Representing Words with Only Subword Information

Alexia Wenxin Xu
Department of Computer Science
alexiaxu@stanford.edu

Abstract

Continuous word representations are the key features in many natural language processing tasks. The recent model introduced by Bojanowski *et al.* [2] takes into account the internal structure of words and represents them as a bag of character n-grams. There are two possible limitations in the model: 1) Using the n-grams, word vectors may be redundant representing a word. In addition, the existence of the word vectors may prevent the n-gram vectors from being fully trained 2) Each character n-gram may occur in multiple words, and they could weigh differently in different words. To address the issues, we propose a model that represents a word as the sum of the attention of the n-grams without using the word vector. Our word representation outperforms the model in [2] in unsupervised tasks and is on par with it in text classification. We also show how a well-trained model segments the words and judges the importance of n-grams.

1 Introduction

Continuous word representations are the key features in many tasks in natural language processing, such as text classification, machine translation, and information retrieval. Back to the 1990s, word representations have been derived from statistical language modeling [12, 13]. More recently, Mikolov *et al.* [1] proposed a skip-gram model as an efficient method to learn vector representations of words from large amounts of unstructured text data.

Most of the previous models represent each word as one vector without utilizing the internal structure of the word. Bojanowski *et al.* [2] proposed a model that learns representations for character n-grams. It represents each word as the sum of the n-gram vectors and the word vector itself. By taking into account the subword information, it improves vector representations for not only English, but also morphologically rich languages like Turkish or Finnish that contain many word forms that rarely occur in the training corpus.

In this paper, we propose to improve the model of Bojanowski *et al.* [2] based on two ideas: 1) Word vectors may be redundant in existence of the character n-grams. N-grams should carry sufficient information to represent the words. When the word vectors present, the n-gram vectors may not be fully trained. 2) Each character n-gram may occur in multiple words, and they could weigh differently in those words. For example, ”her” carries significant semantic information in ”herself”, but not in ”inherit”. Ideally, the model should allow an n-gram to vary in significance in different words. To address the two issues, we built a model that represents a word as solely the sum of the attention of the n-grams. To be consistent with [2], we also evaluate our model on both the correlations with human similarity judgment and text classification tasks. We will also show how a well-trained model segments the words and judges the importance of the n-grams.

2 Related Work

There have been many successful previous works on training word embedding. The creation of word2vec [14] allows large improvements in accuracy at much lower computational cost. A year later, Pennington *et al.* [15] introduced GloVe to us, which is a competitive set of pre-trained embeddings that combines the advantages of the two major model families: global matrix factorization and local context window methods. In recent years, there are also works proposing to incorporate morphological information into word representations. [6], [7], and [16] presented approaches to derive representations of words from morphemes.

In 2017, Bojanowski *et al.* [2] proposed a new approach based on the skip-gram model [1], where each word is represented as a bag of character n-grams. Joulin *et al.* [3] built a fast text classifier (fastText) based on the skip-gram model of [2]. fastText is often on par with deep learning classifiers in terms of accuracy and is many orders of magnitude faster for training and evaluation.

Our model was built upon the model in [2] by representing words using only character n-grams. Our model outperforms the model in [2] in correlations with human judgment on word similarity and is competitive with fastText in sentiment analysis.

3 Approach

In this section, we will describe how our model learns word representations by gathering morphological information in subwords. In the first subsection, we will show the general framework and the error metric in training word embeddings. In the second subsection, we will present the attention-based model to generate word vectors from subwords. At last, we will present the modifications to the model in training supervised tasks.

3.1 General Framework

The model described in this subsection is the skig-gram model with negative sampling proposed in [1] and [2]. We trained a skip-gram model to learn word representations, where each word vector is trained to predict the words in its context. Given a corpus with word sequence $\{w_1, w_2, ..., w_T\}$, the objective of the skip-gram model is to maximize the following likelihood:

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} logp(w_c|w_t),$$

where $\mathcal{C}_t$ is the context of word w_t, which consists of indices of words surrounding w_t. Assuming we have a similarity function $s(w_t, w_c) \in \mathbb{R}$ that maps the two word vectors to a scalar score, we could define $p(w_c|w_t)$ as follows:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{W} e^{s(w_t,j))}}$$

