PBoS: Probabilistic Bag-of-Subwords for Generalizing Word Embedding

Zhao Jinman Shawn Zhong Xiaomin Zhang Yingyu Liang
University of Wisconsin-Madison, Madison, WI, USA
{jz,yliang}@cs.wisc.edu
{shawn.zhong,xzhang682}@wisc.edu

Abstract

We look into the task of *generalizing* word embeddings: given a set of pre-trained word vectors over a finite vocabulary, the goal is to predict embedding vectors for out-of-vocabulary words, *without* extra contextual information. We rely solely on the spellings of words and propose a model, along with an efficient algorithm, that simultaneously models subword segmentation and computes subword-based compositional word embedding. We call the model probabilistic bag-of-subwords (PBoS), as it applies bag-of-subwords for all possible segmentations based on their likelihood. Inspections and affix prediction experiment show that PBoS is able to produce meaningful subword segmentations and subword rankings without any source of explicit morphological knowledge. Word similarity and POS tagging experiments show clear advantages of PBoS over previous subword-level models in the quality of generated word embeddings across languages.

1 Introduction

Word embeddings pre-trained over large texts have demonstrated benefits for many NLP tasks, especially when the task is label-deprived. However, many popular pre-trained sets of word embeddings assume fixed finite-size vocabularies [1, 2], which hinders their ability to provide useful word representations for out-of-vocabulary (OOV) words.

We look into the task of *generalizing* word embeddings: extrapolating a set of pre-trained word embeddings to words out of its fixed vocabulary, *without* extra access to contextual information (e.g. example sentences or text corpus). In contrast, the more common task of *learning* word embeddings, or often just *word embedding*, is to obtain distributed representations of words directly from large unlabeled text. The motivation here is to extend the usefulness of pre-trained embeddings without expensive retraining over large text.

There have been works showing that contextual information can also help generalize word embeddings (for example, Khodak et al., 2018; Schick and Schütze, 2019a,b). We here, however, focus more on the research question of how much one can achieve from just word compositions. In addition, our proposed way of utilizing word composition information can be combined with the contextual embedding algorithms to further improve the performance of generalized embeddings.

The hidden assumption here is that words are made of meaningful parts (cf. morphemes) and that the meaning of a word is related to the meaning of their parts. This way, humans are often able to guess the meaning of a word or term they have never seen before. For example, "postEMNLP" probably means "after EMNLP".

Different models have been proposed for that task of generalizing word embeddings using word compositions, usually under the name of subword(-level) models. Stratos (2017); Pinter et al. (2017); Kim et al. (2018b) model words at the character level. However, they have been surpassed by later subword-level models, probably because of putting too much burden on the models to form and discover meaningful subwords from characters. Bag-of-subwords (BoS) is a simple yet effective model for learning (Bojanowski et al., 2017) and generalizing (Zhao et al., 2018) word embeddings. BoS composes a word embedding vector by taking the sum or average of the vectors of the subwords (character n-grams) that appear in the given word. However, it ignores the importance of different subwords since all of them are given the

[1] https://code.google.com/archive/p/word2vec/, Mikolov et al. (2013).
[2] https://nlp.stanford.edu/projects/glove/, Pennington et al. (2014).

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 596–611
November 16 - 20, 2020.

same weight. Intuitively, "farm" and "land" should be more relevant in composing representation for word "farmland" than some random subwords like "armla".

Even more favorable would be a model's ability to discover meaningful subword segmentations on its own. Cotterell et al. (2016) bases their model over morphemes but needs help from an external morphological analyzer such as Morfessor (Virpioja et al., 2013). Sasaki et al. (2019) use trainable self-attention to combine subword vectors. While the attention implicitly facilitates interactions among subwords, there has been no explicit enforcement of mutual exclusiveness from subword segmentation, making it sometimes difficult to rule out less relevant subwords. For example, "her" is itself a likely subword, but is unlikely to be relevant for "higher" as the remaining "hig" is unlikely.

We propose the probabilistic bag-of-subwords (PBoS) model for generalizing word embedding. PBoS simultaneously models subword segmentation and composition of word representations out of subword representations. The subword segmentation part is a probabilistic model capable of handling ambiguity of subword boundaries and ranking possible segmentations based on their overall likelihood. For each segmentation, we compose a word vector as the sum of all subwords that appear in the segmentation. The final embedding vector is the expectation of the word vectors from all possible segmentations. An alternative view is that the model assigns word-specific weights to subwords based on how likely they appear as meaningful segments for the given word. Coupled with an efficient algorithm, our model is able to compose better word embedding vectors with little computational overhead compared to BoS.

Manual inspections show that PBoS is able to produce subword segmentations and subword weights that align with human intuition. Affix prediction experiment quantitatively shows that the subword weights given by PBoS are able to recover most eminent affixes of words with good accuracy.

To assess the quality of generated word embeddings, we evaluate with the intrinsic task of word similarity which relates to the semantics; as well as the extrinsic task of part-of-speech (POS) tagging which requires rich information to determine each word's role in a sentence. English word similarity experiment shows that PBoS improves the correlation scores over previous best models under various settings and is the only model that consistently improves over the target pre-trained embeddings. POS tagging experiment over 23 languages shows that PBoS improves accuracy compared in all but one language to the previous best models, often by a big margin.

We summarize our contributions as follows:

- We propose PBoS, a subword-level word embedding model that is based on probabilistic segmentation of words into subwords, the first of its kind (Section 2).
- We propose an efficient algorithm that leads to an efficient implementation [3]of PBoS with little overhead over previous much simpler BoS. (Section 3).
- Manual inspection and affix prediction experiment show that PBoS is able to give reasonable subword segmentations and subword weights (Section 4.1 and 4.2).
- Word similarity and POS tagging experiments show that word vectors generated by PBoS have better quality compared to previously proposed models across languages (Section 4.3 and 4.4).

2 PBoS Model

Following the above intuition, in this section we describe the PBoS model in detail.

We first develop a model that segments a word into subword and associates each subword segmentation with a likelihood based on the meaningfulness of each subword segment. We then apply BoS over each segmentation to compose a "segmentation vector". The final word embedding vector is then the probabilistic expectation of all the segmentation vectors. The subword segmentation and likelihood association part require no explicit source of morphological knowledge and are tightly integrated with the word vector composition part, which in turn gives rise to an efficient algorithm that considers *all* possible segmentations simultaneously (Section 3). The model can be trained by fitting a set of pre-trained word embeddings.

