Lexically Grounded Subword Segmentation

Jindřich Libovický and **Jindřich Helcl**
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
V Holešovičkách 2, 180 00 Prague, Czech Republic
{libovicky, helcl}@ufal.mff.cuni.cz

Abstract

We present three innovations in tokenization and subword segmentation. First, we propose to use unsupervised morphological analysis with Morfessor as pre-tokenization. Second, we present an algebraic method for obtaining subword embeddings grounded in a word embedding space. Based on that, we design a novel subword segmentation algorithm that uses the embeddings, ensuring that the procedure considers lexical meaning. Third, we introduce an efficient segmentation algorithm based on a subword bigram model that can be initialized with the lexically aware segmentation method to avoid using Morfessor and large embedding tables at inference time. We evaluate the proposed approaches using two intrinsic metrics and measure their performance on two downstream tasks: part-of-speech tagging and machine translation. Our experiments show significant improvements in the morphological plausibility of the segmentation when evaluated using segmentation precision on morpheme boundaries and improved Rényi efficiency in 8 languages. Although the proposed tokenization methods do not have a large impact on automatic translation quality, we observe consistent performance gains in the arguably more morphological task of part-of-speech tagging.

1 Introduction

Statistical approaches to subword segmentation are the state of the art in most natural language processing (NLP) applications of neural networks, most notably the Transformer model (Vaswani et al., 2017). The Unigram model from SentencePiece (Kudo and Richardson, 2018) and Byte-Pair Encoding (BPE; Sennrich et al., 2016) are among the two most widely employed tokenization techniques. These methods gained popularity because of their versatility – they are language-independent and have convenient properties for model training, reducing the vocabulary size while assuring even learning of the token representations.

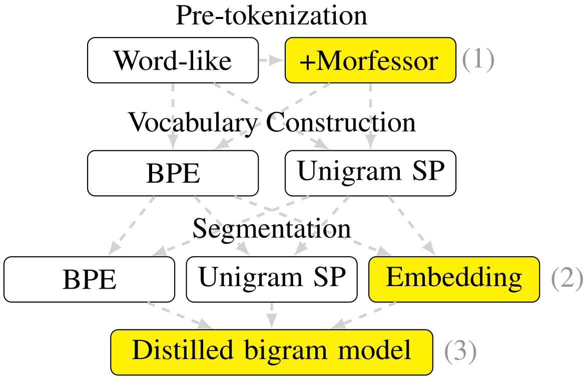

Figure 1: We organize subword tokenization learning into four steps: pre-tokenization, vocabulary learning, inference, and distillation for efficiency. Steps (1)–(3) highlighted in yellow are specific contributions of this paper.

Despite the indisputable advantages, one aspect of the statistical word segmentation algorithms has remained a thorn in the eyes of many linguistically-oriented researchers: *Subwords do not reflect morphology*. This problem is especially pronounced in multilingual models, which share a common vocabulary across all languages. Without a careful and balanced data selection, lower-resourced languages tend to have fewer allocated subwords, resulting in a large token-to-word ratio (Haddow et al., 2022; Limisiewicz et al., 2023).

We posit that a strong segmentation retains the property of the statistical approaches, i.e., that frequent words are split into fewer tokens than rare words. However, once a word is split into more tokens, the subword boundaries should ideally match the actual morpheme boundaries.[1] We hypothe-

[1] We use the word *morpheme* for morphologically motivated subword units. Some theories (Žabokrtský et al., 2022) distinguish *morphs* as surface realizations of abstract morphemes as the smallest units of meaning. Where appropriate, we follow this distinction for clarity. By *morpheme boundaries*, we mean boundaries between morphs within a word.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 7403–7420
November 12-16, 2024 ©2024 Association for Computational Linguistics

size that the standard algorithms lack morphology awareness because they do not work with lexical meaning, which is a crucial concept in language morphology.

Following Schmidt et al. (2024), we conceptualize tokenization as a process with three steps (as illustrated in Figure 1): pre-tokenization, vocabulary construction, and segmentation. Within this conceptual framework, we propose three innovations throughout the whole process:

(1) We consider unsupervised morphological segmentation as an alternative for pre-tokenization.

(2) Propose a novel lexically grounded segmentation algorithm based on word and subword embeddings.

(3) We propose an efficient statistical segmentation algorithm using subword bigram statistics that can be used to distill complex tokenization pipelines into an efficient algorithm.

In Section 2, we discuss pre-tokenization and vocabulary construction. Besides the standard pre-tokenization, which splits the text into word-like units (words, punctuation, etc.), we also experiment with Morfessor (Smit et al., 2014), which we apply on top of the word-like pre-tokenized text.

For lexically grounded segmentation, we derive *a formula for computing subword embeddings* using a pre-trained word embedding model and a training corpus (Section 3.1). Next, we use the subword embeddings to design *a subword segmentation algorithm based on semantic similarity* between the word and its subwords (Section 3.2).

Finally, we propose *a subword-bigram-based statistical segmentation algorithm* that retains the properties of the embedding-based segmentation (Section 4). With the bigram-based algorithm, we can have a model for subword segmentation that does not require running Morfessor or storing a large embedding table.

We test our approach using two intrinsic evaluation metrics and two downstream tasks (Section 5.1). In the intrinsic evaluation, we test our approach on the SIGMORPHON 2018 shared task dataset (Batsuren et al., 2022) and observe significantly better morphological generalization in both proposed algorithms with a fixed vocabulary size. We also measure the Rényi efficiency (Rényi, 1961) of the unigram distribution of the segmented text, which has been shown to correlate with downstream model performance (Zouhar et al., 2023). Additionally, we evaluate our segmentation algorithm on Part-of-Speech (POS) Tagging using Universal Dependencies (Zeman et al., 2024), showing an improvement compared to other segmentations. Finally, we evaluate our tokenization on machine translation using a simulated low-resource IWSLT 2017 dataset (Cettolo et al., 2017) where we reach results comparable with currently used subword tokenizers.

We show the code examples in Appendix A and we release the code for the segmentation tool, LEGROS,[2] as well as the experimental code.[3]

2 Pre-tokenization and Vocabulary Construction

Neural networks can only have limited vocabularies in order 10^4–10^5, which rules out using word-based vocabularies. A common solution is statistical heuristics that keep frequent words intact and split rare words into smaller units, ensuring that there are no rare tokens, such that embeddings of all tokens get updated reasonably often. The most popular methods are Byte-Pair Encoding (BPE; Sennrich et al., 2016) based on greedily merging the most frequent token pairs and the Unigram model (as implemented in SentencePiece; Kudo, 2018) that returns high-probability segmentations using a unigram language model. However, these methods manifest low morphological generalization, which in turn might lead to reduced interpretability, compositional generalization, and cross-lingual transfer capabilities.

Perhaps the most straightforward approach for lexically grounded word segmentation is to use unsupervised morphological analyzers, such as Morfessor. However, direct use of these linguistically motivated tools leads to worse results (Macháček et al., 2018) and is only beneficial in low-resource scenarios (Soulos et al., 2021; Gaser et al., 2023). Furthermore, morphological analysis does not fully address the problems of rare tokens and vocabulary size. To address these issues, we propose only using morphological analyzers during pre-tokenization (Step 1 in Figure 1). After pre-tokenization, we apply the well-established statistical methods for vocabulary construction. This combination ensures that there will be a low number

[2] https://github.com/ufal/legros
[3] https://github.com/ufal/legros-paper

of rare tokens and efficient control of vocabulary size while still preserving the lexical meaning of the subwords.

3 Segmentation with Subword Embeddings

In this section, we describe a novel lexically-grounded segmentation method (Step 2 in Figure 1).

When considering language morphology, we assume the word can be decomposed into several smaller meaningful units that carry the meaning of the original word when combined together. We consider the segmentation of a word to be lexically grounded when it respects the word's meaning and does not introduce subword boundaries in the middle of meaningful units. To find such a segmentation, we need to model the meaning of both words and subword units jointly.[4]

A widely used proxy for capturing the lexical meaning of words is word embeddings. To capture the meaning of subwords, we introduce a method to compute subword embeddings in a shared space with the word embeddings (§ 3.1). We also describe a segmentation algorithm that takes the subword embeddings into account (§ 3.2).

3.1 Subword Embeddings

We obtain the joint embedding model of words and subwords by extending the skip-gram model (Mikolov et al., 2013) to subword units. Specifically, we derive a formula for computing the embedding of any substring in a training dataset, situating its representation within the skip-gram model embedding space.

Skip-gram models are trained to produce a probability distribution of words that are likely to appear within a certain context window around a given input word x. When we extend this model to handle substrings, each substring is used to predict the whole words that appear within the context window of any word that contains the substring. As a result, the embeddings of the substrings are determined by the contexts of the words they are part of.

To compute the subword embeddings, we require a tokenized training dataset $\mathcal{D}$ and a trained skip-gram word embedding model with a vocabulary $\mathcal{V}$. In addition to its input embedding matrix $E \in \mathbb{R}^{|\mathcal{V}|\times d}$ where d is the dimension of the word embedding vectors, we also need the output matrix $W \in \mathbb{R}^{d\times|\mathcal{V}|}$.

