not specifically look for either character n-grams or morphs, but rather it aims at splitting a corpus $\mathcal{C}$ into N tokens, where N is user defined. Even though BPE is not grounded in morphosyntactic theory, the characteristics of the sub-word units generated by BPE will be directly influenced by morphosyntactic patterns in a language. In particular, it is reasonable to expect that the statistical characteristics of BPE to be different between languages with different typologies. One issue with this tokenization scheme is that models based on BPE provide vector representations for the BPE tokens (which we call *token embeddings* from now on), while one is typically interested in representations for the semantically meaningful units in the original texts, *words*. In sum, one wants to combine *token embeddings* into *word embeddings*.

Our main goal is to explore how to best combine token embeddings in the context of sequence classification on words, that is, the task of assigning a label to every word in a sentence. Coming back to our example, we must combine the token embeddings assigned to the BPE tokens "scient", "ifical" and "ly" to form a word representation of "scientifically" (as a vector) which we can then assign a label to.

Language	Typology	$\frac{\text{BPE}}{\text{word}}$	Tags	Train	Validation	Test
Basque-BDT	Agglutinative	1.79	919	97336	12206	11901
Finnish-TDT	Agglutinative	1.98	591	161791	19876	20541
Turkish-IMST	Agglutinative	1.73	1056	46417	5708	5734
Estonian-EDT	Agglutinative	1.86	512	346986	43434	43825
Spanish-AnCora	Fusional	1.25	177	439925	55196	54449
Arabic-PADT	Fusional	1.39	300	225494	28089	28801
Czech-CAC	Fusional	1.77	990	395043	50087	49253
Polish-LFG	Fusional	1.75	634	104730	13161	13076

Table 1: Treebank statistics showing the language typology, average number of BPE tokens per word, the number of (composite) morphological tags and the size of the datasets in terms of words.

To our knowledge, this is a little-studied problem. For the original BERT model Devlin et al. (2018) simply state that for named entity recognition the first sub-word token is used as the word representation. For morphological sequence classification Kondratyuk (2019); Kondratyuk and Straka (2019) report that only small differences in performance were found between averaging, taking the maximum value or first sub-word token. In this paper we explore the problem in further detail and identify the effect that different methods have on the final performance of a model. Additionally, with the increased interest in multilingual NLP it becomes important to explore how different computational methods perform cross-linguistically. That is, because languages are different morphosyntactically, one can expect various computational methods not to be uniformly effective.

2 Task

To investigate composition methods for token embeddings we focus on the task of morphological sequence classification. The task is to assign a tag to a word that represent its grammatical features, such as gender, number and so on. In addition to the word-form, the system can use information from context words as cues. While the grammatical features primarily are given by the word-form, useful information is also found in the context.

Thus, we have to identify k different tags for a word, each with C_i possible classes, making the task a multi-class classification problem. We simplify the classification problem by combining the different tags into a composite tag with up to $\prod_i^k C_i$ classes (instead of making k separate predictions). This task is suitable for our goal as the output space is large, ranging from 100 to 1000 possible tags for a word, depending on the grammatical features present in the language[1], and is directly linked to the affixes in the word-form. A system must efficiently encode information about the structure of the target words as well as the context words to be able to predict the correct grammatical features.

3 Data

For both training and testing data, we use the Universal Dependencies dataset (Nivre et al., 2018) annotated with the UniMorph schema (McCarthy et al., 2018). We are mainly interested in how the accuracy is influenced by different composition methods, but also consider the type of morphology a language uses as a factor in this task. With this in mind, we consider both languages that use agglutinative morphology where each morpheme is mapped to one and only one grammatical feature, and languages that use fusional morphology where a morpheme can be mapped to one or more grammatical features. The fusional languages that we consider are Arabic, Czech, Polish and Spanish, and the agglutinative languages that we consider are Finnish, Basque, Turkish, and Estonian. We show the size, the average number of BPE tokens per word, and the number of morphological tags for each treebank in Table 1.