2.1 Terminology

For a given language, let Γ be its alphabet. A *word* w of length $l = |w|$ is a string made of l letters in Γ, i.e. $w = c_1 c_2 \dots c_l \in \Gamma^l$ where $w[i] = c_i$ is

[3]Code used for this work can be found at https://github.com/jmzhao/pbos.

the i-th letter. Let $p_w \in [0, 1]$ be the probability that w appears in the language. Empirically, this is proportional to the unigram frequency of word w observed in large text in that language.

Note that we do not assume a vocabulary. That is, we do not distinguish words from arbitrary strings made out of the alphabet. The implicit assumption here is that a "word" in common sense is just a string associated with high probability. In this sense, p_w can also be seen as the likelihood of string w being a "legit word". This blurs the boundary between words and non-words, and automatically enables us to handle unseen words, alternative spellings, typos, and nonce words as normal cases.

We say a string $s \in \Gamma^+$ is a *subword* of word w, denoted as $s \subseteq w$, if $s = w[i:j] = c_i \dots c_j$ for some $1 \le i \le j \le |w|$, i.e. s is a substring of w. The probability that subword s appears in the language can then be defined as

$$p_s \propto \sum_{w \in \Gamma^+} p_w \sum_{1 \le i \le j \le |w|} \mathbb{1}(s = w[i:j]) \quad (1)$$

where $\mathbb{1}(pred)$ gives 1 and otherwise 0 only if $pred$ holds. Note that a subword s may occur more than once in the same word w. For example, subword "ana" occurs twice in the word "banana".

A *subword segmentation* g of word w of length $k = |g|$ is a tuple $(s_1, s_2, \dots, s_k)$ of subwords of w, so that w is the concatenation of $s_1, \dots, s_k$.

2.2 Probabilistic Subword Segmentation

A subword transition graph for word w is a directed acyclic graph $G_w = (N_w, E_w)$. Let $l = |w|$. The vertices $N_w = \{0, \dots, l\}$ correspond to the positions between $w[i]$ and $w[i+1]$ for all $i \in [l-1]$, as well as to the beginning (vertiex 0) and the end (vertex l) of w. Each edge $(i, j) \in E_w = \{(i, j) : 0 \le i < j \le l\}$ corresponds to subword $w[i:j]$. We use G_w as a useful image for developing our model.

Proposition 1. *Paths from 0 to $|w|$ in G_w are in one-to-one correspondence to segmentations of w.*

Proposition 2. *There are $2^{|w|-1}$ different possible segmentations for word w.*

Each edge (i, j) is associated with a weight $p_{w[i:j]}$ — how likely $w[i:j]$ itself is a meaningful subword. We model the likelihood of segmentation g being a segmentation of w as being proportional to the product of all its subword likelihood – the transition along a path from 0 to $|w|$ in G_w:

$$p_{g|w} \propto \prod_{s \in g} p_s. \quad (2)$$

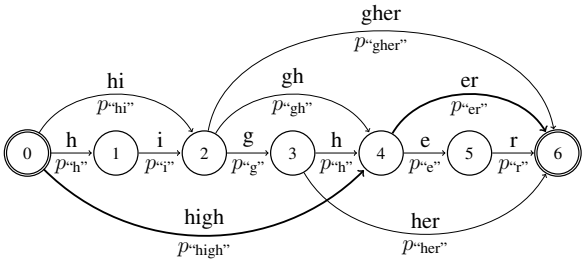

Figure 1: Diagram of probabilistic subwords transitions for word "higher". Some edges are omitted to reduce clutter. Each edge is labeled by a subword s of the word, associated with p_s. **Bold** edges constituent a path from node 0 to 6, corresponding to the segmentation of the word into "high" and "er".

Example. Figure 1 illustrates G_w for word $w =$ "higher" of length 6. Bold edges $(0, 4)$ and $(4, 6)$ form a path from 0 to 6, which corresponds to the segmentation ("high", "er"). The likelihood $p_{(\text{"high"},\text{"er"})|w}$ of this particular segmentation is proportional to $p_{\text{"high"}} p_{\text{"er"}}$ – the product of weights along the path. □

2.3 Probabilistic Bag-of-Subwords

Based on the above modeling of subword segmentations, we propose the Probabilistic Bag-of-Subword (PBoS) model for composing word embeddings.

The embedding vector $\boldsymbol{w}$ for word w is the expectation of all its segmentation-based word embedding:

$$\boldsymbol{w} = \sum_{g \in Seg_w} p_{g|w} \boldsymbol{g} \quad (3)$$

where $\boldsymbol{g}$ is the embedding for segmentation g.

Given a subword segmentation g, we adopt the Bag-of-Subwords (BoS) model (Bojanowski et al., 2017; Zhao et al., 2018) for composing word embedding from subwords. Specifically, we apply BoS [4] over the subword segments in g:

$$\boldsymbol{g} = \sum_{s \in g} \boldsymbol{s}, \quad (4)$$

where $\boldsymbol{s}$ is the vector representation for subword s, as if the current segmentation g is the "golden"

[4] Zhao et al. (2018) used averaging instead of summation. However, both give uniform weights to all subwords and result in vectors only differ by a scalar factor. We thus do not distinguish the two and refer to either of them as BoS.

segmentation of the word. In such case, we assume the meaning of the word is the combination of the meaning of all its subword segments. We maintain a look-up table $S : \Gamma^+ \to \mathbb{R}^d$ for all subword vectors (i.e. $\boldsymbol{s} = S(s)$) as trainable parameters of the model, where d is the embedding dimension.

Combining Eq. (3) and (4), we can compose vector representation for any word $w \in \Gamma^+$ as

$$\boldsymbol{w} = \sum_{g \in Seg_w} p_{g|w} \sum_{s \in g} \boldsymbol{s}. \tag{5}$$

Given a set of target pre-trained word vectors $\boldsymbol{w}^*$ defined for words within a finite vocabulary W, our model can be trained by minimizing the mean square loss:

$$\underset{S}{\text{minimize}} \frac{1}{|W|} \sum_{w \in W} \|\boldsymbol{w} - \boldsymbol{w}^*\|_2^2. \tag{6}$$

3 Efficient Algorithm

PBoS simultaneously considers all possible subword segmentations and their contributions in composing word representations. However, summing over embeddings of all possible segmentations can be awfully inefficient, as simply enumerating all possible segmentations of w takes number of steps exponential to the length of w (Proposition 2). We therefore need an efficient way to compute Eq. (5).