However, there are two problems using softmax as the objective function: 1) each center word has more than one context words; 2) to compute the softmax, it needs to calculate the probability of each word in the vocabulary, which is computationally inefficient. In practice, we use negative sampling loss as the objective function in learning word representations. We aim to maximize the score of true context words, while minimizing that of the random negative samples. Thus we have the following loss function:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{T} \Bigg[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \Bigg],$$

where $\mathcal{N}_{t,c}$ is a random set of negative examples sampled from the vocabulary, and $\ell(x) = log(1 + e^{-x})$. Each word has two vectors. u_{w_t} is the input vector that represent the word when it is the center, and v_{w_c} is the output vector used when the word is in the context. In [1], $s(w_t, w_c) = \boldsymbol{\mu}_{w_t}^{\top} \boldsymbol{\nu}_{w_c}$, while we will re-define $s(w_t, w_c)$ by inducing subword information and attention in the next subsection.

3.2 Attention-based Subword Model

Bojanowski *et al.* [2] proposed to represent a word w by summing up its word vector with its n-gram subword vectors. They hope to take into account the internal structure of the words by using subword information. We believe that the n-gram subwords carry enough information to represent a word both syntactically and semantically. Thus the word vector of w itself is not necessary in the model. Another important point is that the n-grams of a word vary in significance in representing the word. "kind" in "kindness", for example, carries more semantic information than "ndnes". Similarly, one n-gram could occur in multiple words in the vocabulary, and it could weigh differently in representing different words. In this section, we will propose an attention-based subword model to address these issues.

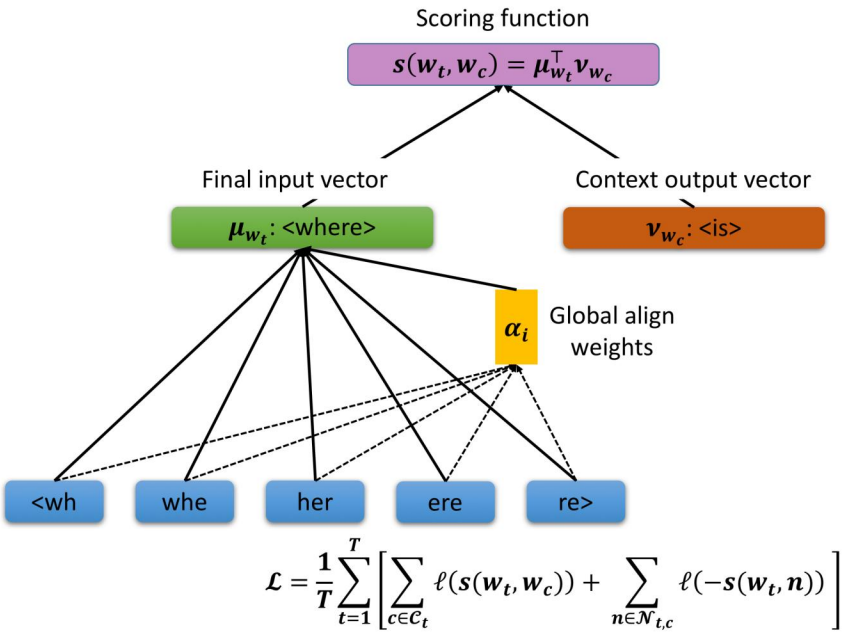

Figure 1: The attention-based skip-gram model with negative sampling, assuming only 3-gram subwords are used. The blue boxes show all character 3-grams of the word $\langle where \rangle$

As shown in Figure 1, each word w is represented as the sum of the attention over its character n-grams in our model. Similar to [2], we add special symbols $\langle$ and $\rangle$ at the beginning and the end of each word. These symbols help distinguish the word $\langle her \rangle$ from the "her" in $\langle inherit \rangle$. In practice, we extracted all the character n-grams ($n \in [3, 6]$) in computing the input vector $\boldsymbol{\mu}_{w_t}$.

Our model includes a trainable parameter, called bias, for each n-gram to address their different contribution to a word. Given a word w, let's denote the set of n-grams ($n \in [3, 6]$) in w by $\mathcal{G}_w$, the vector representing n-gram $g \in \mathcal{G}_w$ by z_g, and the trainable bias of g by b_g. We define the scoring function as:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} softmax(b_g) \boldsymbol{z}_g^\top \boldsymbol{\nu}_c,$$

The attention module enables sharing n-grams across words while weighing them differently.

As in [2], we use a hash function that maps n-grams to integers in $[1, 2 \times 10^6]$ to bound the memory. In this setting, each word is represented as a list of n-gram indices.