The fusional languages were chosen such that two of them (Czech and Polish) have a higher BPE per word ratio than the other two (Arabic and Spanish). We make this choice because one factor that

[1]For practical reasons, we only consider tag combinations observed in the dataset

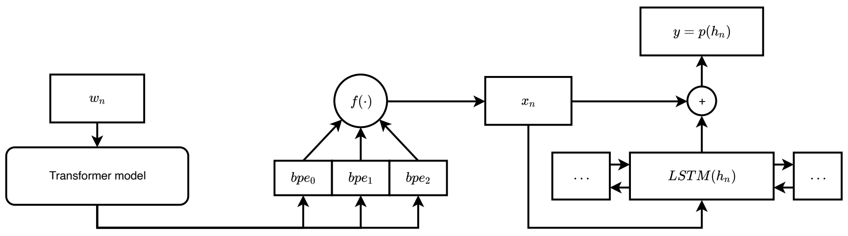

Figure 1: Model outline for one input. A word w_n is tokenized into k BPE tokens. The Transformer model produces one embedding per token per layer. We then calculate a weighted sum over the layers to obtain one representation per token. The resulting token embeddings are then passed to a composition function f that combines the k different token embeddings into a word embedding. The word embedding is then passed to an LSTM followed by a dense prediction layer.

impacts the accuracy obtained by a composition method may be the BPE per word ratio. By having both fusional and agglutinative languages with similar BPE per word ratio we can take this variable into account properly in our analysis.

4 Method

In this section we present the model used for morphological sequence classification, the methods that we use to compose token embeddings, and how the model is trained. [2]

4.1 Model

Our model is composed of three components, each of them detailed below. First, the input sequence of BPE tokens is fed to a Transformer model, which yields a contextual vector representation for each BPE token. The contextual information here is the surrounding BPE tokens in the sentence. Then, the token embeddings are combined using a composition module, which we vary for the purpose of evaluating each variant. This component yields one embedding per original word. Then we pass the word embeddings through a bidirectional LSTM, which is followed by two dense layers with GELU (Hendrycks and Gimpel, 2016) activation. These dense layers act on each word embedding separately (but share parameters across words). An outline of the model is presented in Figure 1, where f represents the different methods we use to combine token embeddings.

4.1.1 Underlying Transformer Model

To extract a embeddings for each BPE token, we use the XLM-RoBERTa (Conneau et al., 2019) model[3]. XLM-R is a masked language model based on the Transformer, specifically RoBERTa (Liu et al., 2019b), and trained on data from 100 different languages, using a shared vocabulary of 250000 BPE tokens. All the languages that we test are included in the XLM-R model. In this experiment we use the XLM-R$_{\text{base}}$ model with 250M parameters. It has 12 encoder layers, 12 attention heads and use 768 dimensions for its hidden size.

4.1.2 Feature extraction

The XLM-R model uses 12 layers to compute a vector representation for a BPE token. It has been shown in previous research (Kondratyuk and Straka, 2019; Raganato et al., 2018; Liu et al., 2019a) that the different layers of the Transformer model encode different types of information.

To take advantage of this variety, we compute token embeddings as a weighted sum of the layer representation (Kondratyuk and Straka, 2019), using a weight vector w, of size l, where l is the number of layers in the Transformer model. The weight vector w is initialized from a normal distribution of mean

[2]Our code is available at: `https://github.com/adamlek/ud-morphological-tagging`

[3]We use the huggingface implementation `https://huggingface.co/transformers/model_doc/xlmroberta.html`

0 and standard deviation 1. If r_{ji} is the layer representation at layer j and token position i, we calculate the weighted sum as follows:

$$x_i = \sum_{j=1}^{l} \mathsf{softmax}(w)_j r_{ji} \tag{1}$$

Consequently, in end-to-end training, the optimiser will find a weight for extracting information from each layer ($\mathsf{softmax}(w)_j$) which maximizes performance.

