Composing Byte-Pair Encodings for Morphological Sequence Classification

Adam Ek, Jean-Philippe Bernardy
{adam.ek, jean-philippe.bernardy}@gu.se

University of Gothenburg
CLASP

Introduction

- In recent years, we see transformers applied to almost all NLP tasks
- The transformer also introduced sub-word tokenization (BPE tokenization) as a "standard tool"
- In this paper we explore how to create word representations from sub-word representations

Byte-Pair Encodings

- Create a vocabulary by splitting a set of strings (sentences) into N tokens, such that we can represent all strings
- The items in the vocabulary won't correspond to traditional linguistic units
- Thus, when doing some lexical task with the vocabulary, we assign *embeddings* to sub-word tokens
- The problem that arises is the following: how do we combine the *token embeddings* into *word embeddings* so we can analyze lexical units?
- $f([\boldsymbol{e}_{scient}, \boldsymbol{e}_{ifica}, \boldsymbol{e}_{lly}]) = \boldsymbol{e}_{scientifically}$

Morphological Sequence Classification

- Morphological sequence classification is the task of predicting grammatical features of a word
- In the task, we are given a *sentence* where we need to predict the grammatical features of each word

Predicting grammatical features

- Grammatical features are primarily given by the *morphemes* in a word, so to predict grammatical features we must obtain information from all BPE tokens.

she	loves	giraffes
3;FEM;NOM;PRO;SG	3;FIN;IND;PRS;SG;V	N;PL

Table: Example from English-EWT.

- But, to some extent it's all about memorization:
- I (1;NOM;PRO;SG) vs We (1;NOM;PRO;PL)
- was (3;FIN;IND;PST;SG;V) vs is (3;FIN;IND;PRS;SG;V)
- This also applies to irregular verbs

Polish feminine nouns

	Hard declension		Soft declension	
	Singular	**Plural**	**Singular**	**Plural**
Nominative	mapa	mapy	granica	granice
Accusative	mapę	mapy	granicę	granice
Genitive	mapy	map	granicy	granic
Locative	mapie	mapach	granicy	granicach
Dative	mapie	mapom	granicy	granicom
Instrumental	mapą	mapami	granicą	granicami
Vocative	mapo	mapy	granico	granice

Turkish madness!

Turkish	English
Muvaffak	Successful
Muvaffakiyet	Success
Muvaffakiyet**siz**	**Un**successful ('**without** success')
Muvaffakiyetsiz**leş***(-mek)*	*(To)* **become** unsuccessful
Muvaffakiyetsizleş**tir***(-mek)*	*(To)* **make one** unsuccessful
Muvaffakiyetsizleştiri**ci**	**Maker of** unsuccessful ones
Muvaffakiyetsizleştirici**leş***(-mek)*	*(To)* **become** a maker of unsuccessful ones
Muvaffakiyetsizleştiricileş**tir***(-mek)*	*(To)* **make one** a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiri**ver***(-mek)*	*(To)* **easily/quickly** make one a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştirivere**bil***(-mek)*	*(To)* **be able to make** one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştirivere**meye**bil*(-mek)*	**Not** *(to)* be able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştireremeyebil**ecek**	**One who is** not able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebilecek**ler**	**Those** who are not able to make one easily/quickly a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebilecekleri**miz**	Those who **we** cannot make easily/quickly a maker unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimiz**den**	**From** those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizden**miş**	**(Would be)** from those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmiş**siniz**	**You** would be from those we can not easily/quickly make a maker of unsuccessful ones
Muvaffakiyetsizleştiricileştiriveremeyebileceklerimizdenmişsiniz**cesine**	**Like** you would be from those we can not easily/quickly make a maker of unsuccessful ones

Data

Language	Typology	$\frac{\text{BPE}}{\text{word}}$	Tags	Train	Dev	Test
Basque-BDT	Agglutinative	1.79	919	97k	12k	11k
Finnish-TDT	Agglutinative	1.98	591	161k	19k	20k
Turkish-IMST	Agglutinative	1.73	1056	46k	5k	5k
Estonian-EDT	Agglutinative	1.86	512	346k	43k	43k
Spanish-AnCora	Fusional	1.25	177	439k	55k	54k
Arabic-PADT	Fusional	1.39	300	225k	28k	28k
Czech-CAC	Fusional	1.77	990	395k	50k	49k
Polish-LFG	Fusional	1.75	634	104k	13k	13k

Model outline

1. Process sentence through the XLM-R$_{\text{base}}$ model.

Model outline

1. Process sentence through the XLM-R_{base} model.
2. Compute weighted sum over transformer layers with a parameter $w \in \mathbb{R}^L$

Model outline

1. Process sentence through the XLM-R$_{\text{base}}$ model.
2. Compute weighted sum over transformer layers with a parameter $w \in \mathbb{R}^L$
3. Align BPE-tokens to words

Model outline

1. Process sentence through the XLM-R$_{\text{base}}$ model.
2. Compute weighted sum over transformer layers with a parameter $w \in \mathbb{R}^L$
3. Align BPE-tokens to words
4. Compute word embeddings with a function f

Composition functions

A word X consists of the aligned BPE token embeddings and is a matrix of size $(T, 768)$ where T is the number of aligned tokens.