3.1 Alternative View: Weighted Subwords

Exchanging the order of summations in Eq. (5) from segmentation first to subword first, we get

$$\boldsymbol{w} = \sum_{s \subseteq w} a_{s|w} \boldsymbol{s} \tag{7}$$

where

$$a_{s|w} \propto \sum_{g \in Seg_w,\ g \ni s} p_{g|w} \tag{8}$$

is the weight accumulated over subword s, summing over all segmentations of w that contain s. [5]

Eq. (7) provides an alternative view of the word vector composed by our model: a weighted sum of all the word's subword vectors. Comparing to BoS, we assign different importance $a_{s|w}$, instead of a uniform weight, to each subword. $a_{s|w}$ can be viewed as the likelihood of subword s being a meaningful segment of the particular word w, considering both the likelihood of s itself being meaningful, and at the same time how likely the rest of the word can still be segmented into meaningful subwords.

Example. Consider the contribution of subword $s =$ "gher" in word $w =$ "higher". Possible contributions only come from segmentations that contain "higher": $g_1 =$ ("h", "i", "gher") and $g_2 =$ ("hi", "gher"). Each segmentation g adds weight $p_{g|w}$ to $a_{s|w}$. In this case, $a_{\text{"gher"}|w}$ will be smaller than $a_{\text{"er"}|w}$ because both $p_{g_1|w}$ and $p_{g_2|w}$ would be rather small. □

3.2 Computing Subword Weights

Now we can efficiently compute Eq. (7) if we can efficiently compute $a_{s|w}$. Here we present an algorithm that computes $a_{s|w}$ for all $s \subseteq w$ in $O(|w|^2)$ time.

The specific structure of the subword transition graph means that edges only go from left to right. Thus, we can split every path going through e into three parts: edges left to e, e itself and edges right to e. In terms of subwords, that is, for $s = w[i:j]$, $l = |w|$, each segmentation g that contains s can be divided into three parts: segmentation $g_{w[1:i-1]}$ over $w[1:i-1]$, subword s itself, and segmentation $g_{w[j+1:l]}$ over $w[j+1:l]$. Based on this, we can rewrite Eq. (8) as

$$a_{s|w} \propto \sum_{\substack{g \in Seg_w \\ g \ni s}} p_s \prod_{s' \in g_{w[1:i-1]}} p_{s'} \prod_{s' \in g_{w[j+1:l]}} p_{s'} \tag{9}$$

$$= p_s b_{1,i-1} b_{j+1,l}, \tag{10}$$

where $b_{i',j'} = \sum_{g' \in Seg_{w[i':j']}} \prod_{s' \in g'} p_{s'}$.

Now we can efficiently compute $a_{s|w}$ if we can efficiently compute $b_{1,i-1}$ and $b_{j+1,l}$ for all $1 \le i, j \le l$. Fortunately, we can do so for $b_{1,i}$ using the following recursive relation

$$b_{1,i} = \sum_{k=0}^{i-1} b_{1,k} p_{w[k+1:i]} \tag{11}$$

for $i = 1, \ldots, l$ with $b_{1,0} = 1$. Similar formulas hold for $b_{j,l}, j = 1, \ldots, l$ with $b_{l+1,l} = 1$.

Based on this, we devise Algorithm 1 for computing $a_{s|w}$ for all $s \subseteq w$. Here we take the alternative view of our model as a weighted average of all possible subwords (thus the normalization in Line 12), and an extension to the unweighted averaging of subwords as used in Zhao et al. (2018).

[5] For simplicity, here we assume all subwords are unique in w. A more careful index-based summation would model the general case but the idea remains the same. We take care of this in Algorithm 1.

Algorithm 1 Computing $a_{s|w}$.

1: **Input:** Word w, p_s for all $s \subseteq w$. $l = |w|$.
2: $b_{1,0} \leftarrow 1$; $b_{l+1,l} \leftarrow 1$;
3: **for** $i \leftarrow 1 \dots l$ **do**
4: $\quad b_{1,i} \leftarrow \sum_{k=0}^{i-1} p_{w[k+1:i]}\ b_{1,k}$
5: $\quad b_{l-i+1,l} \leftarrow \sum_{k=l-i+1}^{l} p_{w[l-i+1:k]}\ b_{k+1,l}$
6: **end for**
7: $\tilde{a}_{s|w} \leftarrow 0$ for all $s \subseteq w$
8: **for** $i \leftarrow 1 \dots l$, $j \leftarrow i \dots l$ **do**
9: $\quad \tilde{a}' \leftarrow p_{w[i:j]}\ b_{1,i-1}\ b_{j+1,l}$
10: $\quad \tilde{a}_{w[i:j]|w} \leftarrow \tilde{a}_{w[i:j]|w} + \tilde{a}'$
11: **end for**
12: $a_{s|w} \leftarrow \tilde{a}_{s|w} / \sum_{s' \subseteq w} \tilde{a}_{s'|w}$ for all $s \subseteq w$
13: **return** $a_{\cdot|w}$

Time complexity As we only access each subword once in each for-statement, the number of multiplications and additions involved is bounded by the number of subword locations of w. Each of Line 4 and Line 5 take i multiplications and $i-1$ additions respectively. So Line 3 to Line 6 in total takes $2l^2$ computations. Line 8 to Line 11 takes $\frac{3l(l+1)}{2}$ computations. Thus, the time complexity of Algorithm 1 is $O(l^2)$. Given a word of length 20, $O(l^2)$ ($20^2 = 400$) is much better than enumerating all $O(2^l)$ ($2^{20} = 1,048,576$) segmentations.

Using the setting in Section 4.3, PBoS only takes 30% more time (590 μs vs 454 μs) in average than BoS (by disabling $a_{s|w}$ computation) to compose a 300-dimensional word embedding vector.

4 Experiments

We design experiments to answer two questions: Do the segmentation likelihood and subword weights computed by PBoS align with their meaningfulness? Are the word embedding vectors generated by PBoS of good quality?

For the former, we inspect segmentation results and subword weights (Section 4.1), and see how good they are at predicting word affixes (Section 4.2). For the latter, we evaluate the word embeddings composed by PBoS at word similarity task (Section 4.3) and part-of-speech (POS) tagging task (Section 4.4).