4.1.3 Composition of BPE token embeddings

The weighted sum yields a token embedding for each BPE token. We proceed to combine them into words as they appear in the data. The model that we use to combine token embeddings is as follows. For each sentence we extract n token embeddings x^0 to x^{n-1} from XLM-R$_{\text{base}}$, and then align them to words. We then pass all token embeddings in a word to a function f which combines the tokens into a word embedding.

We consider four methods for composing token embeddings: taking the first token embedding, summation, averaging, and using an RNN. Taking the first token embedding, summation and averaging have been used in previous work (Sachan et al., 2020; Kondratyuk, 2019; Devlin et al., 2018), but using an RNN has not been explored before to our knowledge.

First: The first method is the standard one used by Devlin et al. (2018), which is to use the first token embedding in a word.

Sum: For the Sum method, we use an element-wise sum. That is, we calculate the vector sum of the token embeddings. Assuming that we have T token embeddings in a word X (the word is a matrix of aligned token embeddings before the operation), for dimension i we calculate the word embedding by summing the token embeddings:

$$f(X)_i = \sum_{j=1}^{T} x_i^j \tag{2}$$

Mean: In the mean method we calculate the sum as above and divide by the number of BPE tokens in the word. Thus, for word X we calculate the word embedding by averaging over the sum:

$$f(X)_i = \frac{1}{T} \sum_{j=1}^{T} x_i^j \tag{3}$$

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

4.1.4 Word-level features and classification

The above methods of composing BPE tokens produce one contextual embedding per word. We then pass the word embeddings through an LSTM to take into account the word contexts. While the BPE token embeddings are already contextual, they are conditioned on the BPE token context, not word context. We pass the hidden states for each word to a residual connection with the pre-LSTM representation. We then pass this to two dense layers with GELU activation followed by a dense layer that computes class-scores for each word. We then use a softmax layer to assign probabilities and compute the loss accordingly.

Commonly, systems analyzing morphology use character embeddings as an additional source of information. We opted not to include character embeddings because this would obfuscate the effect of the composition method and may mask some of the effects of the different methods.

4.1.5 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 issue we introduce label smoothing (Szegedy et al., 2016), 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 small probability.

Let α be our smoothing value, in our model we follow (Kondratyuk and Straka, 2019) and use $\alpha = 0.03$, and C the number of classes, then given a one-hot encoded target vector t of size C, we calculate the smoothed probabilities as:

$$t_{smooth} = (1 - \alpha)t + \frac{\alpha}{C} \tag{4}$$

In words, we remove α from the correct class then distribute α uniformly among all classes.

4.2 Training

In our experiments we consider two possible training regimes. In the first regime we finetune the XLM-R model's parameters, in the second we only extract weights for BPE tokens, that is, we use the model as a feature extractor. In all cases, we use end-to-end training.

When finetuning the model we freeze the XLM-R parameters for the first epoch, effectively not finetuning at first. When training the model we use a cosine annealing learning rate (Loshchilov and Hutter, 2016) with restarts every epoch, that is, the learning rate starts high then incrementally decreases to $1e{-}12$ over N steps, where N is the number of batches in an epoch. We use the Adam optimizer with standard parameters, with a learning rate of 0.001 for layer importance parameter (w in Section 4.1.2), the parameters of the Word-LSTM, of the classification layer, and of the BPE-combination module (when an RNN is used). For the Transformer parameters, we use a lower learning rate of $1e{-}6$. We summarize the hyperparameters used in Table 2.

Parameter	Value
Epochs	15
Batch size	4 / 32
Word LSTM size	768
Linear transform size	1536
Optimizer	Adam
Learning rate	0.001
Learning rate$_{xlmr}$	$1e{-}6$
Weight decay	0.05
Label smoothing	0.03

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

As an additional regularization in addition to weight decay and adaptive learning rate, we use dropout throughout the model. Generally, we apply dropout before some feature is computed. Initial experiments revealed that a high dropout yielded the best results. We summarize the dropout used as: We replace 20 percent of the BPE tokens with `<UNK>`. Then, we compute a weighted sum of the layer representations, to regularize this operation we apply dropout on layer representations with a probability of 0.1, that is we set all representations in the layer to 0. We then combine the token embeddings into word embeddings and apply a dropout of 0.4%, and pass these into the Word-LSTM. Before the contextualized representation is passed to the classification layer, we apply a dropout of 0.4%.

