BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages

Benjamin Heinzerling and Michael Strube

AIPHES, Heidelberg Institute for Theoretical Studies

Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany

{benjamin.heinzerling | michael.strube}@h-its.org

Abstract

We present *BPEmb*, a collection of pre-trained subword unit embeddings in 275 languages, based on Byte-Pair Encoding (BPE). In an evaluation using fine-grained entity typing as testbed, BPEmb performs competitively, and for some languages better than alternative subword approaches, while requiring vastly fewer resources and no tokenization. BPEmb is available at https://github.com/bheinzerling/bpemb.

Keywords: subword embeddings, byte-pair encoding, multilingual

1. Introduction

Learning good representations of rare words or words not seen during training at all is a difficult challenge in natural language processing. As a makeshift solution, systems have typically replaced such words with a generic *UNK* token. Recently, based on the assumption that a word's meaning can be reconstructed from its parts, several subword-based methods have been proposed to deal with the unknown word problem: character-based recurrent neural networks (RNN) (Luong and Manning, 2016), character-based convolutional neural networks (CNN) (Chiu and Nichols, 2016), word embeddings enriched with subword information (FastText) (Bojanowski et al., 2017), and byte-pair encoding (BPE) (Sennrich et al., 2016), among others. While pre-trained FastText embeddings are publicly available, embeddings for BPE units are commonly trained on a per-task basis (e.g. a specific language pair for machine-translation) and not published for general use.

In this work we present *BPEmb*, a collection of pre-trained subword embeddings in 275 languages, and make the following contributions:

- We publish BPEmb, a collection of pre-trained byte-pair embeddings in 275 languages;

- We show the utility of BPEmb in a fine-grained entity typing task; and

- We show that BPEmb performs as well as, and for some languages better than, alternative approaches while being more compact and requiring no tokenization.

2. BPEmb: Byte-pair Embeddings

Byte Pair Encoding is a variable-length encoding that views text as a sequence of symbols and iteratively merges the most frequent symbol pair into a new symbol. E.g., encoding an English text might consist of first merging the most frequent symbol pair *t h* into a new symbol *th*, then merging the pair *th e* into *the* in the next iteration, and so on. The number of merge operations o determines if the resulting encoding mostly creates short character sequences (e.g. $o = 1000$) or if it includes symbols for many frequently

occurring words, e.g. $o = 30,000$ (cf. Table 1). Since the BPE algorithm works with any sequence of symbols, it requires no preprocessing and can be applied to untokenized text.

We apply BPE[1] to all Wikipedias[2] of sufficient size with various o and pre-train embeddings for the resulting BPE symbol using GloVe (Pennington et al., 2014), resulting in byte-pair embeddings for 275 languages. To allow studying the effect the number of BPE merge operations and of the embedding dimensionality, we provide embeddings for 1000, 3000, 5000, 10000, 25000, 50000, 100000 and 200000 merge operations, with dimensions 25, 50, 100, 200, and 300.

3. Evaluation: Comparison to FastText and Character Embeddings

To evaluate the quality of BPEmb we compare to Fast-Text, a state-of-the-art approach that combines embeddings of tokens and subword units, as well as to character embeddings.

FastText enriches word embeddings with subword information by additionally learning embeddings for character n-grams. A word is then represented as the sum of its associated character n-gram embeddings. In practice, representations of unknown word are obtained by adding the embeddings of their constituting character 3- to 6-grams. We use the pre-trained embeddings provided by the authors.[3]

Character embeddings. In this setting, mentions are represented as sequence of the character unigrams[4] they consist of. During training, character embeddings are learned for the k most frequent characters.

Fine-grained entity typing. Following Schütze (2017) and Yaghoobzadeh and Schütze (2017), we use fine-grained en-

---

[1] We use the SentencePiece BPE implementation: https://github.com/google/sentencepiece.

[2] We extract text from Wikipedia articles with WikiExtract (http://attardi.github.io/wikiextractor), lowercase all characters where applicable and map all digits to zero.

