Subword-level Composition Functions for Learning Word Embeddings

Bofang Li[1], **Aleksandr Drozd**[2], **Tao Liu**[3], and **Xiaoyong Du**[4]

[1,2]School of Computing, Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan
[1,3,4]School of Information, Renmin University of China, Beijing, China
[1,3,4]Key laboratory of Data Engineering and Knowledge Engineering, MOE, Beijing, China
[1]libofang@ruc.edu.cn
[2]alex@smg.is.titech.ac.jp
[3]tliu@ruc.edu.cn
[4]duyong@ruc.edu.cn

Abstract

Subword-level information is crucial for capturing the meaning and morphology of words, especially for out-of-vocabulary entries. We propose CNN- and RNN-based subword-level composition functions for learning word embeddings, and systematically compare them with popular word-level and subword-level models (Skip-Gram and FastText). Additionally, we propose a hybrid training scheme in which a pure subword-level model is trained jointly with a conventional word-level embedding model based on lookup-tables. This increases the fitness of all types of subword-level word embeddings; the word-level embeddings can be discarded after training, leaving only compact subword-level representation with much smaller data volume. We evaluate these embeddings on a set of intrinsic and extrinsic tasks, showing that subword-level models have advantage on tasks related to morphology and datasets with high OOV rate, and can be combined with other types of embeddings.

1 Introduction

Word embeddings are used in many natural language processing tasks (Collobert et al., 2011; Socher et al., 2013; Kim, 2014). In word embedding models, words are mapped or "embedded" into low-dimensional real-valued vectors. Such mapping is based, implicitly or explicitly, on word co-occurrence statistics (Levy and Goldberg, 2014b).

Naturally, frequent words provide a better representation of their distributional properties; thus the quality of word embeddings is in direct relation to the frequency of words (Drozd et al., 2015). However, even in large corpora, most words occur very few times. For example, Baroni (2009) shows that the words occurring 3 times or less constitute almost 70% of the vocabulary. Consequently, most of the in-vocabulary words (for a given task/corpora) have to be discarded or embedded into low-quality vectors. Therefore, word-level models suffer from data sparsity.

Another issue with word-level models is that they do not make use of morphological information. Different forms of the same word are treated as completely unrelated entities. For example, as shown in Section 4.2, we find that the word "physicist" and "physicists" are not close to each other in a well-know word embedding model Skip-Gram (Mikolov et al., 2013).

These two issues are addressed by the emerging methodology of subword-level representations, as discussed in Section 2. The most notable example of such representations is FastText (Bojanowski et al., 2017). It represents each word as a bag-of-character n-grams. Representations for character n-grams, once they are learned, can be combined (via simple summation) to represent out-of-vocabulary (OOV) words.

This paper contributes to the discussion of composition functions for constructing subword-level embeddings and their evaluation. We propose and evaluate several models (including convolutional and recurrent neural networks) that can embed arbitrary character sequences into vectors. Our models do not rely on any external resource. We also propose a hybrid training scheme, which makes these neural networks directly integrated into Skip-Gram model. We train two sets of word embeddings simultaneously: one is from a lookup table as in traditional Skip-Gram, and another is from convolutional or recurrent neural network. The former is better at capturing semantic similarity. The latter is more focused on morphology and can learn embeddings for OOV words. We conduct experiments on five tasks, and compare our models with original Skip-Gram and the state-of-the-art performer FastText.

2 Related Work

2.1 Morphology-based Models

Morphology has long been considered as an important feature for word representations. For example, Lazaridou et al. (2013) investigate several algebraic composition functions (e.g. addition or multiplication) for morphologically complex words, which generate better representations compared to traditional distributional semantic models. Luong et al. (2013) train a recursive neural network for morphological composition, and show its effectiveness on (rare) word similarity task. Qiu et al. (2014) propose Morpheme CBOWs for word similarity and word analogy tasks, which improves on CBOW model (Mikolov et al., 2013) by learning morphology embeddings and word embeddings simultaneously. Alexandrescu and Kirchhoff (2006) take morphological tags as features (one-hot representation) for training a language model, which reduce the perplexity on rare word language modeling scenarios.

For both language modeling and machine translation tasks, LBL++ Botha and Blunsom (2014) show the effectiveness of summing morphology vectors in log-bilinear model (Mnih and Hinton, 2007) on 6 morphologically rich languages. Similarly, Morph-LBL (Cotterell and Schütze, 2015) improves on LBL model by predicting both context words and words' morphological tags in a semi-supervised fashion, which outperforms both Word2Vec and LBL on German morphological analysis.

However, all the above models rely on prior morphological knowledge, which is obtained by morphology analysis tools such as Morfessor (Creutz and Lagus, 2007), or an annotated morphology corpus such as CELEX (Baayen et al., 1995) and TIGER (Brants et al., 2004).