5 Results

Even though our aim is to compare the relative performance of various BPE-combination methods rather than to improve on the state of the art in absolute terms, we compare our results against the baseline reported by McCarthy et al. (2019). This comparison serves the purpose of checking that our system is generally sound. In particular, the actual state of the art, as reported by McCarthy et al. (2019); Kondratyuk (2019), uses treebank concatenation or other methods to incorporate information from all treebanks available in a language, which means that results are not reported on a strict per-treebank basis and thus our numbers are not directly comparable. We report the accuracy of prediction morphological tags for each of our composition methods, and for our two training regimes in Table 3.

Our system performs better than the baseline. As a general trend we see that the RNN method tends to perform better than all other tested methods. This trend is consistent across both language families (agglutinative and fusional) and training regimes showing that, while the advantage of the RNN is small,

		Finetuning				Feature extraction			
Treebank	Baseline	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.676	.857	.884	.877	**.901**	.759	.789	.780	**.834**
Finnish-TDT	.751	.961	.958	.960	**.965**	.853	.856	.847	**.899**
Turkish-IMST	.620	.848	.859	.855	**.884**	.742	.741	.735	**.775**
Estonian-EDT	.740	.956	.955	.955	**.961**	.855	.856	.853	**.901**
Spanish-AnCora	.842	.977	.977	.977	**.979**	.951	.954	.952	**.962**
Arabic-PADT	.770	.946	.946	.947	**.951**	.920	.923	.920	**.936**
Czech-CAC	.771	.968	.968	.968	**.975**	.863	.887	.881	**.924**
Polish-LFG	.657	.956	.953	.953	**.959**	.828	.844	.840	**.878**
Average	.728	.933	.937	.936	**.946**	.846	.856	.851	**.888**

Table 3: Accuracy for morphological tagging. We show scores both for finetuning the XLM-R model and extracting features.

	Finetuning				Feature extraction			
Treebank	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.739	.802	.790	**.835**	.657	.715	.703	**.774**
Finnish-TDT	.940	.946	.946	**.952**	.780	.805	.794	**.861**
Turkish-IMST	.730	.780	.778	**.818**	.653	.683	.664	**.711**
Estonian-EDT	.938	.939	.939	**.949**	.779	.805	.803	**.868**
Spanish-AnCora	.956	.961	.959	**.964**	.922	.937	.930	**.947**
Arabic-PADT	.889	.896	.898	**.907**	.902	.909	.906	**.923**
Czech-CAC	.940	.947	.947	**.959**	.786	.849	.840	**.900**
Polish-LFG	.917	.920	.918	**.927**	.696	.761	.752	**.812**
Average	.881	.899	.897	**.913**	.772	.808	.799	**.849**

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

it occurs consistently. In general we find that finetuning yields higher accuracy than plain feature extraction, on average the difference is about 5.8 percentage points. This difference is to be expected when finetuning has 250M more parameters tuned to the task than the feature extraction.

Focusing on the finetuning regime only, we see the largest benefits of the RNN method for Basque with an increased performance of 3.25 points, and 2.7 points for Turkish over using mean or averaging. The First method for Basque and Turkish performs worse with a decrease of 4.4 percentage points for Basque and 3.6 points for Turkish compared to the RNN method. In the bare features extraction regime, we see a larger benefit for the RNN, of 3.7 percentage points (Turkish) and 4.95 points (Basque). Again, this is not unexpected: When finetuning the error rate is smaller, and therefore there is a smaller margin for a subsequent phase to yield and improvement.

Table 3 reports average accuracy for every word, including those which are only composed of a single BPE token. To highlight the strengths and weaknesses of each composition method, we also compute the accuracy for longer words only (composed of two or more BPE tokens). The results can be seen in Table 4. We see the same trend for accuracy on words that are composed of two or more BPE tokens, as in the overall accuracy, where the RNN outperforms all other methods. We can also see that the average increase in accuracy when using an RNN is larger. This holds both when finetuning or extracting bare features. Given that the number of BPE tokens per word varies in the different languages, we also look at the accuracy of the different methods given the number of BPE tokens. We show per-language performance with the different methods in Figure 2.