3.3 Text Classification

We also tested our model on a supervised task, sentiment analysis, as in [3]. In sentiment analysis, we represent a sentence as the mean of the bigrams it contains, which means every pair of adjacent words in the sentence, and denote the sentence vector as $\boldsymbol{\mu}_s$. We denote the set of the bigrams by $\mathcal{G}_s$. To embed a bigram, we first transform it into a set of character n-grams, denoted by $\mathcal{G}_c$. Then we represent the word bigram as the sum of the attention of the character n-grams, and denote the bigram vector by $\boldsymbol{\mu}_{bigram}$. Labels are also embedded as vectors, denoted by $\boldsymbol{\nu}_L$. The scoring function of each (sentence, label) pair is calculated as the dot product of the two vectors. We use softmax cross-entropy as the loss function:

$$\mathcal{L} = CE(\boldsymbol{y}, softmax(\boldsymbol{\mu}_s^\top \boldsymbol{\nu}_L)),$$

where

$$\boldsymbol{\mu}_s = \frac{1}{|\mathcal{G}_s|} \sum_{bigram \in \mathcal{G}_s} \boldsymbol{\mu}_{bigram},$$

$$\boldsymbol{\mu}_{bigram} = \sum_{g \in \mathcal{G}_c} softmax(b_g) \boldsymbol{z}_g$$

All models were trained asynchronously on multiple CPUs using stochastic gradient descent and a linearly decaying learning rate.

4 Experiments

4.1 Datasets

Training word embeddings We trained our model respectively on enwik8 (also called text8)[1], enwik9 (also called text9)[2], and the English Wikipedia data released by Shaoul and Westbury (2010) [4]. We normalize all datasets using Matt Mahoney's preprocessing perl script [3]. All the datasets are shuffled. We evaluate the quality of our representations on the task of word similarity / relatedness. In evaluation, we use the WS353 dataset introduced by Finkelstein *et al.* [5] and the rare word dataset (RW), introduced by Luong *et al.* [6].

Text classification We employ the same 8 datasets and evaluation protocol as [3], which are the Ag News, Sogou News, DBpedia, Yelp Review Polarity, Yelp Review Full, Yahoo Answers, Amazon Review Full, and Amazon Review Polarity.

4.2 Experimental Setup

To compare with [2], we adopt the same experimental setup. In unsupervised tasks, the dimension of the word representations is 300. The sizes of the n-grams range from 3 to 6. In negative sampling, 5 negative examples are sampled at random for each positive example, with probability proportional to the square root of the uni-gram frequency. When building the word dictionary, we keep the words that appear at least 5 times in the training set. The ceiling of the hash function is 2×10^6. We solve the optimization problem by performing stochastic gradient descent on the loss function, and use a linear decay of the step size. Each dataset is trained for 5 epochs. In text classification tasks, we set the dimension of the vectors to 10. The initial learning rate of the eight datasets were $\{$0.25 0.5 0.5 0.1 0.1 0.1 0.05 0.05$\}$.

5 Results

For simplicity, we will refer to the model of [2] and [3] as fastText (the name of the library based on [2] and [3]) and our attention-based model as fastAttn in this section.

[1] http://mattmahoney.net/dc/enwik8.zip

[2] http://mattmahoney.net/dc/enwik9.zip

[3] http://mattmahoney.net/dc/textdata.html

5.1 Effect of the Corpus Size

In this subsection, we explore the effect of the size of the training sets. By training on a large set, n-gram vectors capture the semantics and syntax more precisely. On the other hand, training becomes much slower. To balance between training time and quality, we trained our model on enwik8 (17M words), enwik9 (124M words), and the Shaoul Wikipedia (990M words) for 5 epochs, respectively. The result is shown in Figure 2. We evaluate the models on the rare word dataset (RW) by calculating the correlation between human judgment and similarity scores. We observe that the performance of the model trained on enwik9 is similar to that on wikipedia (the correlation score is ~ 45 vs. ~ 47 on wikipedia), but it takes less time to train. We will use the models trained on enwik9 to generate the results in Section 5.2 and Section 5.4.