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

[4] We also studied character bigrams and trigrams. Results were similar to unigrams and are omitted for space.

Merge ops	Byte-pair encoded text
5000	豊 田 駅 (と よ だ え き) は 、 東京都 日 野 市 豊 田 四 丁目 にある
10000	豊 田 駅 (と よ だ えき) は 、 東京都 日 野市 豊 田 四 丁目にある
25000	豊 田駅 (と よ だ えき) は 、 東京都 日 野市 豊田 四 丁目にある
50000	豊 田駅 (と よ だ えき) は 、 東京都 日 野市 豊 田 四丁目にある
Tokenized	豊田 駅 （ と よ だ えき ） は 、 東京 都 日野 市 豊田 四 丁目 に ある
10000	豐 田 站 是 東 日 本 旅 客 鐵 道 (JR 東 日 本) 中央 本 線 的 鐵路 車站
25000	豐田 站是 東日本旅客鐵道 (JR 東日本) 中央 本 線的鐵路車站
50000	豐田 站是 東日本旅客鐵道 (JR 東日本) 中央 本線的 鐵路車站
Tokenized	豐田站 是 東日本 旅客 鐵道 （ JR 東日本 ） 中央本線 的 鐵路車站
1000	to y od a _station is _a _r ail way _station _on _the _ch ū ō _main _l ine
3000	to y od a _station _is _a _railway _station _on _the _ch ū ō _main _line
10000	toy oda _station _is _a _railway _station _on _the _ch ū ō _main _line
50000	toy oda _station _is _a _railway _station _on _the _chū ō _main _line
100000	toy oda _station _is _a _railway _station _on _the _chūō _main _line
Tokenized	toyoda station is a railway station on the chūō main line

Table 1: Effect of the number of BPE merge operations on the beginning of the Japanese (top), Chinese (middle), and English (bottom) Wikipedia article TOYODA_STATION. Since BPE is based on frequency, the resulting segmentation is often, but not always meaningful. E.g. in the Japanese text, 豊 (toyo) and 田 (ta) are correctly merged into 豊田 (Toyoda, a Japanese city) in the second occurrence, but the first 田 is first merged with 駅 (eki, *train station*) into the meaningless 田駅 (ta-eki).

tity typing as test bed for comparing subword approaches. This is an interesting task for subword evaluation, since many rare, long-tail entities do not have good representations in common token-based pre-trained embeddings such as word2vec or GloVe. Subword-based models are a promising approach to this task, since morphology often reveals the semantic category of unknown words: The suffix *-shire* in *Melfordshire* indicates a location or city, and the suffix *-osis* in *Myxomatosis* a sickness. Subword methods aim to allow this kind of inference by learning representations of subword units (henceforth: SUs) such as character ngrams, morphemes, or byte pairs.

Method. Given an entity mention m such as *Melfordshire*, our task is to assign one or more of the 89 fine-grained entity types proposed by Gillick et al. (2014), in this case /location and /location/city. To do so, we first obtain a subword representation

$$s = SU(m) \in R^{l \times d}$$

by applying one of the above SU transformations resulting in a SU sequence of length l and then looking up the corresponding SU embeddings with dimensionality d. Next, s is encoded into a one-dimensional vector representation

$$v = A(s) \in R^d$$

by an encoder A. In this work the encoder architecture is either averaging across the SU sequence, an LSTM, or a CNN. Finally, the prediction y is:

$$y = \frac{1}{1 + exp(-v)}$$

(Shimaoka et al., 2017).

Data. We obtain entity mentions from Wikidata (Vrandečić and Krötzsch, 2014) and their entity types by mapping to Freebase (Bollacker et al., 2008), resulting

in 3.4 million English[5] instances like (*Melfordshire*: /location,/location/city). Train and test set are random subsamples of size 80,000 and 20,000 or a proportionally smaller split for smaller Wikipedias. In addition to English, we report results for a) the five languages having the largest Wikipedias as measured by textual content; b) Chinese and Japanese, i.e. two high-resource languages without tokenization markers; and c) eight medium- to low-resource Asian languages.