Due to the page limit, we only report the most relevant settings and results in this section. Other details, including hardware, running time and detailed list of hyperparameters, can be found in Appendix A.

4.1 Subword Segmentation

In this subsection, we provide anecdotal evidence that PBoS is able to assign meaningful segmentation likelihood and subword weights.

Table 1 shows top subword segmentations and subsequent top subwords calculated by PBoS for some example word, ranked by their likelihood and weights respectively. The calculation is based on the word frequency derived from the Google Web Trillion Word Corpus [6]. We use the same list for word probability p_w throughout our experiments if not otherwise mentioned. All other settings are the same as described for PBoS in Section 4.3.

We can see the segmentation likelihood and subword weight favors the whole words as subword segments if the word appears in the word list, e.g. "higher", "farmland". This allows the model to closely mimic the word embeddings for frequent words that are probably part of the target vectors.

Second to the whole-word segmentation, or when the word is rare, e.g. "penpineanpplepie", "paradichlorobenzene", we see that PBoS gives higher likelihood to meaningful segmentations such as "high/er", "farm/land", "pen/pineapple/pie" and "para/dichlorobenzene"against other possible segmentations. [7] Subsequently, respective subword segments get higher weights among all possible subwords for the word, often by a good amount. This behavior would help PBoS to focus on meaningful subwords when composing word embedding. The fact that this can be achieved without any explicit source of morphological knowledge is itself interesting.

4.2 Affix Prediction

We quantitatively evaluate the quality of subword segmentations and subsequent subword weights by testing if our PBoS model is able to discover the most eminent word affixes. Note this has nothing to do with embeddings, so no training is involved in this experiment.

The affix prediction task is to predict the most eminent affix for a given word. For example, "-able" for "replaceable" and "re-" for "rename".

Models We get affix prediction from our PBoS by taking the top-ranked subword that is one of the possible affixes. To show our advantage, we

[6] https://www.kaggle.com/rtatman/english-word-frequency

[7] A slight exception is "farmlan/d", probably because "-d" is a frequent suffix.

Word w	Top segmentation g (and their $p_{g\|w}$)	Top subword s (and their $a_{s\|w}$)
higher	higher (0.924), high/er (0.030), highe/r (0.027), h/igher (0.007), hig/her (0.004).	higher (0.852), high (0.031), er (0.029), r (0.029), highe (0.025).
farmland	farmland (0.971), farmlan/d (0.010), farm/land (0.006), f/armland (0.005).	farmland (0.941), d (0.010), farmlan (0.009), farm (0.008), land (0.007).
penpineapplepie	pen/pineapple/pie (0.359), pen/pineapple/pi/e (0.157), pen/pineapple/p/ie (0.101).	pineapple (0.238), pen (0.186), pie (0.131), p (0.101), e (0.099).
paradichlorobenzene	para/dichlorobenzene (0.611), par/a/dichlorobenzene (0.110), paradi/chlorobenzene (0.083).	dichlorobenzene (0.344), para (0.283), a (0.061), par (0.054), ichlorobenzene (0.042).

Table 1: Top segmentations and subword weights by PBoS for some example words

Model	Precision	Recall	F1
BoS	0.493	0.465	0.425
PBoS	0.861	0.874	0.829

Table 2: Affix prediction results based on subword weights. All metrics are macro.

compare it with a BoS-style baseline affix predictor. Because BoS gives same weight to all subwords in a given word, we randomly choose one of the possible affixes that appear as subword of the word.

Benchmark We use the derivational morphology dataset [8] from Lazaridou et al. (2013). The dataset contains 7449 English words in total along with their most eminent affixes. Because no training is needed in this experiment, we use all the words for evaluation. To make the task more challenging, we drop trivial instances where there is only one possible affix appears as a subword in the given word. For example, "rename" is dropped because only prefix "re-" is present; on the other hand, "replaceable" is kept because both "re-" and "-able" are present. Besides excluding the trivial cases described above, we also exclude instances labeled with suffix "-y", because it is always included by "-ly" and "-ity". Altogether, we acquire 3546 words with 17 possible affixes for this evaluation.

Results Affix prediction results in terms of macro precision, recall, and F1 score are shown in Table 2. We can see a definite advantage of PBoS at predicting most word affixes, where all the metrics boost about 0.4 and F1 almost doubles compared to BoS, providing evidence that PBoS is able to assign meaningful subword weights.

4.3 Word Similarity

Given that PBoS is able to produce sensible segmentation likelihood and subword weights, we now turn our focus onto the quality of the generated word embeddings. In this section, we evaluate the word vectors' ability to capture word senses using the intrinsic task of word similarity.

Word similarity aims to test how well word embeddings capture words' semantic similarity. The task is given as pairs of words, along with their similarity scores labeled by language speakers. Given a set of word embeddings, we compute the similarity scores induced by the cosine distance between the embedding vectors of each pair of words. The performance is then measured in Spearman's correlation ρ for all pairs.

Benchmarks We use WordSim353 (WS) from Finkelstein et al. (2001) which mainly consists of common words. To better access models' ability to generalize word embeddings towards OOV words, we include the rare word datasets RareWord (RW) from Luong et al. (2013) and the newer Card-660 (Card) from Pilehvar et al. (2018).

Model Setup PBoS composes word embeddings out of subword vectors exactly as described in Section 3. Unlike some of previous models, we do not add special characters to indicate word boundaries and do not set any constraint on subword lengths. PBoS is trained 50 epochs using vanilla SGD with initial learning rate 1 and inverse square root decay.

For baselines, we compare against the bag-of-subword model (BoS) from Zhao et al. (2018), and the best attention-based model (KVQ-FH) from Sasaki et al. (2019). For BoS, we use our implementation by disabling subword weight computation. For KVQ-FH, we use the implementation given in the paper. All the hyperparameters are set the same as described in the original papers. We choose to not include the character-RNN model (MIMICK) from Pinter et al. (2017), as it has been shown clearly outperformed by the two.

[8] http://marcobaroni.org/PublicData/affix_complete_set.txt.gz

	WS	RW	Card
Polyglot: 100k tokens × 64 dim			
IV pairs	45	41	10
All pairs	36	10	5
OOV %	5%	58%	90%
Google: 160k tokens × 300 dim			
IV pairs	69	53	34
All pairs	68	45	10
OOV %	1%	11%	79%

Table 3: Target vectors statistics and word similarity performance measured in Spearman's $\rho \times 100$.