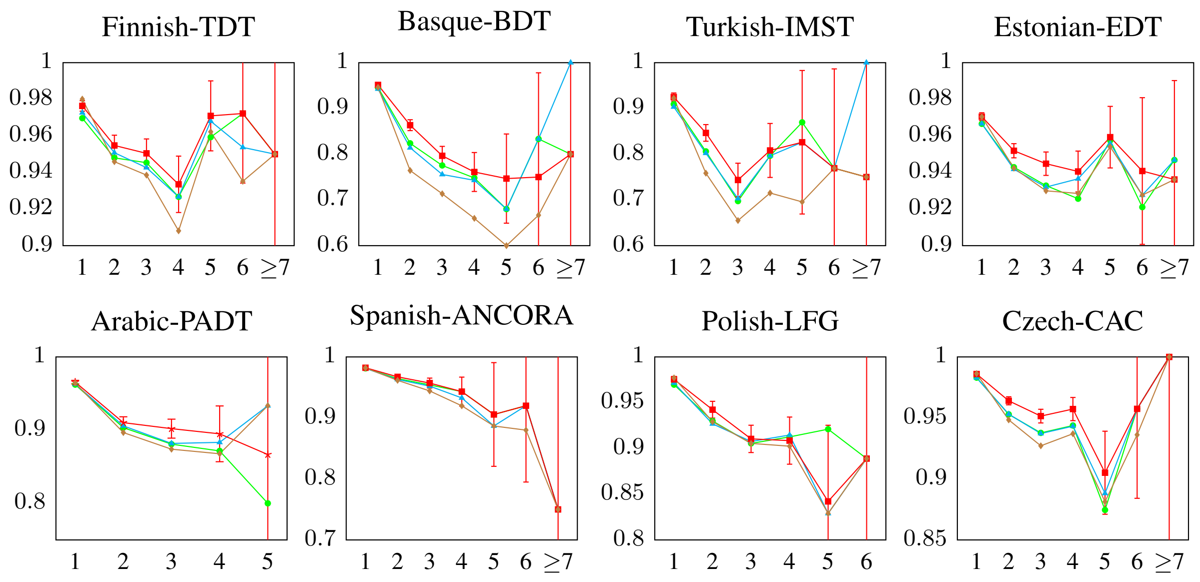

Figure 2: Per-language accuracy on tokens with different numbers of BPE components, for the finetuning training regime. The last data point on the x-axis refers to all tokens composed of seven or more BPE tokens. We indicate the method by encoding First as brown, summation as green, averaging as blue and RNN as red. The accuracy is given on the y-axis. We show the Agresti-Coull approximation of a 95%-confidence interval for the RNN method (Agresti and Coull, 1998). We do not show the intervals for other methods to avoid excessive clutter.

6 Discussion

For predicting morphological features, the RNN method is more effective than the other proposed methods (summing, averaging or taking the first BPE token). This holds regardless of training regime (finetuning versus feature extraction) and across languages with different BPE per word ratios.

As we see it, the advantage of the RNN over commutative methods (Sum, Mean) and taking the first BPE token is that it can take the order of elements into account. In broad terms, information about the order of elements in morphology allows a system to determine what is a stem, prefix, or suffix. Thus allowing a model to collect more predictive information from token embeddings.

We can suspect that the average BPE per word ratio in a language affects the performance of the composition method used. To further control this variable, in Figure 3 we plot the average number of BPE tokens per word in each language (x-axis), and compare this average against the gain in accuracy yielded by using the RNN method over summation (y-axis). For finetuning we see that in general the average number of BPE tokens does not matter that much. The two cases where it does matter is for Turkish and Basque, where we see a substantial improvement of about 3 percentage points. We note however that these are also the languages with the lowest amount of training data. For the other languages the improvements lie in the range .6 to 1.2 percentage points. This indicates that when finetuning, the model can provide information that allows commutative methods to properly compose BPE tokens. However, looking at bare feature extraction we see that there is a larger gap between the low BPE-ratio and the high BPE-ratio languages.