Experimental Setup. We evaluate entity typing performance with the average of strict, loose micro, and loose macro precision (Ling and Weld, 2012). For each combination of SU and encoding architecture, we perform a Tree-structured Parzen Estimator hyper-parameter search (Bergstra et al., 2011) with at least 1000 hyper-parameter search trials (English, at least 50 trials for other languages) and report score distributions (Reimers and Gurevych, 2017). See Table 3 for hyper-parameter ranges.

4. Results and Discussion

4.1. Subwords vs. Characters vs. Tokens

Figure 1 shows our main result for English: score distributions of 1000+ trials for each SU and architecture. Token-based results using two sets of pre-trained embeddings (Mikolov et al., 2013; Pennington et al., 2014) are included for comparison.

Subword units. BPEmb outperforms all other subword units across all architectures (BPE-RNN mean score 0.624 ± 0.029, max. 0.65). FastText performs slightly worse (FastText-RNN mean 0.617 ± 0.007, max. 0.63)[6],

---
[5] Numbers for other languages omitted for space.

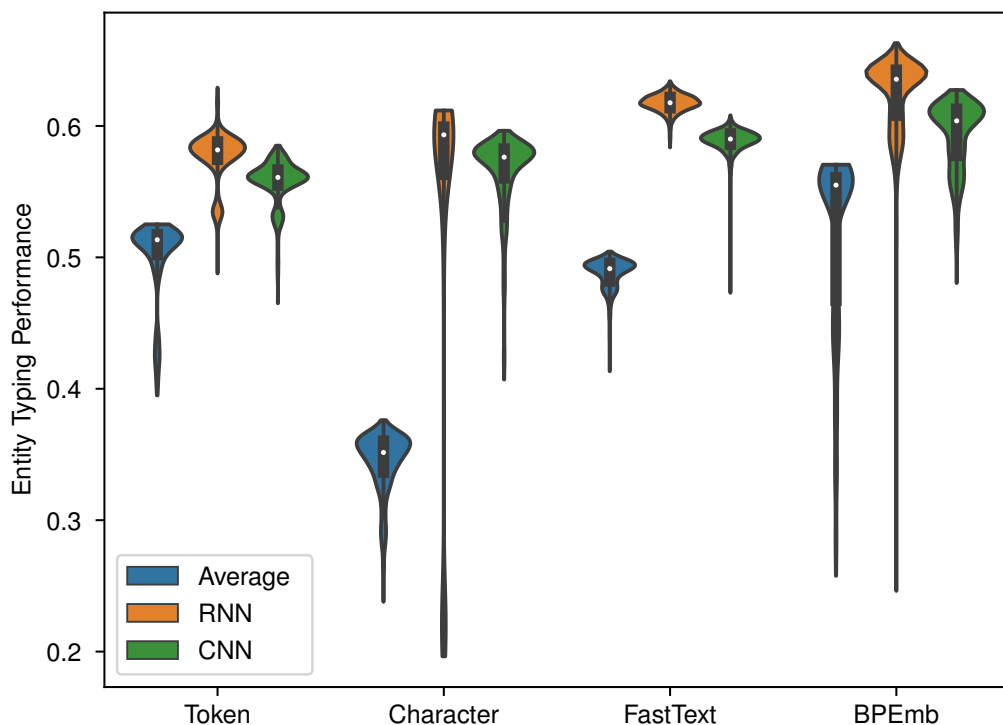[6] Difference to BPEmb significant, $p < 0.001$, Approximate Randomization Test.

Figure 1: English entity typing performance of subword embeddings across different architectures. This violin plot shows smoothed distributions of the scores obtained during hyper-parameter search. White points represent medians, boxes quartiles. Distributions are cut to reflect highest and lowest scores.

even though the FastText vocabulary is much larger than the set of BPE symbols.