- First: $f(X)_i = X_i^0$
- Sum: $f(X)_i = \sum_{j=1}^{T} x_i^j$
- Mean: $f(X)_i = \frac{1}{T} \sum_{j=1}^{T} x_i^j$
- RNN: Use final output from a LSTM

Model outline

1. Process sentence through the XLM-$\mathrm{R}_{\mathrm{base}}$ model.
2. Compute weighted sum over transformer layers with a parameter $w \in \mathbb{R}^L$
3. Align BPE-tokens to words
4. Compute word embeddings with a function f
5. Run the sentence through a word LSTM

Model outline

1. Process sentence through the XLM-R_{base} model.
2. Compute weighted sum over transformer layers with a parameter $w \in \mathbb{R}^L$
3. Align BPE-tokens to words
4. Compute word embeddings with a function f
5. Run the sentence through a word LSTM
6. Predict grammatical features for the words

Training and Experiments

- We explore both finetuning the XLM-R model, and extracting bare features
- When finetuning, we freeze the XLM-R model the first epoch
- Adam optimizer (using cosine annealing learning rate with hard resets) with a learning rate of 0.001
- We use a lower learning rate for the XLM-R model ($1\mathrm{e}{-6}$)
- Label smoothing of 0.03
- Weight decay of 0.05 and dropout throughout the model

Recap/outline

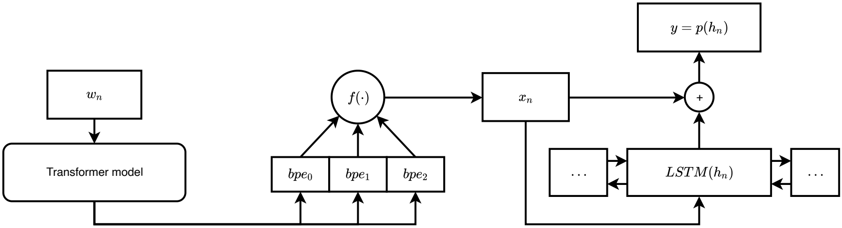


Figure: Model outline for a single word

Results - Finetuning

		Finetuning			
Treebank	Baseline	First	Sum	Mean	RNN
Basque-BDT	.676	.857	.884	.877	**.901**
Finnish-TDT	.751	.961	.958	.960	**.965**
Turkish-IMST	.620	.848	.859	.855	**.884**
Estonian-EDT	.740	.956	.955	.955	**.961**
Spanish-AnCora	.842	.977	.977	.977	**.979**
Arabic-PADT	.770	.946	.946	.947	**.951**
Czech-CAC	.771	.968	.968	.968	**.975**
Polish-LFG	.657	.956	.953	.953	**.959**
Average	.728	.933	.937	.936	**.946**

Table: Accuracy for morphological tagging for the finetuning regime.

Results - Feature extraction

Treebank	Baseline	Feature extraction First	Sum	Mean	RNN
Basque-BDT	.676	.759	.789	.780	**.834**
Finnish-TDT	.751	.853	.856	.847	**.899**
Turkish-IMST	.620	.742	.741	.735	**.775**
Estonian-EDT	.740	.855	.856	.853	**.901**
Spanish-AnCora	.842	.951	.954	.952	**.962**
Arabic-PADT	.770	.920	.923	.920	**.936**
Czech-CAC	.771	.863	.887	.881	**.924**
Polish-LFG	.657	.828	.844	.840	**.878**
Average	.728	.846	.856	.851	**.888**

Table: Accuracy for morphological tagging for the feature extraction regime.