Our sample of languages contain both fusional and agglutinative languages, and the typology does not appear to have an effect in our experiments. We see about the same trends for the fusional languages with a high BPE per word ratio as the agglutinative languages.

6.1 First method

The idea behind the First method is that the Transformer is sufficiently powerful to pool the relevant information into the first BPE token embedding. However, our experiments reveal that it is less efficient than any other method we tested for morphological sequence classification across languages. We see in

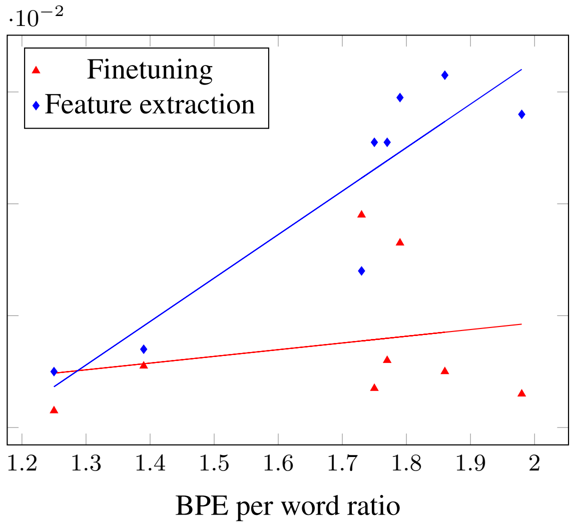

Figure 3: The difference in accuracy between summation and RNN plotted against average number of BPE tokens per word in all languages, with a linear regression line.

Table 3 that the method is, on average, .4 and 1 percentage points lower than the next lowest scoring method for finetuning and feature extraction respectively. This effect is further enhanced when we consider the accuracy of words composed of more than two BPE tokens in Table 4, where the difference is 1.6 and 2.7 points, compared against the next lowest scoring method, for finetuning and feature extraction respectively. When we compare the performance against the RNN this difference only increases, showing a gain of 3.2 percentage points and 7.7 points for finetuning and feature extraction respectively.

While the First method may be effective, primarily because of the expressivity of the Transformer architecture, the method forces the model to push the predictive information of several BPE token embeddings into the first one. This puts an additional burden on the Transformer model, and we believe that this is the reason for the performance degradation which we observe. Besides, putting this burden on the model is not necessary: pooling information from several BPE embeddings can be done effectively using additional layers.

6.2 Sum and Mean

When we consider the commutative methods of combining token embeddings, summation or averaging, we see no clear advantage for either of them over the other one, when doing finetuning. However, when extracting features only we see hints that summation is more effective than averaging. For feature extraction, summation is .5 percentage points better than averaging, and words composed of two or more BPE tokens exhibit an advantage of .9 point for summation.

This discrepancy suggests that by averaging, we are removing some predictive information from the pretrained BPE token embeddings, that is, by reducing the values in the token embeddings uniformly across a sequence of token embeddings we lose useful information. We believe that some token embeddings contain more predictive information than others, and by summing them we retain all the information. But when we finetune, the difference between summing and averaging almost disappears: the model appears to learn how to distribute the information uniformly across the token embeddings that compose a word and is thus able to retain the information better. Interestingly, the model learns to distribute the information across multiple BPE token embeddings more efficiently than pushing the information into the first token. This is shown by the large difference in accuracy between finetuning and feature extraction for the First and averaging method.