BPEmb performs well with low embedding dimensionality Figure 2, right) and can match FastText with a fraction of its memory footprint (6 GB for FastText's 3 million embeddings with dimension 300 vs 11 MB for 100k BPE embeddings (Figure 2, left) with dimension 25.). As both Fast-Text and BPEmb were trained on the same corpus (namely, Wikipedia), these results suggest that, for English, the compact BPE representation strikes a better balance between learning embeddings for more frequent words and relying on compositionality of subwords for less frequent ones.

FastText performance shows the lowest variance, i.e., it robustly yields good results across many different hyper-parameter settings. In contrast, BPEmb and character-based models show higher variance, i.e., they require more careful hyper-parameter tuning to achieve good results.

Architectures. Averaging a mention's associated embeddings is the worst architecture choice. This is expected for character-based models, but somewhat surprising for token-based models, given the fact that averaging is a common method for representing mentions in tasks such as entity typing (Shimaoka et al., 2017) or coreference resolution (Clark and Manning, 2016). RNNs perform slightly better than CNNs, at the cost of much longer training time.

Language	FastText	BPEmb	Δ
English	62.9	**65.4**	2.5
German	65.5	**66.2**	0.7
Russian	**71.2**	70.7	-0.5
French	**64.5**	63.9	-0.6
Spanish	**66.6**	66.5	-0.1
Chinese	71.0	**72.0**	1.0
Japanese	**62.3**	61.4	-0.9
Tibetan	37.9	**41.4**	3.5
Burmese	**65.0**	64.6	-0.4
Vietnamese	81.0	81.0	0.0
Khmer	**61.5**	52.6	-8.9
Thai	63.5	**63.8**	0.3
Lao	44.9	**47.0**	2.1
Malay	75.9	**76.3**	0.4
Tagalog	**63.4**	62.6	-1.2

Table 2: Entity typing scores for five high-resource languages (top), two high-resource languages without explicit tokenization, and eight medium- to low-resource Asian languages (bottom).

4.2. Multilingual Analysis

Table 2 compares FastText and BPEmb across various languages. For high-resource languages (top) both approaches
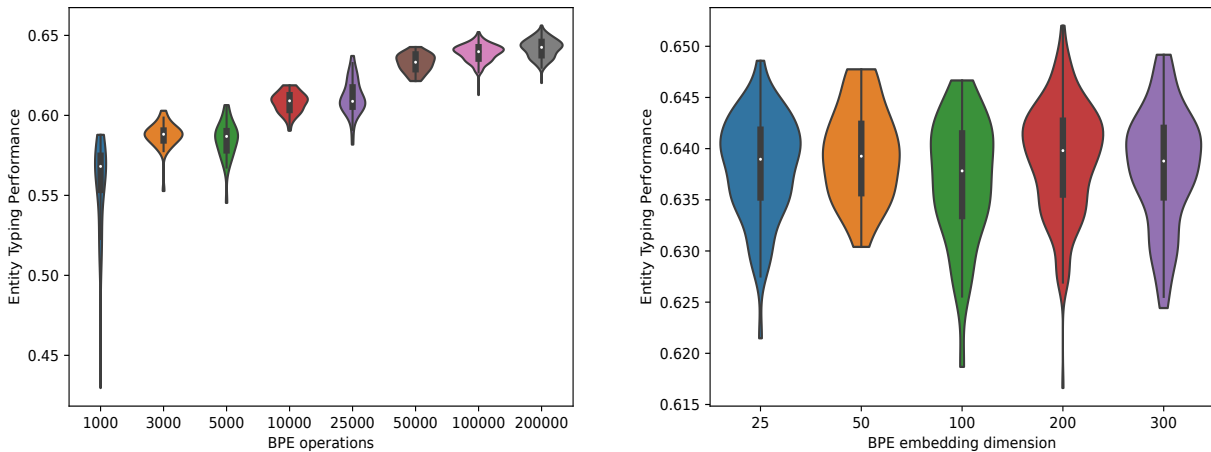
Figure 2: Impact of the number of BPE merge operations (left) and embedding dimension (right) on English entity typing.

perform equally, with the exception of BPEmb giving a significant improvement for English. For high resources languages without explicit tokenization (middle), byte-pair encoding appears to yield a subword segmentation which gives performance comparable to the results obtained when using FastText with pre-tokenized text[7].