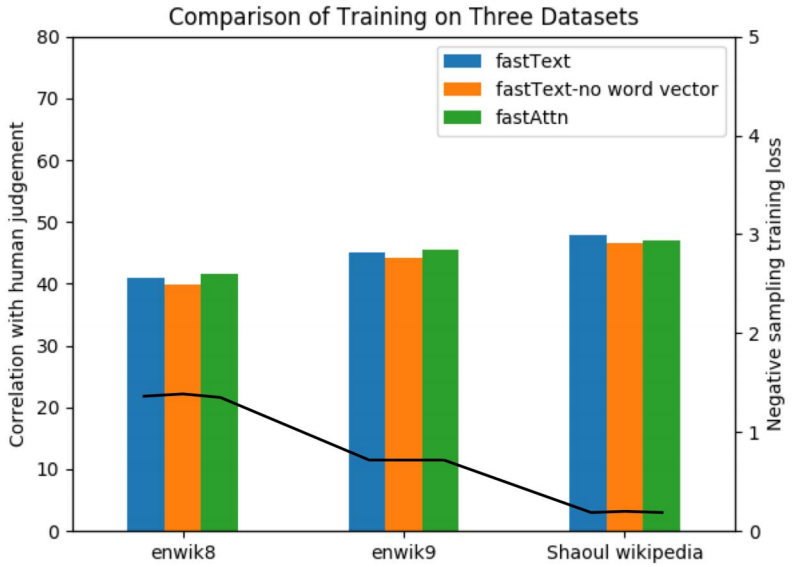

Figure 2: Comparison of training on enwik8, enwik9, and the Shaoul Wikipedia. The bars show the Spearmans rank correlation coefficient between human judgement and simlarity scores computed from different models. The black line show the negative sampling training loss of the models on the three datasets

5.2 Human Similarity Judgement

In this subsection, we will show that subword representation is sufficient to carry the semantic meaning of the words. To compare with Bojanowski *et al.* [2], we similarly evaluate our word representations on the task of word similarity / relatedness. We evaluate the model by computing the Spearmans rank correlation coefficient between human similarity judgement and the cosine similarity between the vector representations. Both the rare word dataset (RW) and the WS353 dataset is employed to cross-validate each other. We also compared our model with the previous works on word vectors incorporating subword information on word similarity tasks. The methods used are: the recursive neural network of Luong et al. (2013) [6], the morpheme cbow of Qiu et al. (2014) [7] and the morphological transformations of Soricut and Och (2015) [8]. The results are shown in Table 1, where our model outperforms that of Bojanowski *et al.* [2].

5.3 Text Classification

We also evaluate our model and compare it to fastText and other existing text classifiers on sentiment analysis tasks. We employ the same 8 datasets and evaluation protocol as in [3]. Other models that we compare with are the baseline and the character level convolutional model (char-CNN) of Zhang and LeCun (2015) [9], the character based convolution recurrent network (char-CRNN) of (Xiao and Cho, 2016) [10] and the very deep convolutional network (VDCNN) of Conneau et al. (2016)

Table 1: Correlation Between Human Judgment & Similarity Scores

	WS353	Rare Word
Luong et al. (2013)	64	34
Qiu et al. (2014)	65	33
Soricut and Och (2015)	71	42
fastText	71.8	45.0
fastText-no word vector	71.3	44.7
fastAttn	**73.6**	**45.4**

Table 2: Accuracy on Sentiment Datasets

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah.A.	Amz.F.	Amz.P.
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang et al., 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
fastText	92.3	96.8	98.6	95.7	63.9	72.5	60.3	94.6
fastAttn	92.4	96.3	97.9	95.9	62.2	71.8	61.4	94.5

[11]. Table 2 shows that the performance of our model is comparable to fastText, the char-CNN, and the char-CRNN models, but worse than the VDCNN. However, since no neural networks are involved, our model should train much faster than VDCNN. We used word bigrams to train both fastText and our fastAttn model. If we use trigrams, the performance of both models will increase. Table 2 indicates that removing word vectors from fastText still allows the model to achieve similar performance.

5.4 Qualitative analysis

Most significant character n-grams In a view of linguistics, it is interesting to show how a well trained model weighs the character n-grams, and if the most significant n-grams correspond to morphemes. We will quantify the significance of the n-grams using a modified method of [2]. We define the significance of an n-gram g as follows:

$$sig_g = cos(\boldsymbol{\mu}_w, \boldsymbol{\mu}_{w \setminus g}),$$

where $\boldsymbol{\mu}_{w \setminus g}$ is the the restricted representation obtained by omitting n-gram g in the word vector

$$\boldsymbol{\mu}_{w \setminus g} = \boldsymbol{\mu}_w - softmax(b_g)\boldsymbol{z}_g,$$

We picked some words that contains clear morphemes and show the most important 3 n-grams in these words in Table 3. We show that the model mostly "correctly" segments the words based on morphemes. However, we observe that our model tends to up-weight longer n-grams without clear semantic meaning like "teness" in "kindness". We doubt that in absence of the whole word vector, the model could "cheat" by learning long uncommon n-grams as the whole word vector. Training the model on larger dataset might mitigate the problem when all n-grams occur in multiple words with different meanings.