Results - Words composed of two or more tokens

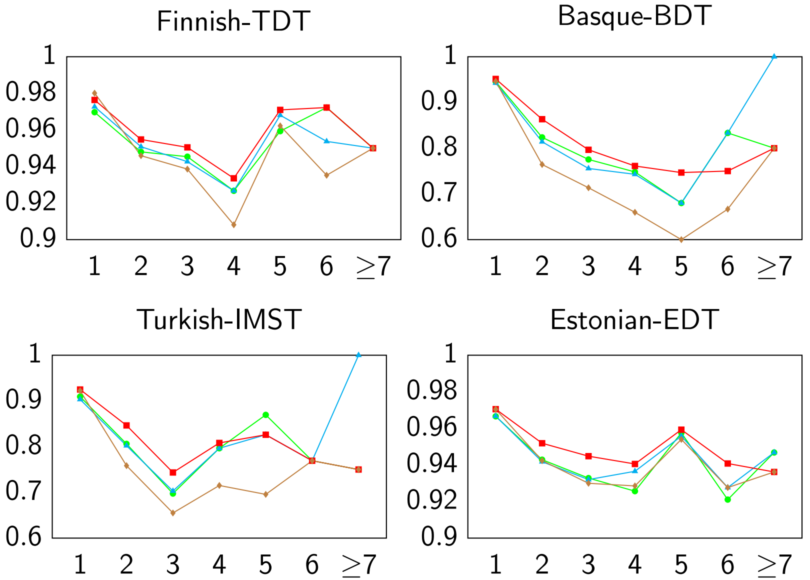
	Finetuning				Feature extraction			
Treebank	First	Sum	Mean	RNN	First	Sum	Mean	RNN
eu-BDT	.739	.802	.790	**.835**	.657	.715	.703	**.774**
fi-TDT	.940	.946	.946	**.952**	.780	.805	.794	**.861**
tr-IMST	.730	.780	.778	**.818**	.653	.683	.664	**.711**
et-EDT	.938	.939	.939	**.949**	.779	.805	.803	**.868**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

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

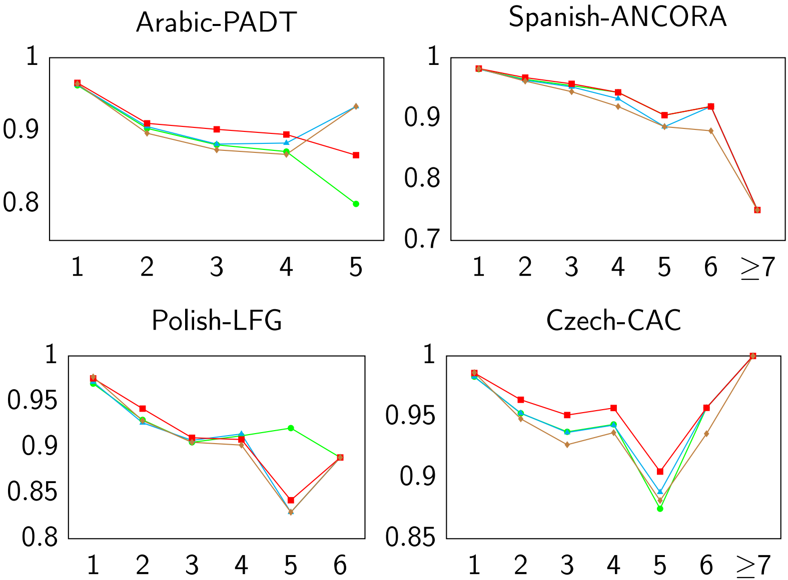
Results - Accuracy given tokens-per-words

- We might also be interested in how the accuracy looks when the number of tokens per word varies
- Roughly the same trends are observed, but with some variation
- Note: the distribution of tokens per word is zipfian

Accuracy given tokens per word - Agglutinative languages



Accuracy given tokens per word - Fusional languages



Commutative methods

- **First:** This method adds an implicit objective to the transformer model, push all the predictive information to the first token of a word
- **Sum and Mean:** Sum performs slightly better than averaging in the feature-extraction training regime, but the difference is essentially gone when finetuning.
- An advantage the RNN method have over these three methods is more capacity (in terms of additional parameters)
- To make a fair comparison, we parameterize these methods with a non-linear transformation with ReLU activation, which we pass all token embeddings through.

Parameterization of First, Mean and Sum

	Finetuning				Feature extraction			
	First	Sum	Mean	RNN	First	Sum	Mean	RNN
eu	.864	.894	.890	**.901**	.772	.793	.794	**.834**
fi	.958	.959	.961	**.965**	.857	.856	.855	**.899**
tr	.850	.875	.867	**.884**	.742	.722	.729	**.775**
et	.956	.958	.958	**.961**	.865	.856	.853	**.901**
sp	.978	.977	.978	**.979**	.953	.954	.952	**.962**
ar	.949	.945	.947	**.951**	.925	.923	.920	**.936**
cz	.969	.972	.972	**.975**	.873	.887	.881	**.924**
pl	.957	.953	.955	**.959**	.832	.844	.840	**.878**
Avg.	.935	.942	.941	**.946**	.852	.854	.853	**.888**
Diff.	+.002	+.005	+.005	-	+.006	-.002	+.002	-

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

Conclusions

- Using an RNN to compute word embeddings for morphological tagging consistently outperforms three other methods, across eight languages with varying morphology.
- For future work: Test this on all languages in UD to improve the robustness of the results.
- Does the composition function matter as much for other tasks?
- Are there alternative non-commutative methods that is more effective than using an RNN?