The statistics of skip-gram models. Using data $\mathcal{D}$, we denote the symmetric word cooccurrence matrix $C \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ that for each pair of words $x, y \in \mathcal{V}$, $C_{x,y}$ contains the frequency of x and y appearing within the same context window in $\mathcal{D}$. Then, our method relies on the following observation:

$$\mathrm{softmax}(EW) \approx \mathrm{norm}(C) \tag{1}$$

where norm means row-wise normalization.

This follows from the fact that the skip-gram model optimizes cross-entropy between the predicted distribution of neighboring words and the empirical distribution in the training data. It is usually approximated by stochastic minibatch training with negative sampling instead of computing the full softmax. The empirical distribution can be obtained by normalizing the count matrix C, which leads to the following optimization problem:

$$\min_{E,W} \mathrm{XENT}(\mathrm{softmax}(EW), \mathrm{norm}(C)) \tag{2}$$

By Gibbs inequality, the cross-entropy is minimum if $\mathrm{softmax}(EX) = \mathrm{norm}(C)$. This leads to Equation 1. We use the approximation sign ($\approx$) to stress that stochastic optimization solves the problem only approximately. When training word embeddings, we must find both E and W. When extending the model for subwords, we keep the W fixed, and we only need to find the (newly added) subword portion of E, which we call E_s.

Extension to subwords. Next, we choose a set of subwords $\mathcal{S}$. We either select the set of all substrings present in $\mathcal{D}$ up to a certain length, or we use the set of subwords from an existing segmentation. We then define a segmentation matrix $A \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{V}|}$ such that:

$$A_{s,x} = \begin{cases} 1, & \text{if } s \text{ belongs to } x, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Then, the multiplication AC corresponds to the subword-word cooccurrence matrix. Thus, we can find the substring embedding matrix $E_s \in \mathbb{R}^{|\mathcal{S}|\times d}$ by solving the following formula:

$$\mathrm{softmax}(E_s W) \approx \mathrm{norm}(AC), \tag{4}$$

which can be solved using a least-square approximation as:

$$E_s = \log(\mathrm{norm}(AC)) W^{-1}_{\mathrm{right}} \tag{5}$$

[4]Linguistic theories often work with the concept of morphs and morphemes as the smallest meaningful units. However, our solution tries to be theory-agnostic, so it can work with any subword units regardless of their theoretical justification.

where W_{right}^{-1} is the right-inverse of the skip-gram's output matrix W.

3.2 Segmentation

In this section, we apply the subword embedding model to lexically grounded subword segmentation. We propose an algorithm based on the word-subword similarities within the shared embedding space. Following the Unigram model from SentencePiece (Kudo, 2018), which searches for a segmentation that maximizes the probability under a subword unigram model, we use a dynamic programming algorithm (shown in Algorithm 1 in Appendix A) to find the segmentation (sequence of subwords) that maximizes a similarity-based score.

Formally, for a word x and a segmentation $s_1, s_2, \ldots, s_n$, the similarity score is the sum of cosine similarities between the embedding of x and the embeddings of each of the subwords s_i, minus a length penalty of α per each subword:

$$\sum_{i=1}^{n} \frac{(E(x)) \cdot (E_s(s_i))}{\|E(x)\| \cdot \|E_s(s_i)\|} - \alpha. \tag{6}$$

Increasing the value of α forces the algorithm to use fewer subwords. In other words, α controls what weight we put to the semantic similarity and what weight we put to minimize the number of subwords. Based on preliminary results, we set α to 1 and keep it fixed in all experiments.

Unlike the Unigram segmentation, the subword scores are not static but depend on the segmented word. Therefore, the segmentation can be viewed as a word-specific unigram model.

As stated in the previous section, the computation of the subword embeddings requires an existing subword vocabulary $\mathcal{S}$ and the segmentation matrix A. We initialize $\mathcal{S}$ with the set of subwords used by another segmentation algorithm. We only set $A_{s,x} = 1$ when s has been used as a subword of x.

After initialization, we iteratively refine the segmentation in two alternating steps until convergence.

1. For a segmentation matrix A, calculate subword embeddings E_s (Equation 5).

2. For subword embeddings E_s, find a new best segmentation and update the segmentation matrix A accordingly. Note that subwords not used in this step are never used again, and therefore, the vocabulary shrinks as the algorithm proceeds.

4 Bigram model

The segmentation algorithm described in the previous section has several drawbacks: It requires storing relatively large embedding tables for words and subwords and does not generalize for OOV words without embeddings. Moreover, pre-tokenization with Morfessor requires running language-specific models, making the segmentation more computationally demanding than the established method.

We avoid this drawback by introducing an alternative segmentation algorithm based on subword bigram statistics. It is a straightforward generalization of the commonly used Unigram model. At inference time, we search for a segmentation that maximizes probability predicted by a subword bigram model instead of a unigram model. The optimization problem is solvable using dynamic programming, similar to the Unigram model. However, the algorithm has a quadratic complexity in the segmented string length. Therefore, we propose using a linear-time beam search algorithm that only considers k best segmentations in each step. The full algorithm is described in Algorithm 2 in Appendix A.

We use the subword bigram statistic obtained by counting subword bigram and unigram frequencies in a corpus tokenized by a tokenizer that we want to distill into the bigram model. To account for unknown bigrams encountered during inference, we need to eliminate zero probabilities from the bigram distribution. To this end, we apply Laplacian smoothing, i.e., we increase the frequency of every bigram $(s_i|s_{i-1})$ by one. Additionally, if s_{i-1} is an unknown unigram, we assign the unigram probability of s_i to the bigram. If both s_i and s_{i-1} are unknown unigrams, we assign uniform probability $1/|\mathcal{S}|$ to the bigram.

5 Experiments

We evaluate our proposed methods intrinsically using morpheme boundary precision and Rényi efficiency, as well as extrinsically on two downstream tasks: part-of-speech tagging and machine translation.

5.1 Intrinsic Evaluation

We evaluate the capability of our framework to capture morphological boundaries and compare it with commonly used segmentation methods. Our main evaluation metrics are precision on morpheme boundaries (given a fixed vocabulary size budget) and Rényi efficiency (Rényi, 1961) of the token distribution, which was shown to be a good predictor of downstream performance of a tokenizer (Zouhar et al., 2023).

Test data. For the morpheme boundary evaluation, we use the test set from the SIGMORPHON 2022 Shared Task on Morpheme Segmentation (Batsuren et al., 2022), which contains test data for nine languages (Czech, English, Spanish, Hungarian, French, Italian, Russian, Latin, Mongolian). We omit Latin due to the lack of resources for training word embeddings. Except for Czech (which contains surface-level segmentation into morphs), each test set consists of word decompositions into morphemes. This means that the original words cannot be reconstructed by simply concatenating the morphemes. To be able to evaluate word segmentations in all languages, we use a set of heuristic rules to map the morphemes to the surface form.

To measure the Rényi efficiency of the token distribution, we use 4,000 sentences randomly sampled from the (plain text) training data described in the following paragraph.

Experimental settings. We use the skip-gram model from FastText (Bojanowski et al., 2017) to train the word embeddings. For all languages except Mongolian, we train the model on 50M sentences from NewsCrawl (Kocmi et al., 2022). We use 15M sentences from CC-100 (Conneau et al., 2020) for Mongolian. We lowercase and pre-tokenize the text using Sacremoses,[5] and for experiments with Morfessor pre-tokenization, we train Morfessor (Smit et al., 2014) with the default parameters. We apply Morfessor on already pre-tokenized text as a second step. We use a vocabulary size of 200k, an embedding dimension of 200, and a window size of 5. We train the embeddings for 10 epochs for both pre-tokenization setups.

As a baseline, we prepare BPE and Unigram tokenizers with vocabularies 1k, 2k, 4k, 8k, 16k, 24k, 32k, and 48k using the same plain text dataset.

We use the segmentation from the BPE and Unigram subwords to initialize the matrix A from Equation 3 and iterate our algorithm. Finally, we use the bigram statistics from 200k embedding vocabulary and segment the test set using the subword bigram language model.

Segmentation evaluation. Unlike the original SIGMORPHON shared task evaluation, where the evaluation metric was the F_1 score measured on the morphemes themselves, we measure the morpheme boundary precision for a given vocabulary size. We believe this setup best captures the use of subword tokenizers in neural networks where we have a vocabulary budget given by the model architecture. However, we do report also recall and F_1 score for completeness.

Results. The main results for the 32k vocabulary are presented in Table 1. Across all languages, Unigram reaches better precision than BPE, consistently with previous work (Batsuren et al., 2022). Pre-tokenization using Morfessor consistently outperforms word-like pre-tokenization across all languages in morpheme boundary precision. Using lexically grounded embedding-based segmentation improves compared to the default BPE and Unigram segmentation algorithms. The difference is more pronounced with the word-like pre-tokenization. Distillation into the bigram model usually leads to a small decrease in the boundary precision. The performance of BPE and the Unigram model for vocabulary construction is language-dependent.