2.2 Subword-level Word Embeddings

Another line of work is focused on end-to-end word embedding learning based on subword-level information. FastText (Bojanowski et al., 2017) is probably the most influential and effective recent model. It represents each word as a bag-of-character n-grams. The models proposed in this paper are conceptually derived from FastText, i.e. we also operate on character n-grams level and predict context words from the target word, as in Skip-Gram approach.

Similarly to our proposed RNN, Cao and Rei (2016) train a bi-directional LSTM based on subword information. Instead of using character ngrams, their model feeds the word's prefixes and suffixes into each direction of LSTM respectively. This model is mainly designed to solve morphological boundary recovery task, it performs comparably with dedicated morphological analyzers. Pinter et al. (2017) also utilize BiLSTM to construct word embeddings. However, their model relies on pre-trained word embeddings by minimizing the squared Euclidean distance. In contrast, our proposed RNN requires only a plain text corpus.

2.3 Other Subword-level Models

There are also various task-specific NLP models that utilize character-level information for training deep neural networks in an end-to-end fashion. They often surpass the word-level baselines on language modeling (Mikolov et al., 2012; Sperr et al., 2013; Bojanowski et al., 2015; Kim et al., 2016), part-of-speech tagging (Ling et al., 2015; dos Santos and Zadrozny, 2014), text classification (Zhang et al., 2015), and machine translation (Sennrich et al., 2016; Luong and Manning, 2016), etc. However, these models do not produce representations that could be used in other tasks.

3 Models

3.1 Skip-Gram

Due to its popularity, simplicity, and state-of-the-art performance on a range of linguistic tasks, Skip-Gram (Mikolov et al., 2013) has been widely used as baseline in the word embedding literature (Levy and Goldberg, 2014a; Faruqui et al., 2015; Bojanowski et al., 2017; Zhao et al., 2017). In particular, we use Skip-Gram with negative sampling technique (Figure 1-a). For a vocabulary V of size $|V|$, Skip-Gram learns two set of vectors