Nearest neighbors We hope to explore if our word representations carry enough semantic information to find synonyms, even though they are assembled from n-gram vectors. We show the nearest neighbors of some relatively uncommon words based the cosine similarity of word vectors in Table 4. The trained model was able to character semantic similarities between words even they don't share n-grams in common.

6 Conclusion

In this project, we build a model to learn word representations using only subword information. Our word vectors were computed using the attention of the n-gram representations. Our model

Table 3: The Most Significance N-grams in Words

	1st	2nd	3rd
anarchy	narchy	⟨anarc	⟨anar
monarchy	onarch	monarc	⟨monar
kindness	⟨kindn	⟨kind	ness⟩
politeness	polite	ness⟩	teness
lifetime	etime	⟨life	time⟩

Table 4: Nearest Neighbors Based on Cosine Similarity

Query Word	Nearest Neighbor
eateries	restaurant
tiling	tessellation
honda	toyota*
dna	ribonucleotide*
badminton	racquetball

* toyota is actually the second nearest neighbor, and the first was "hondas"
* riboneucleotide is RNA, so it's slightly different from DNA

outperforms Bojanowski *et al.* [2] in unsupervised tasks and gained higher correlation with human similarity judgment. The performance of our model on sentiment analysis on par with the fastText classifier in [3], showing that n-gram subwords are sufficient to represent the semantic meaning of words. We also showed that qualitatively, our model segments words mostly based on morphemes, and it could find synonyms for relatively less common words even though they don't share any n-grams in common.

For next steps, we are going to train the model on Wikipedia and repeat the experiments in Section 5.2 and Section 5.4. We hope that with a larger dataset, our model could put more significance on the morphemes and further down weight other long n-grams without specific meaning. We will also further test our model on multi-labeled text classification tasks other than sentiment analysis.

References

[1] Mikolov, Tomas, *et al. Distributed representations of words and phrases and their compositionality.* Advances in neural information processing systems. 2013.

[2] Bojanowski, Piotr, *et al. Enriching word vectors with subword information*. CoRR abs/1607.04606. (2016).

[3] Joulin, Armand, *et al. Bag of tricks for efficient text classification.* arXiv preprint arXiv:1607.01759 (2016).

[4] Shaoul, Cyrus. *The westbury lab wikipedia corpus.* Edmonton, AB: University of Alberta (2010).

[5] Finkelstein, Lev, *et al. Placing search in context: The concept revisited.* Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[6] Luong, Thang, Richard Socher, and Christopher Manning. *Better word representations with recursive neural networks for morphology.* Proceedings of the Seventeenth Conference on Computational Natural Language Learning. 2013.

[7] Qiu, Siyu, *et al. Co-learning of word representations and morpheme representations.* Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. 2014.

[8] Soricut, Radu, and Franz Och. *Unsupervised morphology induction using word embeddings.* Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015.

[9] Zhang, Xiang, and Yann LeCun. *Text understanding from scratch.* arXiv preprint arXiv:1502.01710 (2015).

[10] Xiao, Yijun, and Kyunghyun Cho. *Efficient character-level document classification by combining convolution and recurrent layers.* arXiv preprint arXiv:1602.00367 (2016).

[11] Conneau, Alexis, et al. *Very deep convolutional networks for natural language processing.* arXiv preprint arXiv:1606.01781 (2016).

[12] Deerwester, Scott, *et al. Indexing by latent semantic analysis.* Journal of the American society for information science 41.6 (1990): 391.

[13] Schtze, Hinrich. *Dimensions of meaning.* Proceedings of the 1992 ACM/IEEE conference on Supercomputing. IEEE Computer Society Press, 1992.

[14] Mikolov, Tomas, et al. *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781 (2013).

[15] Pennington, Jeffrey, Richard Socher, and Christopher Manning. *Glove: Global vectors for word representation.* Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[16] Botha, Jan, and Phil Blunsom. *Compositional morphology for word representations and language modelling.* International Conference on Machine Learning. 2014.