Model	# Param	WS	RW	Card
Target: Polyglot				
BoS	29.8M	34	**34**	6
KVQ-FH	7.8M	31	32	12
PBoS	37.8M	**41**	25	**15**
Target: Google News				
BoS	162.7M	61	48	11
KVQ-FH	36.2M	64	**49**	21
PBoS	315.7M	**68**	**49**	**25**

Table 4: Word similarity performance of subword-level models measured in Spearman's $\rho \times 100$.

Target Vectors We train all the subword models over English Polyglot vectors [9] and English Google News vectors [10]. Following the protocol of Zhao et al. (2018) and Sasaki et al. (2019), we clean and filter the words in Google vectors. Dimension of word vectors, number of words in target vectors are summarized in Table 3, along with their word similarity scores and OOV rate over the benchmarks. As we can see, both pre-trained embeddings yield decent correlations with human-labeled word similarity. However, the scores drop significantly as the OOV rate goes up. Polyglot vectors yield lower scores probably due to their smaller dimension and smaller token coverage.

Results Word similarity results of the three subword-level models are summarized in Table 4. [11] PBoS achieves scores better than or at least comparable to BoS and KVQ-FH in all but one of the six combinations of target vectors and word similarity benchmarks. Viewed as an extension to BoS, PBoS is in majority cases better than BoS, often by a good margin, suggesting the effectiveness of the subword weighting scheme. Compared to KVQ-FH, PBoS can often match and sometimes surpass it even though PBoS is a much simpler model with better explainability. Compared to the scores by using just the target embeddings (Table 3, All pairs), PBoS is the only model that demonstrates improvement across *all* cases.

The only case where PBoS is not doing well is with Polyglot vectors and RW benchmark. After many manual inspections, we conjecture that it may be related to the vector norm. Sometimes the vector of a relevant subword can be of a small norm, prone to be overwhelmed by less relevant subword vectors. To counter this, we tried to normalize subword vectors before summing them up into a word vector (PBoS-n). PBoS-n showed good improvement for the Polyglot RW case (25 to 32), matching the performance of the other two.

One may argue that PBoS has an advantage for using the most number of parameters. However, this is largely because we do not constrain the length of subwords as in BoS or use hashing as in KVQ-FH. In fact, restricting subword length and using hashing *helped* them for the word similarity task. We found that PBoS is insensitive to subword length constraints and decide to keep the setting simple. Despite being an interesting direction, we decide to not involve hashing in this work to focus on the effect of our unique weighting scheme.

FaxtText Comparison Albeit targeted for a different task (*training* word embedding) which have access to contextual information, the popular fastText (Bojanowski et al., 2017) also uses a subword-level model. We train fastText [12] over the same English corpus on which the Polyglot target vectors are trained, in order to understand the quantitative impact of contextual information. To ensure a fair comparison, we restrict the vocabulary sizes and embedding dimensions to match those of Polyglot vectors. The word similarity scores we get for the trained fastText model are 65/40/14 for WS/RW/Card. We note the great gain for WS and RW, suggesting the helpfulness of contextual information in learning and generalizing word embeddings in the setting of small to moderate OOV rates. Surprisingly, we find that for the case of extremely high OOV rate (Card), PBoS slightly surpasses fastText, suggesting PBoS' effectiveness in generalizing embeddings to OOV words even without any help from contexts.

[9] https://polyglot.readthedocs.io/en/latest/Download.html

[10] https://code.google.com/archive/p/word2vec/

[11] We regard training and prediction time as less of a concern here as all the three models are able to finish a training epoch in under a minute. Details and discussions can be found in Appendix A.2.

[12] https://github.com/facebookresearch/fastText/

Multilingual Results To evaluate and compare the effectiveness of PBoS across languages, we further train the models targeting multilingual Wikipedia2Vec vectors (Yamada et al., 2020) and evaluate them on multilingual WordSim353 and SemLex999 from Leviant and Reichart (2015) which are available in English, German, Italian and Russian. To better access the models' ability to *generalize*, we only take the top 10k words from the target vectors for training, which yields decent OOV rates, ranging from 23% to 84%. Detailed results can be found in Appendix Section A.3. In summary, we find 1) that PBoS surpasses KVQ-FH for English and German and is comparable to KVQ-FH for Italian; 2) that PBoS and KVQ-FH surpasses BoS for English, German and Italian; and 3) no definitive trend among the three models for Russian.

4.4 POS Tagging

We further assess the quality of generated word embedding via the extrinsic task of POS tagging. The task is to categorize each word in a given context into a particular part of speech, e.g. noun, verb, and adjective.

POS Tagging Model We follow the evaluation protocol for sequential labeling used by Kiros et al. (2015) and Li et al. (2017), and use logistic regression classifier [13] as the model for POS tagging. When predicting the tag for the i-th word w_i in a sentence, the input to the classifier is the concatenation of the vectors $\boldsymbol{w}_{i-2}, \boldsymbol{w}_{i-1}, \boldsymbol{w}_i, \boldsymbol{w}_{i+1}, \boldsymbol{w}_{i+2}$ for the word itself and the words in its context. This setup allows a more direct evaluation of the quality of word vectors themselves, and thus gives better discriminative power. [14]

Dataset We train and evaluate the performance of generated word embeddings over 23 languages at the intersection of the Polyglot (Al-Rfou' et al., 2013) pre-trained embedding vectors [15] and the Universal Dependency (UD, v1.4 [16]) dataset. Polyglot vectors contain 64-dimensional vectors over an 100k vocabulary for each language and are used as target vectors for each of the subword-level embedding models in this experiment. For PBoS, we use the Polyglot word counts for each language as the base for subword segmentation and subword weights calculation. UD is used as the POS tagging dataset to train and test the POS tagging model. We use the default partition of training and testing set. Statistics vary from language to language. See Appendix A.4 for more details.