The Rényi efficiency is significantly higher for Morfessor pre-tokenization. Unlike morpheme boundary precision, distilling the embedding-based segmentation into a bigram model has almost no effect on Rényi efficiency. Segmentation based on the Unigram model vocabulary achieves the best results.

Figure 2 shows morpheme boundary precision, recall, and F_1 score for Czech for different vocabulary sizes; additional languages are presented in the Appendix in Figure 3. The boundary precision increases with the increasing vocabulary size, whereas the recall has the opposite trend. Our segmentation methods improve the boundary precision in all cases. Word-like pre-tokenization has a negligible effect on recall. On the other hand, adding Morfessor to pre-tokenization decreases recall.

We also show a random sample of segmented Czech, English, and French words in the Appendix in Table 10.

5.2 POS Tagging Evaluation

In our first extrinsic evaluation, we experiment with POS tagging as a simple task that directly involves language morphology.

Data. We use Universal Dependency (UD) Corpora (Zeman et al., 2024) for the languages from

[5] https://github.com/hplt-project/sacremoses

Vocab.		Inf.	Morpheme boundary precision								Rényi efficiency							
			cs	en	es	fr	hu	it	mn	ru	cs	en	es	fr	hu	it	mn	ru
Word-like	BPE	Orig	76.5	56.6	60.6	57.1	77.0	52.9	78.8	61.7	.419	.429	.396	.421	.373	.437	.470	.414
		Emb.	78.9	65.8	63.9	63.3	82.4	58.3	88.9	64.2	.422	.435	.403	.427	.387	.443	.479	.424
		Big.	79.4	66.1	63.2	62.9	81.5	58.2	88.1	66.1	.423	.435	.404	.428	.388	.444	.480	.425
	Uni.	Orig	84.3	64.6	63.1	64.7	80.5	53.3	90.4	66.8	.424	.432	.398	.425	.382	.442	.478	.423
		Emb.	87.0	68.3	65.2	66.5	82.6	57.0	89.8	67.6	.424	.437	.407	.433	.390	.447	.468	.431
		Big.	86.8	68.8	64.4	66.2	82.2	57.3	89.3	69.1	.425	.437	.408	.434	.391	.448	.469	.433
Morfessor	BPE	Orig	88.4	70.7	**66.4**	66.4	82.3	**63.2**	90.7	69.1	.449	.437	.422	.446	.391	.455	.497	.451
		Emb.	88.9	**72.0**	66.3	67.0	84.9	62.0	**92.5**	71.5	.451	.440	.425	.449	.401	.457	.500	.456
		Big.	88.7	69.9	66.2	**67.5**	84.3	62.8	91.8	71.2	.452	.440	.426	.449	.400	.458	.500	.457
	Uni.	Orig	89.4	70.3	65.3	65.4	84.0	61.4	90.1	70.6	.457	.441	.426	.452	.398	**.460**	**.503**	**.461**
		Emb.	**91.0**	70.3	65.0	65.7	**85.9**	61.7	91.0	**73.6**	.457	.441	**.429**	**.454**	**.403**	**.460**	.496	.458
		Big.	90.2	69.7	65.2	66.4	85.0	61.6	90.7	72.3	**.458**	**.442**	**.429**	**.454**	**.403**	**.460**	.496	.460

Table 1: Morpheme boundary precision on the SIGMORPHON 2018 test set and Rényi efficiency estimated on 4k plain text sentences for tokenizers with 32k-sized vocabularies. The best results in each column are bolded. The blue-yellow scale is fit to the value range per column. Results for 24k and 40k vocabularies are in Appendix in Table 8.

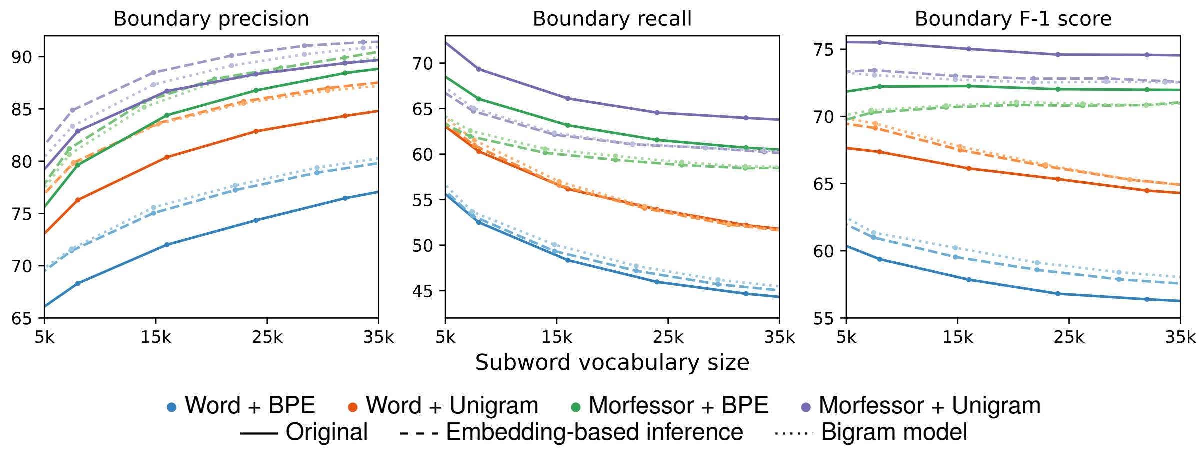

Figure 2: Boundary precision, recall, and F_1 score for Czech in the SIGMORPHON 2018 test set for different vocabulary sizes. For more other languages, see Figure 3 in the Appendix.

the intrinsic evaluation except for Mongolian, which does not have a UD corpus. See Table 6 in the Appendix for details of the corpora.

Model details. We train an LSTM-based tagger. We use an embedding layer of 300, two bidirectional LSTM layers (Hochreiter and Schmidhuber, 1997) of dimension 600, and a final projection into 18 POS tags. We use a batch size of 256 sentences and train for 3,200 steps using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.01. We select the best weights based on the loss on the development set. We prepend each word with a special word-separator token for subword segmentation and copy the POS tag to all its subwords. At inference time, we predict the tag from a distribution that averages the predictions for the individual subwords. We are aware that there are methods that would improve the performance of the tagger trained from scratch, e.g., including character-level features and using pre-trained word embeddings. In our experiments, we are mainly interested in how informative the segmentation is for the tagger.

Data preparation. We experiment with several segmentation methods. As a baseline, we use the word segmentation provided in UD and word segmented using Morfessor. Further, we experimented with word-like pre-tokenization, Morfessor pre-tokenization, and BPE and Unigram for vocabulary construction. For segmentation, we tested both the original subword segmentation corresponding to BPE and the Unigram model (denoted as Orig. in the results) and distilled bigram models created via the lexically grounded embedding-based segmentation (denoted as Ours in the results).

Tokenization			cs	en	es	fr	hu	it	ru	Aggr.
Word vocab			96.16	92.07	94.43	96.14	79.44	96.45	94.16	-2.013
Morfessor			96.01	92.05	94.61	96.19	78.14	96.64	94.48	-1.902
Word-like	BPE	Orig.	98.17	93.73	95.50	97.16	87.76	97.47	97.38	0.340
		Ours	98.19	93.78	95.58	97.23	88.88	97.56	97.40	0.471
	Uni.	Orig.	98.09	93.50	95.41	97.00	88.57	97.41	97.30	0.187
		Ours	98.17	93.76	95.56	97.11	89.68	97.58	97.43	0.447
Morfessor	BPE	Orig.	98.18	93.91	95.44	97.21	90.92	97.48	97.39	0.473
		Ours	**98.21**	**93.96**	**95.72**	**97.33**	**91.63**	97.74	**97.52**	**0.745**
	Uni.	Orig.	98.04	93.86	95.66	97.16	91.12	97.61	97.35	0.541
		Ours	98.11	93.95	95.72	97.29	91.51	**97.75**	97.52	0.712

Table 2: Test accuracies of POS tagging. The final column shows the averaged normalized accuracy (after subtracting the language-specific mean and dividing by the language-specific standard deviation). The blue-yellow scale is fit to the value range per column. More detailed results and additional baselines are in Table 9 in the Appendix.

Tokenization			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	0.0	0.4	0.7	0.4
		Ours	-0.0	0.5	0.8	0.4
	Uni.	Orig.	-0.0	0.9	0.9	0.6
		Ours	-0.2	0.5	0.6	0.3
Morfessor	BPE	Orig.	-1.0	-0.8	-0.7	-0.9
		Ours	-0.2	0.3	0.5	0.2
	Uni.	Orig.	-1.3	-0.9	-0.9	-1.0
		Ours	-0.1	0.3	-0.2	-0.0

Table 3: Mean deviation from the average chrF score for 18 language pairs of the IWSLT 2017. The blue-yellow scale is fit globally to the values across the table.