Results are more varied for mid- to low-resource Asian languages (bottom), with small BPEmb gains for Tibetan and Lao. The large performance degradation for Khmer appears to be due to inconsistencies in the handling of unicode control characters between different software libraries used in our experiments and have a disproportionate effect due to the small size of the Khmer Wikipedia.

5. Limitations

Due to limited computational resources, our evaluation was performed only for a few of the 275 languages provided by BPEemb. While our experimental setup allows a fair comparison between FastText and BPEmb through extensive hyper-parameter search, it is somewhat artificial, since it disregards context. For example, *Myxomatosis* in the phrase *Radiohead played Myxomatosis* has the entity type /other/music, which can be inferred from the contextual music group and the predicate *plays*, but this ignored in our specific setting. How our results transfer to other tasks requires further study.

6. Replicability

All data used in this work is freely and publicly available. BPEmb and code to replicate our experiments is available at https://github.com/bheinzerling/bpemb.

7. Conclusions

We presented BPEmb, a collection of subword embeddings trained on Wikipedias in 275 languages. Our evaluation showed that BPEmb performs as well as, and for some

languages, better than other subword-based approaches. BPEmb requires no tokenization and is orders of magnitudes smaller than alternative embeddings, enabling potential use under resource constraints, e.g. on mobile devices.

Acknowledgments

Unit	Hyper-parameter	Space
Token	embedding type	GloVe, word2vec
Character	vocabulary size	50, 100, 200, 500, 1000
	embedding dimension	10, 25, 50, 75, 100
FastText	-	-
BPE	merge operations	1k, 3k, 5k, 10k, 25k 50k, 10k, 200k
	embedding dimension	25, 50, 100, 200, 300

Architecture	Hyper-parameter	Space
RNN	hidden units	100, 300, 500, 700, 1000, 1500, 2000
	layers	1, 2, 3
	RNN dropout	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	output dropout	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
CNN	filter sizes	(2), (2, 3), (2, 3, 4), (2, 3, 4, 5), (2, 3, 4, 5, 6), (3), (3, 4), (3, 4, 5), (3, 4, 5, 6), (4), (4, 5), (4, 5, 6), (5), (5, 6), (6)
	number of filters	25, 50, 100, 200, 300, 400, 500, 600, 700
	output dropout	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
Average	output dropout	0.0, 0.1, 0.2, 0.3, 0.4, 0.5

Table 3: Subword unit (top) and architecture (bottom) hyper-parameter space searched.

---

[7]Tokenization for Chinese was performed with Stanford CoreNLP (Manning et al., 2014) and for Japanese with Kuromoji (https://github.com/atilika/kuromoji).

8. Bibliographical References

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data,* Vancouver, B.C., Canada, 10–12 June 2008, pages 1247–1250.

Chiu, J. and Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Clark, K. and Manning, C. D. (2016). Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016.

Gillick, D., Lazic, N., Ganchev, K., Kirchner, J., and Huynh, D. (2014). Context-Dependent Fine-Grained Entity Type Tagging. *ArXiv e-prints*, December.

Ling, X. and Weld, D. S. (2012). Fine-grained entity recognition. In *AAAI*.

Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, August. Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the ICLR 2013 Workshop Track*.

Pennington, J., Socher, R., and Manning, C. (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, October. Association for Computational Linguistics.

Reimers, N. and Gurevych, I. (2017). Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *CoRR*, abs/1707.09861.

Schütze, H. (2017). Nonsymbolic text representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 785–796, Valencia, Spain, April. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (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, August. Association for Computational Linguistics.

Shimaoka, S., Stenetorp, P., Inui, K., and Riedel, S. (2017). Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1271–1280, Valencia, Spain, April. Association for Computational Linguistics.

Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Yaghoobzadeh, Y. and Schütze, H. (2017). Multi-level representations for fine-grained typing of knowledge base entities. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 578–589, Valencia, Spain, April. Association for Computational Linguistics.