Language	KVQ-FH	BoS	PBoS
Arabic	0.813	0.754	**0.905**(+0.092)
Basque	0.749	0.829	**0.866**(+0.037)
Bulgarian	0.777	0.793	**0.929**(+0.136)
Chinese	0.633	0.330	**0.833**(+0.200)
Czech	0.799	0.823	**0.917**(+0.094)
Danish	0.801	0.757	**0.904**(+0.103)
English	0.770	0.770	**0.896**(+0.126)
Greek	0.866	0.888	**0.941**(+0.053)
Hebrew	0.775	0.703	**0.915**(+0.140)
Hindi	0.811	0.800	**0.901**(+0.090)
Hungarian	0.777	0.794	**0.893**(+0.099)
Indonesian	0.776	0.828	**0.899**(+0.071)
Italian	0.794	0.787	**0.930**(+0.135)
Kazakh	0.623	0.753	**0.815**(+0.062)
Latvian	0.722	0.756	**0.848**(+0.092)
Persian	0.869	0.782	**0.924**(+0.056)
Romanian	0.774	0.755	**0.898**(+0.123)
Russian	0.775	0.838	**0.911**(+0.073)
Spanish	0.818	0.769	**0.920**(+0.102)
Swedish	0.826	0.840	**0.920**(+0.080)
Tamil	0.702	**0.758**	0.755(-0.003)
Turkish	0.760	0.777	**0.837**(+0.060)
Vietnamese	0.663	0.712	**0.832**(+0.121)

Table 5: POS tagging accuracy over 23 languages. In parentheses are the gains to the best of BoS and KVQ-FH.

Results Table 5 shows the POS tagging accuracy over the 23 languages that appear in both Polyglot and UD. All the subword-level embedding models follow the same hyperparameters as in Section 4.3. Following Sasaki et al. (2019), we tune the regularization term of the logistic regression model when evaluating KVQ-FH. Even with that, PBoS is able to achieve the best POS tagging accuracy in all but one language regardless of morphological types, OOV rates, and the number of training instances (Appendix Table 12). Particularly, PBoS improvement accuracy by greater than 0.1 for 9 languages. For the one language (Tamil) where PBoS is not the most accurate, the difference to the best is small (0.003). KVQ-FH gives no significantly more accurate predictions than BoS despite it is more complex and is the only one tuned with hyperparameters.

Overall, Table 5 shows that the word embeddings

[13] https://scikit-learn.org/0.19/modules/generated/sklearn.linear_model.LogisticRegression.html

[14] As a side note, in our early trials, we tried to evaluate using an LSTM model following Pinter et al. (2017) and Zhao et al. (2018), but found the numbers rather similar across embedding models. One possible explanation is that LSTMs are so good at picking up contextual features that the impact of mild deviations of a single word vector is marginal.

[15] https://polyglot.readthedocs.io/

[16] https://universaldependencies.org/

composed by our PBoS is effective at predicting POS tags for a wide range of languages.

5 Related Work

Popular word embedding methods, such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), often assume finite-size vocabularies, giving rise to the problem of OOV words.

FastText (Bojanowski et al., 2017; Joulin et al., 2017) attempted to alleviate the problem using subword-level model, and was followed by interests of using subword information to improve word embedding (Wieting et al., 2016; Cao and Lu, 2017; Li et al., 2017; Athiwaratkun et al., 2018; Li et al., 2018; Salle and Villavicencio, 2018; Xu et al., 2019; Zhu et al., 2019). Among them are Charagram by Wieting et al. (2016) which, albeit trained on specific downstream tasks, is similar to BoS followed by a non-linear activation, and the systematic evaluation by Zhu et al. (2019) over various choices of word composition functions and subword segmentation methods. However, all works above either pay little attention to the interaction among subwords inside a given word, or treat subword segmentation and composing word representation as separate problems.

Another interesting thread of works (Oshikiri, 2017; Kim et al., 2018a, 2019) attempted to model language solely at the subword level and learn subword embeddings directly from text, providing evidence to the power of subword-level models, especially as the notion of word is thought doubtful by some linguistics (Haspelmath, 2011).

Besides the recent interest in subwords, there have been long efforts of using morphology to improve word embedding (Luong et al., 2013; Cotterell and Schütze, 2015; Cui et al., 2015; Soricut and Och, 2015; Bhatia et al., 2016; Cao and Rei, 2016; Xu et al., 2018; Üstün et al., 2018; Edmiston and Stratos, 2018; Chaudhary et al., 2018; Park and Shin, 2018). However, most of them require an external oracle, such as Morfessor (Creutz and Lagus, 2002; Virpioja et al., 2013), for the morphological segmentations of input words, limiting their power to the quality and availability of such segmenters. The only exception is the character LSTM model by Cao and Rei (2016), which has shown some ability to recover the morphological boundary as a byproduct of learning word embedding.

The most related works in generalizing pretrained word embeddings have been discussed in Section 1 and compared throughout the paper.

6 Conclusion and Future Work

We propose PBoS model for generalizing pretrained word embeddings without contextual information. PBoS simultaneously considers all possible subword segmentations of a word and derives meaningful subword weights that lead to better composed word embeddings. Experiments on segmentation results, affix prediction, word similarity, and POS tagging over 23 languages support the claim.

In the future, it would be interesting to see if PBoS can also help with the task of *learning* word embedding, and how hashing would impact the quality of composed embedding while facilitating a more compact model.

Acknowledgments

The authors would like to thank anonymous reviewers of EMNLP for their comments. ZJ would like to thank Xuezhou Zhang, Sidharth Mudgal, Matt Du and Harit Vishwakarma for their helpful discussions.

References

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

Ben Athiwaratkun, Andrew Wilson, and Anima Anandkumar. 2018. Probabilistic FastText for multi-sense word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Melbourne, Australia. Association for Computational Linguistics.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 490–500, Austin, Texas. 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.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for*

NLP, pages 18–26, Berlin, Germany. Association for Computational Linguistics.

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime 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, Brussels, Belgium. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1292, Denver, Colorado. Association for Computational Linguistics.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660, Berlin, Germany. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

Qing Cui, Bin Gao, Jiang Bian, Siyu Qiu, Hanjun Dai, and Tie-Yan Liu. 2015. KNET: A general framework for learning word embedding using morphological knowledge. *ACM Trans. Inf. Syst.*, 34(1).

Daniel Edmiston and Karl Stratos. 2018. Compositional morpheme embeddings with affixes as functions and stems as arguments. In *Proceedings of the Workshop on the Relevance of Linguistic Structure in Neural Architectures for NLP*, pages 1–5, Melbourne, Australia. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 406–414, New York, NY, USA. Association for Computing Machinery.

Martin Haspelmath. 2011. The indeterminacy of word segmentation and the nature of morphology and syntax. *Folia linguistica*, 45(1):31–80.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Melbourne, Australia. Association for Computational Linguistics.