Results. The results are presented in Table 2 (with more details in Table 9 in the Appendix). In general, subword-based segmentation significantly outperforms word-like and Morfessor-based models. Morfessor pre-tokenization is slightly better than word-like pre-tokenization only in all languages, with a particularly pronounced difference in Hungarian, the only language in our test sets with agglutinative morphology. Our segmentation algorithm consistently improves over the default BPE and Unigram algorithms. The overall best tokenization approach combines the Morfessor pre-tokenization followed by the BPE algorithm for vocabulary construction and our bigram-based segmentation.

5.3 Machine Translation Evaluation

As a second downstream task, we evaluate our segmentation on machine translation (MT) in a simulated low-resource setup.

Experimental setup. We use the IWLST 2017 dataset of 18 language pairs (involving combinations of Arabic, English, Dutch, German, Italian, and Romanian) with the provided data splits for train, validation, and testing. The exact language pairs and dataset statistics are in the Appendix in Table 7. Similarly to POS tagging, we experiment with word-like and Morfessor pre-tokenization, BPE, and Unigram vocabulary construction (jointly on parallel data) and compare the default segmentation (Orig.) algorithms with the bigram-based segmentation distilled from the embedding-based segmentation algorithm (Ours).

We use the Transformer Base model (Vaswani et al., 2017) as implemented in Marian (Junczys-Dowmunt et al., 2018). We train the models using the Adam optimizer with learning rate 10^{-4} and the inverse square learning rate decay with 4,000 warmup steps with effective batch size 18,000 tokens.

Results. We evaluate the MT quality using the chrF scores (Popović, 2015),[6] see Table 11 in the Appendix for complete results. At first glance, there are only minor differences in translation quality across the tested methods and language pairs, except for a few outliers. Therefore, in Table 3,[7] we provide aggregated results across the languages: We first compute the mean chrF score per language pair and subtract it from the scores. Finally, we average the difference from the mean across languages. The results show that the word-based pre-

[6] We use the SacreBLEU implementation (Post, 2018): `chrF2|nrefs:1|case:mixed|eff:yes|nc:6|nw:0|space:no|version:2.0.0`

[7] Table 3 shows normalized chrF scores. See Table 5 in the Appendix for BLEU scores.

cs	en	es	fr	hu	it	ru	**Avg.**
-.30	.73	.69	.60	.95	.74	.33	**.54**

(a) POS-tagging (accuracy)

ar → en	-.70	en → ar	-.73	de → en	-.13	en → de	-.38
en → fr	-.06	fr → en	.19	en → nl	-.47	nl → en	.03
en → ro	-.31	ro → en	-.36	it → en	-.39	en → it	-.46
it → nl	-.43	nl → it	-.40	ro → it	-.04	it → ro	-.17
ro → nl	-.42	nl → ro	-.40	$\Longrightarrow$		**Avg.**	**-.31**

(b) Machine Translation (chrF)

Table 4: Pearson correlation of Rényi efficiency of the training data with the downstream performance. The blue-yellow scale is fit globally to the values across both tables.

tokenization outperforms Morfessor tokenization. Whilst our techniques have a slightly negative effect with the word-like pre-tokenization, adding Morfessor-based pre-tokenization shows significant improvements. Still, the overall MT quality stays behind the full Unigram and BPE preprocessing pipelines.

5.4 Rényi Efficiency

Finally, we evaluate the correlation between the results of our downstream tasks and Rényi Efficiency. Zouhar et al. (2023) conducted a theoretical analysis of information-theoretical properties of tokenizers and suggest to measure their unigram information efficiency. Information efficiency is the ratio of the unigram entropy of tokenized text and the maximum possible entropy given the vocabulary size. Instead of using the more common Shannon entropy, they use parametrized Rényi entropy with $\alpha = 2.5$ that they claim better correlates with the downstream performance on English-German MT.

To verify the claims of Zouhar et al. (2023), we computed the Pearson correlation of the Rényi efficiency of the training data in our experiments with the model performance. Our results are presented in Table 4. For POS tagging, Rényi efficiency is a good predictor of tagger performance in most languages except Czech. However, the correlation varies strongly between languages. In MT, we did not confirm the results of Zouhar et al. (2023): the correlation of the Rényi efficiency of the training data and the MT quality in terms of chrF is mostly negative and highly varies across language pairs.

6 Related Work

Subword embeddings. There are relatively few methods for obtaining static subword embeddings. FastText (Bojanowski et al., 2017) averages subword embeddings to obtain static word embeddings. However, subwords are stored in a hash table with many conflicts for better memory efficiency, making the subword embeddings unusable for our purposes. Heinzerling and Strube (2018) trained subword embedding for 275 languages and various vocabulary sizes using GloVe (Pennington et al., 2014) while treating subwords as standalone tokens. They, however, do not put the subword embeddings into relation to word embeddings. Static subword embeddings are, as the first layer, a part of most neural NLP models. However, none of the methods explicitly models the relationship between the words and subwords.

Subword segmentation. Besides the standard BPE (Sennrich et al., 2016) and the Unigram model (Kudo, 2018), several more recent approaches to subword segmentation exist. Xu et al. (2021) use optimal transport to find a replacement for greedy vocabulary construction of BPE, leading to more efficient bilingual vocabularies. He et al. (2020) and Meyer and Buys (2023) work with Dynamic Programming Encoding that includes subword selection into the language-modeling objective of in MT model with a decoder using character-level inputs. Yehezkel and Pinter (2023) introduce SaGe, which uses skip-gram training objective as a loss to replace unigram perplexity used in the Unigram model. Hofmann et al. (2022) show that changing the segmentation algorithm in a WordPiece (Schuster and Nakajima, 2012) tokenizer and a trained BERT model can improve classification performance. Schmidt et al. (2024) further elaborate on this idea and introduce an alternative segmentation algorithm that produces the minimum number of tokens given a vocabulary.

7 Conclusions

In this paper, we devised morphologically plausible methods for subword segmentation. Inspired by Schmidt et al. (2024), we divide the tokenization process into three steps: pre-tokenization, vocabulary construction, and segmentation.

We described three key contributions of our work. Our first contribution focuses on the pre-tokenization step: Instead of the standard approaches, which split the text into word-like units, we use Morfessor, which splits the text into morphemes. However, we only regard this as pre-tokenization. Next, we proposed a novel segmentation algorithm based on word and subword embeddings, which provides lexical grounding to the segmentation. Finally, we proposed a statistical bigram segmentation model that can be used to simplify complex tokenization pipelines.

The intrinsic evaluation results show that the proposed method better captures language morphology than standard statistical subword segmentation approaches. This is further confirmed by the results we obtained on POS tagging, in which information about morphology is a key feature.

However, our method did not significantly improve the performance of machine translation, which is a more complex NLP task. We argue that a dedicated analysis would be required to determine the exact influence of the lexically grounded segmentation on the translation quality, which might be improved in one dimension but reduced in another.

In our work, we have taken steps to create a more morphologically accurate tokenization method while keeping the benefits of statistical subword segmentation. We believe these methods will improve modeling language overall and contribute to model interpretability and cross-lingual transfer.

8 Limitations

The subword embedding formula derived in Section 3.1 requires a trained word embedding model and, therefore, relies on the quality of available data. This problem manifests mostly in underrepresented languages, many of which would benefit from morphology-aware segmentation.

In Section 5.1, we use a set of heuristic rules to map the morphemes to the surface form for some languages. These rules are language agnostic and may introduce noise into the evaluation. However, the results are consistent with Czech, annotated on the morph level.

Acknowledgements

We thank Tomasz Limisiewicz and Zdeněk Žabokrtský for discussing our early ideas on using subword embeddings for tokenization.

This work was supported by the Charles University project PRIMUS/23/SCI/023.

References

Khuyagbaatar Batsuren, Gábor Bella, Aryaman Arora, Viktor Martinovic, Kyle Gorman, Zdeněk Žabokrtský, Amarsanaa Ganbold, Šárka Dohnalová, Magda Ševčíková, Kateřina Pelegrinová, Fausto Giunchiglia, Ryan Cotterell, and Ekaterina Vylomova. 2022. The SIGMORPHON 2022 shared task on morpheme segmentation. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 103–116, Seattle, Washington. Association for Computational Linguistics.

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.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Marwa Gaser, Manuel Mager, Injy Hamed, Nizar Habash, Slim Abdennadher, and Ngoc Thang Vu. 2023. Exploring segmentation approaches for neural machine translation of code-switched Egyptian Arabic-English text. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3523–3538, Dubrovnik, Croatia. Association for Computational Linguistics.

Barry Haddow, Rachel Bawden, Antonio Valerio Miceli Barone, Jindřich Helcl, and Alexandra Birch. 2022. Survey of low-resource machine translation. *Computational Linguistics*, 48(3):673–732.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2020. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3042–3051, Online. Association for Computational Linguistics.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Valentin Hofmann, Hinrich Schuetze, and Janet Pierrehumbert. 2022. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, and Maja Popović. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5661–5681, Toronto, Canada. Association for Computational Linguistics.