6.3 Parameterization of First, Sum and Mean

One question that arises when looking at Figure 2, specifically considering the performance on words composed of only one BPE token is the following: can the superiority of the RNNs be attributed to

		Finetuning				Feature extraction			
Treebank	Baseline	First	Sum	Mean	RNN	First	Sum	Mean	RNN
Basque-BDT	.676	.864	.894	.890	**.901**	.772	.793	.794	**.834**
Finnish-TDT	.751	.958	.959	.961	**.965**	.857	.856	.855	**.899**
Turkish-IMST	.620	.850	.875	.867	**.884**	.742	.722	.729	**.775**
Estonian-EDT	.740	.956	.958	.958	**.961**	.865	.856	.853	**.901**
Spanish-AnCora	.842	.978	.977	.978	**.979**	.953	.954	.952	**.962**
Arabic-PADT	.770	.949	.945	.947	**.951**	.925	.923	.920	**.936**
Czech-CAC	.771	.969	.972	.972	**.975**	.873	.887	.881	**.924**
Polish-LFG	.657	.957	.953	.955	**.959**	.832	.844	.840	**.878**
Average	.728	.935	.942	.941	**.946**	.852	.854	.853	**.888**

Table 5: The accuracy of morphological tagging when we parameterize the First, Sum and Mean method with a non-linear transformation layer.

its ability to take context into account, or simply to containing more parameters and extra layers? We would expect that for the words with only one BPE token, the performance of the model would be the same for all methods. For practical reasons, we push all word embeddings through an RNN, effectively doing a non-linear transformation with tanh activations on the words composed of only one BPE token. Typically, the difference in accuracy between various methods for one-BPE-token words is small (barely visible in Figure 2). But for example in Finnish, we see a larger difference. Although in general if we perform better on longer words consisting of BPE tokens that also appear as words in the data, we could also expect the performance to be better for words of BPE length one, because we will have more accurate representations of the contextual words.

We test this hypothesis by parameterizing the First, Sum, and Mean method. Essentially, we need to increase the capabilities of these methods. This is done by passing all BPE token embeddings through a non-linear transformation with ReLU activation before we compute the Sum, Mean, or select the first BPE-token. Our experiment, whose results are shown in Table 5, shows that while adding parameters to the First, Sum, and Mean method generally improve their performance slightly, ranging between a change of -0.2 and $+0.6$ percentage points, but their performance never exceeds that of the RNN method.

7 Conclusions and Future Work

In conclusion, our results indicate that using an RNN to compose word representations from token representations, obtained from a large Transformer model, is more efficient than two commutative methods, summing and averaging, and also more effective than letting a Transformer model automatically pool the predictive word-level information into the first BPE token embedding. We show this for the task of morphological sequence classification, in eight different languages with varying morphology and word-lengths in term of BPE tokens, as well as for two training regimes, finetuning and feature extraction.

In future work, we want to continue experimenting with the different BPE token embedding composition methods, specifically looking at more complex syntactic and semantic tasks, such as dependency and/or constituency parsing, semantic role labeling, named entity recognition, and natural language inference. We also wish to run our experiments on the hundreds of available UD treebanks to improve the robustness of our results.

Acknowledgments

The research reported in this paper was supported by grant 2014-39 from the Swedish Research Council, which funds the Centre for Linguistic Theory and Studies in Probability (CLASP) in the Department of Philosophy, Linguistics, and Theory of Science at the University of Gothenburg.

References

Alan Agresti and Brent A Coull. 1998. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R Mortensen, and Jaime G Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3285–3295.

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

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

Ahmed El Kholy and Nizar Habash. 2012. Orthographic and morphological processing for English–Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Dan Kondratyuk. 2019. Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 12–18.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094.

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

Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Arya D McCarthy, Miikka Silfverberg, Ryan Cotterell, Mans Hulden, and David Yarowsky. 2018. Marrying universal dependencies and universal morphology. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 91–101.

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. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh

Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Alessandro Raganato, Jörg Tiedemann, et al. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. The Association for Computational Linguistics.

Devendra Singh Sachan, Yuhao Zhang, Peng Qi, and William Hamilton. 2020. Do syntax trees help pre-trained transformers extract information? *arXiv preprint arXiv:2008.09084*.

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

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

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

Yang Xu and Jiawei Liu. 2017. Implicitly incorporating morphological information into word embedding. *arXiv preprint arXiv:1701.02481*.