Geewook Kim, Kazuki Fukui, and Hidetoshi Shimodaira. 2018a. Word-like character n-gram embedding. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 148–152, Brussels, Belgium. Association for Computational Linguistics.

Geewook Kim, Kazuki Fukui, and Hidetoshi Shimodaira. 2019. Segmentation-free compositional n-gram embedding. 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 3207–3215, Minneapolis, Minnesota. Association for Computational Linguistics.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018b. Learning to generate word representations using subword information. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2551–2561, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1517–1526, Sofia, Bulgaria. Association for Computational Linguistics.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of*

the Second Workshop on Subword/Character LEvel Models, pages 38–48, New Orleans. Association for Computational Linguistics.

Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers, and Xiaoyong Du. 2017. Investigating different syntactic context types and context representations for learning word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2421–2431, Copenhagen, Denmark. Association for Computational Linguistics.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Takamasa Oshikiri. 2017. Segmentation-free word embedding for unsegmented languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 767–772, Copenhagen, Denmark. Association for Computational Linguistics.

Suzi Park and Hyopil Shin. 2018. Grapheme-level awareness in word embeddings for morphologically rich languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1401, Brussels, Belgium. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

Alexandre Salle and Aline Villavicencio. 2018. Incorporating subword information into matrix factorization word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 66–71, New Orleans. Association for Computational Linguistics.

Shota Sasaki, Jun Suzuki, and Kentaro Inui. 2019. Subword-based Compact Reconstruction of Word Embeddings. 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 3498–3508, Minneapolis, Minnesota. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2019a. Attentive mimicking: Better word embeddings by attending to informative contexts. 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 489–494, Minneapolis, Minnesota. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2019b. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6965–6973.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado. Association for Computational Linguistics.

Karl Stratos. 2017. Reconstruction of word embeddings from sub-word parameters. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 130–135, Copenhagen, Denmark. Association for Computational Linguistics.

Ahmet Üstün, Murathan Kurfalı, and Burcu Can. 2018. Characters or morphemes: How to represent words? In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 144–153, Melbourne, Australia. Association for Computational Linguistics.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline. D4 julkaistu kehittämis- tai tutkimusraportti tai -selvitys, Aalto University.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas. Association for Computational Linguistics.

Yang Xu, Jiawei Liu, Wei Yang, and Liusheng Huang. 2018. Incorporating latent meanings of morphological compositions to enhance word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1232–1242, Melbourne, Australia. Association for Computational Linguistics.

Yang Xu, Jiasheng Zhang, and David Reitter. 2019. Treat the word as a whole or look inside? subword embeddings model language change and typology. In *Proceedings of the 1st International Workshop on Computational Approaches to Historical Language Change*, pages 136–145, Florence, Italy. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 601–606, Brussels, Belgium. Association for Computational Linguistics.

Yi Zhu, Ivan Vulić, and Anna Korhonen. 2019. A systematic study of leveraging subword information for learning word 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 912–932, Minneapolis, Minnesota. Association for Computational Linguistics.

A Experimental Details

Here we list the details of our experiments that are omitted in the main paper due to space constraints.

We run all our experiments on a machine with an 8-core Intel i7-6700 CPU @ 3.40GHz, 32GB Memory, and GeForce GTX 970 GPU.

A.1 Hyperparameters

The meaning of hyperparameters shown in Table 6, Table 7 and Table 8 as explained as follows.

Subwords

- `min_len`: The minimum length for a subword to be considered.
- `max_len`: The maximum length for a subword to be considered.
- `word_boundary`: Whether to add special characters to annotate word boundaries.

Training

- `epochs`: The number of training epochs.
- `lr`: Learning rate.
- `lr_decay`: Whether to set learning rate to be inversely proportional to the square root of the epoch number.
- `normalize_semb`: Whether to normalize subword embeddings before composing word embeddings.
- `prob_eps`: Default likelihood for unknown characters.

Evaluation

- `C`: The inverse regularization term used by the logistic regression classifier.

A.2 Word Similarity

Table 6 and Table 8 show the hyperparameter values used in the word similarity experiment (Section 4.3). We transform all words in the benchmarks into lowercase, following the convention in FastText (Bojanowski et al., 2017; Joulin et al., 2017), BoS (Zhao et al., 2018), and KVQ-FH (Sasaki et al., 2019).

During the evaluation, we use 0 as the similarity score for a pair of words if we cannot get word vector for one of the words, or the magnitude of the word vector is too small. This is especially the case when we evaluate the target vectors, where OOV rates can be significant.

Table 9 lists experimental result for word similarity in greater detail.

Regarding the training epoch time, note that KVQ-FH uses GPU and is implemented using a deep learning library [17] with underlying optimized C code, whereas our PBoS is implemented using pure Python and uses only single thread CPU. We omit the prediction time for KVQ-FH, as we found it hard to separate the actual inference time from time used for other processes such as batching and data transfer between CPU and GPU. However, we believe the overall trend should be similar as for the training time.

One may notice that the prediction time for BoS in Table 9 is different from what was reported at the end of Section 3. This is largely because the BoS in Table 9 has a different (smaller) set of possible subwords to consider due to the subword length limits. In Section 3, to fairly access the impact of subword weights computation, we ensure that BoS and PBoS work with the same set of possible subwords (that used by PBoS in Section 4.3), and thus observe a slight longer prediction time for BoS.

A.3 Multilingual Word Similarity

We use Wikipedia2Vec (Yamada et al., 2020) as target vectors, and keep the most frequent 10k words to get decent OOV rates. The OOV rates and word similarity scores can be found in Table 10.

We do not clean or filter words as we did for the English word similarity, because we found it difficult to have a consistent way of pre-processing words across languages. For PBoS, we use the word frequencies from Polyglot for subword segmentation and subword weight calculation as the same for the multilingual POS tagging experiment (Section 4.4).

We evaluate all the models on multilingual WordSim353 (mWS) and SemLex999 (mSL) from Leviant and Reichart (2015), which is available for English, German, Italian and Russian. The dataset also contains the relatedness (rel) and similarity (sim) benchmarks derived from mWS.

We list the results for multilingual word similarity in Table 11.

A.4 POS Tagging

Table 7 and Table 8 show the hyperparameter values used in the POS tagging experiment (Section 4.4). For the prediction model, we use the logistic regression classifier from scikit-learn 0.19.1

[17] Chainer, https://chainer.org/

with the default settings.