Dominik Macháček, Jonás Vidra, and Ondrej Bojar. 2018. Morphological and language-agnostic word segmentation for NMT. In *Text, Speech, and Dialogue - 21st International Conference, TSD*, volume 11107 of *Lecture Notes in Computer Science*, pages 277–284, Brno, Czech Republic. Springer.

Francois Meyer and Jan Buys. 2023. Subword segmental machine translation: Unifying segmentation and target sentence generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2795–2809, Toronto, Canada. Association for Computational Linguistics.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. 2024. Tokenization is more than compression. *CoRR*, abs/2402.18376.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden. Association for Computational Linguistics.

Paul Soulos, Sudha Rao, Caitlin Smith, Eric Rosen, Asli Celikyilmaz, R. Thomas McCoy, Yichen Jiang, Coleman Haley, Roland Fernandez, Hamid Palangi, Jianfeng Gao, and Paul Smolensky. 2021. Structural biases for improving transformers on translation into morphologically rich languages. In *Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021)*, pages 52–67, Virtual. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7361–7373, Online. Association for Computational Linguistics.

Shaked Yehezkel and Yuval Pinter. 2023. Incorporating context into subword vocabularies. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia. Association for Computational Linguistics.

Zdeněk Žabokrtský, Niyati Bafna, Jan Bodnár, Lukáš Kyjánek, Emil Svoboda, Magda Ševčíková, and Jonáš Vidra. 2022. Towards universal segmentations: UniSegments 1.0. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1137–1149, Marseille, France. European Language Resources Association.

Daniel Zeman, Joakim Nivre, et al. 2024. Universal dependencies 2.14. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

A Code Examples

Below, we list Python implementations of the two proposed segmentation algorithms: Segmentation based on subword embeddings (Algorithm 1) and bigram segmentation (Algorithm 2).

B Statistis of Used Datasets

Table 6 contains statistics of the UD Treebanks used for POS Tagging evaluation. Table 7 contains basic statics of the IWSLT 2017 data used for machine translation evaluation.

Tokenization			Vocabulary 4k	8k	16k	Avg.
Word-like	BPE	Orig.	0.3	0.7	0.7	0.5
		Ours	0.3	0.4	0.5	0.4
	Uni.	Orig.	0.3	0.9	0.7	0.7
		Ours	0.2	0.8	0.3	0.5
Morfessor	BPE	Orig.	-0.7	-0.7	-0.8	-0.7
		Ours	-0.4	-0.1	-0.0	-0.2
	Uni.	Orig.	-1.0	-0.8	-1.0	-0.9
		Ours	-0.1	-0.0	-0.5	-0.2

Table 5: Mean deviation from the average BLEU score for 18 language pairs of the IWSLT 2017. The blue-yellow scale is fit globally to the values across the table.

C Additional Results

Here, we present additional results: Precision, recall, and F_1 Score on the SIGMORPHON 2018 test set (Figure 3) and segmentations of randomly sampled words in Czech, English, and French (Table 10). Table 9 contains more detailed results of POS tagging. Table 5 contains aggregated BLEU scores for MT experiments, and Table 11 contains individual chrF scores for the 18 language pairs.[8]

[8]SacreBLEU signature: `BLEU|nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0`

```
1 def embedding_segment(
2         word: str,
3         word_embedding: np.ndarray,
4         subword_embeddings: Dict[str, np.ndarray]) -> List[str]:
5 
6     # Costs of segmenting the word up to a certain length
7     costs = [0. for _ in range(len(word) + 1)]
8     # Backward pointers: position i says from what index we can get position i
9     prev = [0 for _ in word]
10 
11     # 1. Populate the segmentation cost table
12     for i in range(1, len(word) + 1):
13         # Now, we know how to segment everything up to position i-1 and want to find position i
14         indices = []  # Indices j from where we can go to position i
15         scores = []   # Scores corresponding to the indices
16         for j in range(i):  # 0..i
17             subword = word[j:i]
18             if subword in subword_embeddings:
19                 subword_embedding = subword_embeddings[subword]
20                 new_cost = costs[j] + cosine_similarity(
21                     word_embedding, subword_embedding) - 1
22                 scores.append(new_cost)
23                 indices.append(j)
24         # Best index from which we get to position i, i.e., the argmax of scores
25         idx = max(range(len(scores)), key=lambda i: scores[i])
26         costs[i] = scores[idx]
27         prev[i - 1] = indices[idx]
28 
29     # 2. Reconstruct the best segmentation by following the backward pointers
30     subwords = []
31     idx = len(prev) - 1
32     while idx >= 0:
33         new_idx = prev[idx]
34         sbwrd = word[new_idx:idx + 1]
35         subwords.append(sbwrd)
36 
37         idx = new_idx - 1
38     return list(reversed(subwords)), costs[-1]
```

Algorithm 1: Python code showing the segmentation algorithm using subword embeddings. On the input, `word` is the word to be segmented, `word_embedding` is its embedding, and `subword_embedding` is the subword embedding matrix.

It is a dynamic programming algorithm that first computes the scores of the best segmentation up to a given position in the string (kept in list `costs`) and what was the start index of the last subword in the best-scoring segmentation (kept in list `prev`). When moving to the next index in the for loop on line 12, we can rely on knowing the best segmentation score for all indices up to $i - 1$ from the previous iteration. Therefore, in the for loop on line 16, we can try all subwords that will bring us to index i. figure out the best possible subword that will extend the segmentation to index i.

In the second step, we use the list `prev` to reconstruct what subwords were used to the best score starting at the end of the word.

```python
def beam_search_segment(
        word: str,
        vocab: Set[str]
        beam_size: int = 5) -> List[str]:
    max_subword_length = max(len(tok) for tok in vocab)

    # List where the i-th position contains possible segmentations ending at position i
    segmentations = [[(["###"], 0.0)]] + [[] for _ in token]
    for start in range(len(token)):
        # Try to expand all segmentations ending at index `start`
        # with subwords of all possible lengths
        for length in range(1, vocab.max_subword_length + 1):
            end = start + length
            if end > len(token):
                break

            subword = token[start:end]
            if subword not in vocab and len(subword) > 1:
                continue

            # Expand from the current segmentations ending at index `start`
            for prev_segmentation, prev_score in segmentations[start]:
                # Compute the bigram log probability of the current `subword`
                # given the last subword of `prev_segmenation`
                score = log_probability(subword, prev_segmentation[-1])
                new_segmentation = prev_segmentation + [subword]
                new_score = prev_score + score  # Summing log probabilities
                segmentations[end].append((new_segmentation, new_score))

        # For each end index that follows, keep only the best `beam_size` segmentations
        for i, seg_list in enumerate(segmentations[start + 1:]):
            if len(seg_list) > beam_size:
                seg_list.sort(key=lambda x: x[1], reverse=True)
                segmentations[start + 1 + i] = seg_list[:beam_size]

    best_segmentation = max(segmentations[-1], key=lambda x: x[1])
    return best_segmentation[0][1:]
```

Algorithm 2: Python code for bigram segmentation. On the input, `token` is the token to be tokenized, `vocab` is the subword vocabulary, and `max_subword_length` controls the maximum number of characters in a subword. We assume there is a function `log_probability` that computes the log probability of a subword bigram.

Treebank	Train		Dev		Test	
	Sent.	Tokens	Sent	Tokens	Sent.	Tokens
Czech PDT	68k	1,192k	9k	162k	10k	177k
English EWT	12k	207k	2k	25k	2k	25k
Spanish GSD	14k	389k	1k	37k	1k	12k
French GSD	14k	364k	1k	36k	1k	10k
Hungarian Szeged	1k	20k	1k	11k	1k	10k
Italian ISDT	13k	294k	1k	12k	1k	11k
Russian SynTagRus	69k	1206k	8k	153k	8k	157

Table 6: Basic statistics of the splits of the UD treebanks used in the POS tagging evaluation in terms of number sentences and number of tokens.

Language pair	Train			Dev			Test		
	Sent	Src. tok	Tgt. tok.	Sent	Src. tok	Tgt. tok.	Sent	Src. tok	Tgt. tok.
ar-en	231k	3,817k	4,865k	1k	15k	21k	8k	136k	184k
de-en	206k	3,923k	4,318k	1k	19k	21k	8k	149k	162k
en-fr	232k	4,888k	5,360k	1k	21k	21k	8k	184k	193k
en-nl	237k	4,540k	4,009k	1k	20k	19k	2k	33k	31k
en-ro	220k	4,594k	4,201k	1k	20k	20k	2k	33k	32k
it-en	231k	4,846k	4,450k	1k	20k	19k	2k	32k	31k
it-nl	233k	4,105k	3,944k	1k	18k	19k	2k	29k	31k
ro-it	217k	4,169k	4,148k	1k	18k	20k	2k	29k	31k
ro-nl	206k	3,809k	3,939k	1k	18k	20k	2k	30k	32k

Table 7: Sizes of the IWSLT 2017 datasets in terms of the number of sentence pairs and the number of tokens on the source and the target side.

Vocab.		Inf.	Morpheme boundary precision								Rényi efficiency							
			cs	en	es	fr	hu	it	mn	ru	cs	en	es	fr	hu	it	mn	ru
Word-like	BPE	Orig	74.3	54.7	59.4	55.3	75.3	51.0	76.0	60.7	.430	.435	.404	.428	.382	.444	.478	.425
		Emb.	77.2	63.2	62.8	60.7	80.5	56.4	86.2	63.1	.432	.441	.410	.434	.396	.450	.487	.434
		Big.	77.7	63.7	63.0	60.5	79.7	56.3	85.8	64.9	.433	.441	.411	.435	.396	.451	.488	.435
	Uni.	Orig	82.9	62.2	61.2	62.5	78.5	51.2	88.9	65.0	.434	.437	.404	.430	.391	.448	.484	.432
		Emb.	85.7	66.4	63.4	64.3	80.8	55.6	88.2	66.5	.433	**.443**	.414	.439	.398	.453	.474	.439
		Big.	85.5	66.7	63.3	64.4	80.6	55.6	87.9	67.9	.435	**.443**	.415	.440	.399	.454	.475	.442
Morfessor	BPE	Orig	86.8	68.3	**64.5**	64.2	80.0	60.1	88.6	67.4	.454	.440	.425	.449	.398	.458	.499	.455
		Emb.	87.9	**70.4**	64.4	65.1	83.1	59.7	90.7	69.9	.456	**.443**	.428	.452	.407	.460	.502	.460
		Big.	87.6	67.9	64.2	**65.3**	82.5	60.3	90.2	69.6	.457	**.443**	.429	.452	.407	.461	.503	.461
	Uni.	Orig	88.3	68.8	63.9	64.1	82.1	60.1	89.9	68.9	.461	.442	.427	.453	.404	**.462**	**.504**	**.464**
		Emb.	**90.1**	69.0	64.4	65.0	**84.1**	**60.9**	**90.9**	**72.3**	.461	**.443**	.430	**.456**	**.409**	.461	.497	.461
		Big.	89.2	68.1	63.9	65.2	82.8	60.4	90.4	70.7	**.462**	**.443**	**.431**	**.456**	**.409**	**.462**	.497	.463

Vocab.		Inf.	Morpheme boundary precision								Rényi efficiency							
			cs	en	es	fr	hu	it	mn	ru	cs	en	es	fr	hu	it	mn	ru
Word-like	BPE	Orig	78.1	58.1	61.4	58.6	78.1	54.6	80.3	62.6	.412	.425	.392	.417	.366	.433	.466	.407
		Emb.	81.3	67.2	65.1	65.2	83.5	60.1	90.4	64.8	.415	.431	.400	.424	.381	.439	.475	.418
		Big.	81.8	68.7	65.4	66.3	83.4	61.0	90.4	67.7	.416	.431	.401	.424	.382	.440	.476	.419
	Uni.	Orig	85.6	66.7	65.0	66.9	81.8	55.1	92.0	67.8	.417	.429	.394	.421	.376	.438	.474	.417
		Emb.	87.9	69.9	66.7	68.1	83.7	58.7	91.5	68.4	.418	.434	.404	.430	.384	.443	.464	.426
		Big.	88.2	71.0	66.4	69.1	83.9	60.5	91.6	70.9	.420	.434	.405	.430	.385	.444	.465	.427
Morfessor	BPE	Orig	90.6	**73.5**	**69.1**	**69.7**	85.4	**66.8**	92.5	71.8	.446	.436	.421	.446	.388	.454	.497	.448
		Emb.	89.9	72.6	67.3	68.5	86.3	63.6	**93.0**	72.6	.449	.439	.424	.448	.397	.456	**.499**	.453
		Big.	89.4	71.4	67.1	68.6	85.5	64.6	92.5	72.5	.449	.439	.424	.448	.396	.456	**.499**	.454
	Uni.	Orig	90.2	70.8	65.7	65.8	85.4	63.3	91.9	71.6	.456	**.441**	.426	.452	.395	.452	.491	**.460**
		Emb.	**91.4**	70.4	65.0	65.7	**86.9**	63.3	92.7	**73.7**	.456	**.441**	.428	**.454**	**.400**	.458	.492	.456
		Big.	90.8	70.0	65.5	66.4	86.1	63.2	92.1	73.1	**.457**	**.441**	**.429**	**.454**	**.400**	**.459**	.492	.459

Table 8: Morpheme boundary precision on the SIGMORPHON 2018 test set and Rényi efficiency estimated on 4k plain text sentences for tokenizers with 24k and 40k-sized vocabularies. The best results in each column are bolded. The blue-yellow scale is fit to the value range per column.

Tokenization			cs	en	es	fr	hu	it	ru	Aggr.
Most frequent unigram			91.70	83.30	88.00	89.60	60.40	90.30	88.80	
HMM Tagger			93.70	87.60	91.70	93.00	72.80	93.30	91.00	
Word vocab			96.16 (0.19)	92.07 (0.56)	94.43 (0.24)	96.14 (0.30)	79.44 (1.10)	96.45 (0.27)	94.16 (0.51)	-2.013
Morfessor			96.01 (0.35)	92.05 (0.64)	94.61 (0.22)	96.19 (0.24)	78.14 (1.86)	96.64 (0.15)	94.48 (0.45)	-1.902
Word-like	BPE	Orig.	98.17 (0.03)	93.73 (0.16)	95.50 (0.14)	97.16 (0.08)	87.76 (1.02)	97.47 (0.10)	97.38 (0.05)	0.340
		Ours	98.19 (0.03)	93.78 (0.15)	95.58 (0.09)	97.23 (0.12)	88.88 (0.92)	97.56 (0.07)	97.40 (0.03)	0.471
	Uni.	Orig.	98.09 (0.08)	93.50 (0.17)	95.41 (0.09)	97.00 (0.06)	88.57 (0.50)	97.41 (0.10)	97.30 (0.04)	0.187
		Ours	98.17 (0.04)	93.76 (0.20)	95.56 (0.11)	97.11 (0.12)	89.68 (0.50)	97.58 (0.09)	97.43 (0.05)	0.447
Morfessor	BPE	Orig.	98.18 (0.02)	93.91 (0.16)	95.44 (0.13)	97.21 (0.13)	90.92 (0.40)	97.48 (0.06)	97.39 (0.04)	0.473
		Ours	98.21 (0.03)	93.96 (0.17)	95.72 (0.12)	97.33 (0.10)	91.63 (0.31)	97.74 (0.09)	97.52 (0.03)	**0.745**
	Uni.	Orig.	98.04 (0.06)	93.86 (0.18)	95.66 (0.06)	97.16 (0.07)	91.12 (0.44)	97.61 (0.07)	97.35 (0.05)	0.541
		Ours	98.11 (0.04)	93.95 (0.18)	95.72 (0.14)	97.29 (0.11)	91.51 (0.29)	97.75 (0.09)	97.52 (0.04)	0.712

Table 9: Test accuracies for POS tagging including standard deviations over 10 random seeds and simple baselines from NTLK.

Word (Czech)	Gold segmentation	BPE	Unigram	Ours
vykrášlit	vy␣krāš␣l␣i␣t	vy␣krá␣š␣lit	vy␣krá␣š␣lit	vy␣krá␣šl␣it
fluorově	fluor␣ov␣ě	flu␣or␣ově	fl␣u␣or␣ově	f␣lu␣or␣ově
horách	hor␣ách	horách	horách	horách
zkamenět	z␣kamen␣ě␣t	z␣kamen␣ě␣t	z␣ka␣me␣ně␣t	z␣kamen␣ě␣t
akcií	akci␣í	akcií	akcií	akcií
zdegenerovat	z␣de␣gener␣ova␣t	zde␣gener␣ovat	zde␣gen␣er␣ovat	zde␣gener␣ovat
rezervy	re␣zerv␣y	rezervy	rezervy	rezervy
neměly	ne␣m␣ě␣l␣y	neměly	neměly	neměly
poplatků	po␣plat␣k␣ů	poplatků	poplatků	poplatků
obnitkovat	ob␣nit␣k␣ova␣t	ob␣ni␣tk␣ovat	ob␣nit␣kovat	ob␣nit␣kovat
znesnadňovat	z␣ne␣snad␣ň␣ova␣t	zne␣snad␣ňovat	z␣ne␣snad␣ňovat	zne␣snad␣ňovat
přesunovat	pře␣sun␣ova␣t	přesu␣novat	přesun␣ovat	přesun␣ovat
jednota	jedn␣ot␣a	jedno␣ta	jedno␣ta	jedno␣ta
obklíčit	ob␣klíč␣i␣t	ob␣klí␣čit	ob␣klíč␣it	ob␣klíč␣it
krysí	krys␣í	kry␣sí	krys␣í	krys␣í
premií	prem␣i␣í	premi␣í	pre␣mi␣í	pre␣mi␣í
bříško	bříš␣k␣o	bří␣ško	bříško	bříško
odpovídat	od␣po␣víd␣a␣t	odpovídat	odpovídat	odpovídat
zakuklit	za␣kukl␣i␣t	za␣ku␣kli␣t	za␣ku␣kli␣t	za␣kukl␣it