Following the observation in Sasaki et al. (2019), we tune the regularization parameter $\mathtt{C}$ for KVQ-FH for all values $a \times 10^b$ where $a = 1, \ldots, 9$ and $b = -1, 0, \ldots, 4$. We use the POS tagging accuracy for English as criterion, and choose $\mathtt{C} = 70$.

Table 12 lists some statistics of the datasets used in the POS tagging experiment. PBoS is able to achieve better accuracy over BoS and KVQ-FH in all languages regardless of their morphological type, OOV rate and number of training instances for POS tagging.

Settings		Model		
		BoS	PBoS	PBoS-n
Subwords	min_len	3	1	1
	max_len	6	None	None
	word_boundary	True	False	False
Training	epochs	50	50	50
	lr	1.0	1.0	1.0
	lr_decay	True	True	True
	normalize_semb	False	False	True
	prob_eps	0.01	0.01	0.01

Table 6: Training settings used in word similarity experiment for BoS, PBoS, and PBoS-n

Settings		Model	
		BoS	PBoS
Subwords	min_len	3	1
	max_len	6	None
	word_boundary	True	False
Training	epochs	20	20
	lr	1.0	1.0
	lr_decay	True	True
	prob_eps	0.01	0.01
Evaluation	C	1	1

Table 7: Training settings used in POS tagging experiment for BoS and PBoS

Settings		Experiment	
		Word similarity	POS tagging
Subwords	min_len	3	3
	max_len	30	30
	word_boundary	True	True
Training	epochs	300	300
	limit_size	500,000	500,000
	bucket_size	40,000	40,000
Evaluation	C	N/A	70

Table 8: Training settings used in experiments for KVQ-FH.

Model	# Param	Dataset			Training Time		Prediction Time	
		WS	RW	Card	Total	Per epoch	Total	Per word
Target: Polyglot								
BoS	29.8M	34	**34**	6	505s	10.1s	1.9s	161μs
KVQ-FH	7.8M	31	32	12	2,669s	8.9s	–	–
PBoS	37.8M	**41**	25	**15**	966s	19.3s	4.2s	365μs
Target: Google News								
BoS	162.7M	61	48	11	1,110s	22.2s	4.8s	414μs
KVQ-FH	36.2M	64	**49**	21	10,638s	35.5s	–	–
PBoS	315.7M	**68**	**49**	**25**	2,065s	41.3s	6.8s	590μs

Table 9: Word similarity performance of subword-level models measured in Spearman's $\rho \times 100$, along with training and prediction time.

	mWS	mWS-rel	mWS-sim	mSL
English: 10k tokens × 300 dim				
IV pairs	65	56	71	26
All pairs	29	36	24	7
OOV	27%	23%	30%	36%
Germen: 10k tokens × 300 dim				
IV pairs	58	50	60	35
All pairs	8	14	7	7
OOV	54%	52%	55%	67%
Italian: 10k tokens × 300 dim				
IV pairs	52	50	54	24
All pairs	11	20	8	2
OOV	48%	45%	50%	54%
Russian: 10k tokens × 300 dim				
IV pairs	47	32	48	12
All pairs	1	4	2	9
OOV	73%	69%	75%	84%

Table 10: Multilingual target vectors statistics and word similarity performance measured in Spearman's $\rho \times 100$.

Model	# Param	mWS	mWS rel	mWS sim	mSL
English					
BoS	20.2M	32	29	34	23
KVQ-FH	36.0M	36	41	34	13
PBoS	30.4M	53	44	61	22
Germen					
BoS	21.3M	32	24	37	13
KVQ-FH	36.0M	18	19	19	14
PBoS	45.8M	38	30	38	12
Italian					
BoS	18.8M	8	-2	17	25
KVQ-FH	36.0M	19	22	21	9
PBoS	35.7M	25	16	27	13
Russian					
BoS	20.0M	20	15	21	14
KVQ-FH	36.0M	19	11	24	9
PBoS	35.6M	18	12	22	12

Table 11: Multilingual word similarity performance of subword-level models measured in Spearman's $\rho \times 100$.

Language	Morphological Type	OOV %	N_{train}	Model		
				KVQ-FH	BoS	PBoS
Arabic	Fusional	27.1%	225,853	0.813	0.754	**0.905**
Basque	Agglutinative	39.2%	72,974	0.749	0.829	**0.866**
Bulgarian	Fusional	33.7%	50,000	0.777	0.793	**0.929**
Chinese	Isolating	70.8%	98,608	0.633	0.330	**0.833**
Czech	Fusional	58.5%	1,173,282	0.799	0.823	**0.917**
Danish	Fusional	33.3%	88,980	0.801	0.757	**0.904**
English	Analytic	26.2%	204,587	0.770	0.770	**0.896**
Greek	Fusional	18.5%	47,449	0.866	0.888	**0.941**
Hebrew	Fusional	20.3%	135,496	0.775	0.703	**0.915**
Hindi	Fusional	27.1%	281,057	0.811	0.800	**0.901**
Hungarian	Agglutinative	29.2%	33,017	0.777	0.794	**0.893**
Indonesian	Agglutinative	20.0%	97,531	0.776	0.828	**0.899**
Italian	Fusional	24.3%	289,440	0.794	0.787	**0.930**
Kazakh	Agglutinative	22.8%	4,949	0.623	0.753	**0.815**
Latvian	Fusional	23.7%	13,781	0.722	0.756	**0.848**
Persian	Agglutinative	16.9%	121,064	0.869	0.782	**0.924**
Romanian	Fusional	29.4%	163,262	0.774	0.755	**0.898**
Russian	Fusional	31.3%	79,772	0.775	0.838	**0.911**
Spanish	Fusional	29.1%	382,436	0.818	0.769	**0.920**
Swedish	Analytic	37.4%	66,645	0.826	0.840	**0.920**
Tamil	Agglutinative	28.4%	6,329	0.702	**0.758**	0.755
Turkish	Agglutinative	37.8%	41,748	0.760	0.777	**0.837**
Vietnamese	Analytic	63.8%	31,800	0.663	0.712	**0.832**

Table 12: Statistics for the languages used in POS tagging experiment.
N_{train} is the number of training instances for the POS tagging model. OOV % is the percentage of the words in the POS tagging testing set that is out of the vocabulary of the Polyglot vectors in that language. Experimental results are included for convenience.