Word (English)	Gold segmentation	BPE	Unigram	Ours
macroclumps	macro␣clump␣s	macro␣clum␣ps	macro␣cl␣ump␣s	macro␣clump␣s
gibbets	gibbet␣s	gib␣bets	gibb␣ets	gibb␣ets
phenoconverts	pheno␣convert␣s	phen␣o␣conver␣ts	phe␣no␣con␣vert␣s	ph␣eno␣convert␣s
ahura	ahura	a␣hur␣a	a␣h␣ura	ahu␣ra
bimonopoles	bi␣mono␣pole␣s	b␣im␣on␣opol␣es	bi␣mon␣o␣pole␣s	bi␣mono␣poles
nonwriter	non␣write␣r	non␣writer	non␣writer	non␣writer
molelike	mole␣like	mol␣eli␣ke	mole␣like	mole␣like
barnardsville	barnard␣s␣ville	bar␣nar␣d␣sville	barnard␣sville	barnard␣sville
pogues	pogue␣s	po␣gues	po␣gue␣s	po␣gu␣es
infractors	infractor␣s	infr␣actors	in␣fra␣ctor␣s	in␣fr␣actors
battlings	battling␣s	batt␣lings	battling␣s	battling␣s
larrup	larrup	lar␣r␣up	la␣rr␣up	lar␣ru␣p
detransformation	de␣trans␣form␣ation	de␣transformation	de␣trans␣form␣ation	de␣transform␣ation
deexciting	de␣excit␣ing	de␣exciting	de␣ex␣citing	de␣exciting
kalasies	kalasie␣s	kal␣as␣ies	kala␣s␣ies	kala␣s␣ies
canebrakes	cane␣brake␣s	can␣e␣brakes	can␣e␣bra␣kes	ca␣ne␣brakes
eskimological	eskimo␣log␣ical	es␣kim␣ological	es␣kim␣ological	es␣kim␣ological
unmisleading	un␣mis␣lead␣ing	un␣misleading	un␣mis␣leading	un␣misleading
neurofibromins	neuro␣fib␣r␣om␣in␣s	neuro␣fibro␣mins	neuro␣fi␣bro␣mins	neuro␣fibro␣mins

Word (French)	Gold segmentation	BPE	Unigram	Ours
parassiens	parassien␣s	par␣assi␣ens	par␣assi␣ens	pa␣ras␣siens
complaira	com␣plair␣a	compl␣ai␣ra	comp␣la␣ira	com␣plaira
salindrois	salindr␣ois	sal␣in␣dr␣ois	sali␣nd␣rois	sali␣nd␣rois
nampontois	nampont␣ois	nam␣pon␣tois	n␣amp␣ont␣ois	nam␣pont␣ois
sédimentologique	sédimentologi␣que	sé␣di␣ment␣ologique	s␣édi␣ment␣ologique	sé␣dim␣ent␣ologique
esquivée	esquiv␣é␣e	esqui␣vée	es␣qui␣vé␣e	es␣qu␣iv␣ée
flanc-garde	flanc␣-␣garde	fl␣anc␣-␣garde	flanc␣-␣garde	flanc␣-␣garde
moyen	moyen	moyen	moyen	moyen
antigangs	anti␣gang␣s	anti␣gangs	anti␣g␣ang␣s	anti␣gangs
forer	forer	for␣er	for␣er	fo␣rer
captivités	captivité␣s	capti␣vités	captivité␣s	captivité␣s
dépolymérisés	dé␣poly␣m␣é␣r␣is␣é␣s	dé␣poly␣m␣ér␣isés	dé␣po␣ly␣mé␣r␣isés	dé␣poly␣mé␣ris␣és
prévoiriez	pré␣voir␣iez	pré␣voi␣riez	prévoir␣iez	prévoir␣iez
déracinerais	dé␣racine␣r␣ais	dé␣rac␣in␣erais	d␣éra␣cine␣rais	dé␣racine␣rais
corécipiendaire	co␣récipiendaire	coré␣ci␣pi␣end␣aire	cor␣é␣ci␣pi␣end␣aire	co␣ré␣cip␣ien␣da␣ire
crustacyanines	crustacyanine␣s	cru␣st␣ac␣yan␣ines	crus␣t␣ac␣yan␣ines	crus␣tac␣yan␣ines
chambardés	chambard␣é␣s	cham␣bar␣dés	chamb␣ard␣és	cham␣bard␣és
joyeusain	joyeus␣ain	joy␣eu␣sain	joy␣e␣us␣ain	jo␣ye␣usain
influions	influ␣ions	influ␣ions	in␣flu␣ions	inf␣lu␣ions

Table 10: Example segmentations from the SIGMORPHON 2018 Czech, English, and French test sets. Green space symbols denote morphologically valid splits, and the red space symbols denote splits inside morphemes.

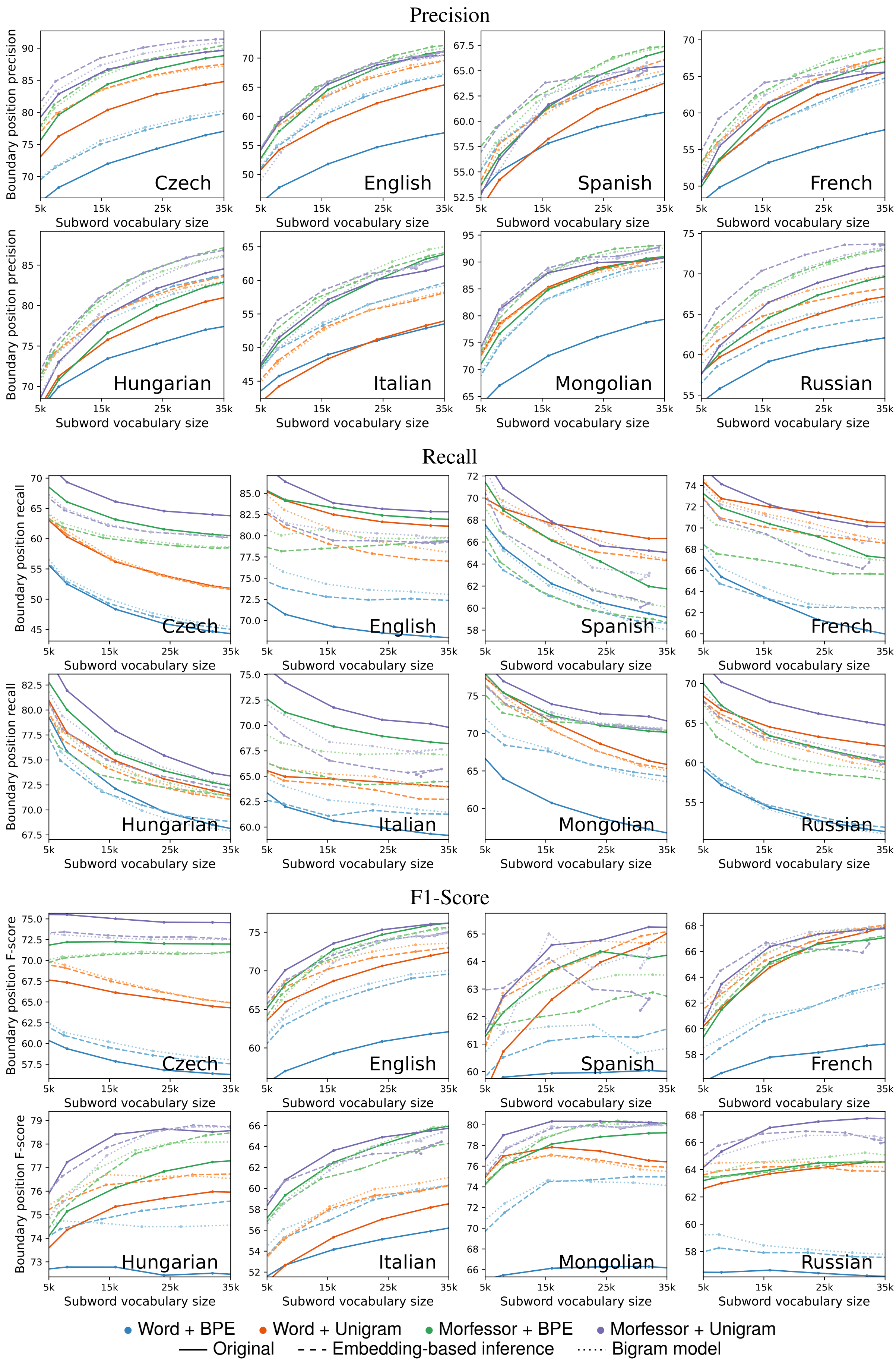

Figure 3: Boundary precision, recall and F1-score on the SIGMORPHON 2018 test set.

ara-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	38.8	39.4	39.9	39.4
		Ours	38.6	39.8	40.3	39.6
	Uni.	Orig.	40.1	40.6	41.3	40.7
		Ours	39.4	40.6	41.0	40.4
Morfessor	BPE	Orig.	39.9	40.1	40.7	40.2
		Ours	39.4	40.2	41.0	40.2
	Uni.	Orig.	39.5	40.5	40.0	40.0
		Ours	40.0	41.3	40.1	40.5

eng-ara			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	33.4	34.2	34.7	34.1
		Ours	33.4	34.0	34.4	33.9
	Uni.	Orig.	33.9	35.2	34.7	34.6
		Ours	33.9	34.4	34.9	34.4
Morfessor	BPE	Orig.	33.6	34.5	34.5	34.2
		Ours	34.1	35.0	35.1	34.7
	Uni.	Orig.	33.3	34.3	34.5	34.0
		Ours	33.9	35.0	34.3	34.4

deu-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	43.6	43.8	43.9	43.8
		Ours	43.2	43.8	43.9	43.6
	Uni.	Orig.	42.9	44.3	44.8	44.0
		Ours	43.7	44.1	44.6	44.1
Morfessor	BPE	Orig.	42.6	42.3	42.1	42.3
		Ours	43.1	43.2	43.5	43.2
	Uni.	Orig.	41.9	42.0	41.6	41.9
		Ours	43.2	42.9	43.5	43.2

eng-deu			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	44.8	45.4	45.0	45.1
		Ours	44.2	45.5	45.0	44.9
	Uni.	Orig.	44.4	45.6	45.6	45.2
		Ours	44.6	44.6	46.2	45.1
Morfessor	BPE	Orig.	42.2	44.1	45.0	43.8
		Ours	44.3	44.8	45.0	44.7
	Uni.	Orig.	43.2	43.5	43.8	43.5
		Ours	44.6	45.2	44.3	44.7

fra-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	50.3	51.3	51.1	50.9
		Ours	50.3	50.9	51.6	50.9
	Uni.	Orig.	50.1	51.1	51.6	50.9
		Ours	50.0	51.6	49.3	50.3
Morfessor	BPE	Orig.	48.9	48.8	48.9	48.9
		Ours	50.5	50.3	50.2	50.3
	Uni.	Orig.	48.6	49.0	48.5	48.7
		Ours	49.4	50.3	49.5	49.7

eng-fra			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	52.1	52.9	53.2	52.7
		Ours	52.5	52.3	52.9	52.6
	Uni.	Orig.	52.1	53.4	53.4	53.0
		Ours	51.6	53.0	51.0	51.9
Morfessor	BPE	Orig.	50.7	51.5	51.8	51.3
		Ours	51.8	52.0	53.1	52.3
	Uni.	Orig.	51.0	51.7	51.6	51.4
		Ours	52.2	52.8	52.3	52.4

nld-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	47.8	48.6	48.4	48.3
		Ours	48.2	47.4	48.1	47.9
	Uni.	Orig.	48.4	48.3	48.2	48.3
		Ours	47.7	48.6	48.4	48.3
Morfessor	BPE	Orig.	47.5	46.6	47.5	47.2
		Ours	47.4	47.7	47.5	47.6
	Uni.	Orig.	46.8	47.1	46.7	46.8
		Ours	47.5	47.2	47.2	47.3

eng-nld			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	47.4	47.9	47.6	47.6
		Ours	45.9	48.0	47.7	47.2
	Uni.	Orig.	46.3	48.1	47.4	47.3
		Ours	46.3	47.3	48.1	47.2
Morfessor	BPE	Orig.	45.7	45.5	46.6	45.9
		Ours	46.7	46.8	47.7	47.1
	Uni.	Orig.	46.1	46.0	46.1	46.1
		Ours	46.2	47.4	46.1	46.6

eng-ron			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	44.4	44.6	44.5	44.5
		Ours	44.7	45.3	45.1	45.1
	Uni.	Orig.	43.8	45.4	44.8	44.7
		Ours	44.3	45.1	44.5	44.6
Morfessor	BPE	Orig.	42.7	42.5	42.4	42.5
		Ours	42.9	44.6	44.5	44.0
	Uni.	Orig.	41.7	42.4	43.0	42.4
		Ours	43.6	44.2	43.8	43.9

ron-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	46.5	47.1	47.5	47.0
		Ours	47.5	47.0	48.5	47.7
	Uni.	Orig.	47.2	47.5	48.2	47.7
		Ours	46.5	46.6	47.6	46.9
Morfessor	BPE	Orig.	45.4	45.2	45.4	45.3
		Ours	46.3	46.8	47.1	46.7
	Uni.	Orig.	44.3	45.5	44.6	44.8
		Ours	46.4	46.0	46.7	46.4

eng-ita			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	46.7	46.8	47.7	47.1
		Ours	46.1	47.1	47.1	46.8
	Uni.	Orig.	46.7	48.2	47.8	47.6
		Ours	46.8	47.8	47.5	47.3
Morfessor	BPE	Orig.	46.2	45.8	45.6	45.9
		Ours	46.5	47.2	47.7	47.2
	Uni.	Orig.	45.7	46.1	45.7	45.8
		Ours	46.0	47.3	45.4	46.2

ita-eng			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	46.6	47.3	48.2	47.4
		Ours	46.7	47.5	47.4	47.2
	Uni.	Orig.	47.5	47.6	47.7	47.6
		Ours	46.5	47.7	47.6	47.3
Morfessor	BPE	Orig.	45.4	46.0	45.2	45.5
		Ours	46.2	46.5	46.9	46.5
	Uni.	Orig.	45.2	45.3	45.4	45.3
		Ours	46.5	46.7	46.2	46.5

ita-nld			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	36.3	35.8	36.3	36.1
		Ours	35.9	36.1	37.2	36.4
	Uni.	Orig.	35.9	36.2	37.0	36.3
		Ours	35.3	35.9	36.9	36.0
Morfessor	BPE	Orig.	35.4	35.4	34.3	35.1
		Ours	34.7	36.3	36.6	35.9
	Uni.	Orig.	35.0	35.2	35.6	35.3
		Ours	36.8	36.5	36.4	36.6

nld-ita			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	36.5	37.0	37.1	36.9
		Ours	36.3	36.7	37.8	36.9
	Uni.	Orig.	36.3	37.5	37.2	37.0
		Ours	36.8	35.5	37.3	36.6
Morfessor	BPE	Orig.	35.3	36.1	36.1	35.8
		Ours	37.4	36.8	37.9	37.4
	Uni.	Orig.	35.7	35.7	36.3	35.9
		Ours	36.7	36.8	36.2	36.6

ron-ita			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	40.2	41.1	40.7	40.7
		Ours	41.0	40.8	40.4	40.8
	Uni.	Orig.	40.0	40.1	40.5	40.2
		Ours	39.8	40.9	39.8	40.2
Morfessor	BPE	Orig.	39.4	38.8	38.6	38.9
		Ours	40.3	40.3	40.4	40.3
	Uni.	Orig.	39.0	39.4	39.1	39.2
		Ours	40.8	40.0	40.3	40.4

ita-ron			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	37.5	37.9	38.3	37.9
		Ours	37.8	38.1	38.0	37.9
	Uni.	Orig.	37.0	38.1	37.7	37.6
		Ours	37.6	37.7	37.6	37.6
Morfessor	BPE	Orig.	35.6	36.0	36.3	36.0
		Ours	38.2	38.2	37.8	38.1
	Uni.	Orig.	35.0	35.5	35.5	35.3
		Ours	37.4	37.3	37.0	37.2

ron-nld			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	36.0	35.9	37.0	36.3
		Ours	35.7	36.5	36.7	36.3
	Uni.	Orig.	35.7	36.0	36.5	36.1
		Ours	35.1	35.2	36.6	35.6
Morfessor	BPE	Orig.	34.7	35.9	35.2	35.3
		Ours	36.1	36.0	36.1	36.1
	Uni.	Orig.	35.3	34.6	35.1	35.0
		Ours	35.6	36.8	35.6	36.0

nld-ron			Vocabulary			Avg.
			4k	8k	16k	
Word-like	BPE	Orig.	33.6	33.5	34.7	33.9
		Ours	33.6	34.1	35.1	34.3
	Uni.	Orig.	33.4	34.7	35.0	34.4
		Ours	33.5	35.2	34.4	34.3
Morfessor	BPE	Orig.	32.7	32.9	33.1	32.9
		Ours	33.6	34.8	33.6	34.0
	Uni.	Orig.	32.3	32.8	32.3	32.5
		Ours	33.1	33.9	33.7	33.6

Table 11: The chrF scores for 18 language pairs of the IWSLT 2017. The blue-yellow scale is fit to the value range across each table.