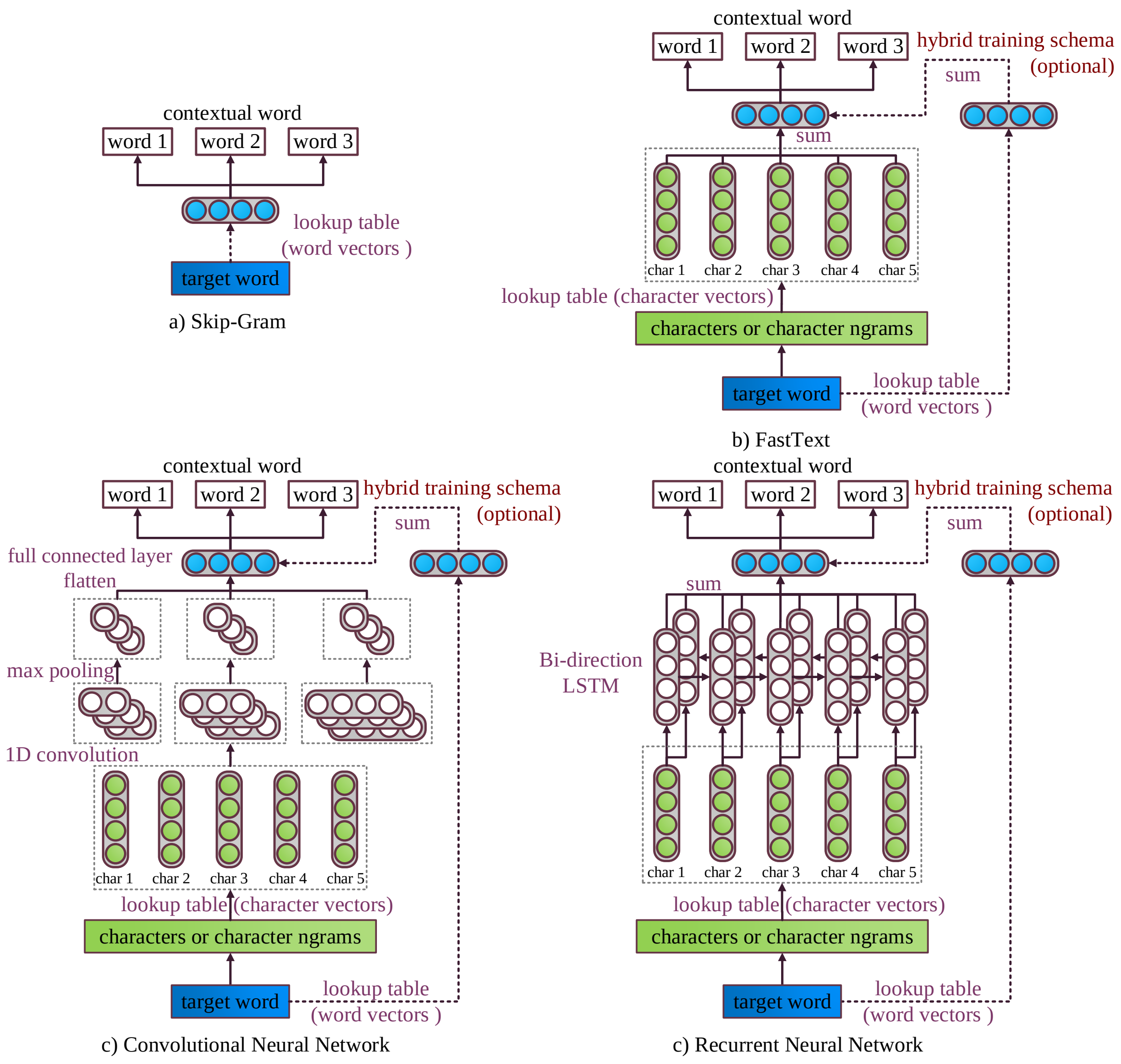

Figure 1: Illustration of original Skip-Gram and subword-level models.

$W, C \in \mathbb{R}^{|V|*N}$, namely word vectors and contextual word vectors. N is the dimension of vectors. Given a training corpus, Skip-Gram iterates through all words w and their contexts c, and maximizes the objective function $p(c|\vec{w})$, which is defined as:

$$\log \sigma\left(\vec{w} \cdot \vec{c}\right) + \sum_{k=1}^{K} \mathbb{E}_{c_i \sim \mathcal{N}_{w,c}} \left[\log \sigma\left(-\vec{w} \cdot \vec{c_i}\right)\right] \tag{1}$$

where σ is the sigmoid function. $\vec{w} \in W$ and $\vec{c} \in C$ are the vectors for word w and context c respectively. K is the negative sampling size. $\mathcal{N}_{w,c}$ is the negative example that sampled from the vocabulary V. The negative sampling probability is empirically defined as the unigram probability of a word raised to the power of $3/4$.

3.2 Utilizing Subword Information

In order to make use of subword information, we first generalize the objective function of Skip-Gram by replacing word vector $\vec{w}$ with a composition function $f(w)$. $f(w)$ takes word w as an input and outputs a vector of length N. Overall, the objective function of generalized Skip-Gram is defined as:

$$\begin{aligned} p(c|f(w)) &= \log \sigma\left(f(w) \cdot \vec{c}\right) \\ &+ \sum_{k=1}^{K} \mathbb{E}_{c_i \sim \mathcal{N}_{w,c}} \left[\log \sigma\left(-f(w) \cdot \vec{c_i}\right)\right] \end{aligned} \tag{2}$$

In the original Skip-Gram model, the function

$f(w)$ is simply a lookup table, which projects word w to its corresponding vector $\vec{w}$ in the table. Depending on the definition, $f(w)$ could also take the subword information of w into consideration. Naturally, composition functions, especially neural networks, can be considered.

3.2.1 A Hybrid Training scheme for Subword-level Models

Intuitively, compared to Skip-Gram, subword-level models should be better suited for capturing morphology instead of similarity. In order to take advantage of both representations, we incorporate Skip-Gram into subword-level models. Formally, in our hybrid training scheme we define the objective function of subword-level models as:

$$p(c|\vec{w} + f(w)) \qquad (3)$$

In this case, each word will have two embeddings: one from the lookup table (the same as Skip-Gram), and another from the composition function f. These two types of embeddings are learned simultaneously. We denote the embedding model from the word-level lookup table as Model$_{word}$, and the one from composition function as Model$_{subword}$. As a baseline we additionally train embeddings using only subword-level composition function; these models will be referred to as Model$_{vanilla}$.

3.2.2 FastText (Summation)

Probably the most simple and intuitive way of utilizing subword information is to sum all vectors of characters and character ngrams belonging to a word (Figure 1-b), which is pioneered by Bojanowski et al. (2017). Formally, the composition function is defined as $f(w) = \sum_{g \in \mathcal{G}_w} \vec{g}$, where g is the character n-gram, and $\vec{g}$ is its corresponding n-gram vector with length N. $\mathcal{G}_w$ is the set of character n-grams for word w. For example, when $n = 3$, $\mathcal{G}_w$ for word “bigger” is defined as `<bi, big, igg, gge, ger, er>`. The angle brackets are padding at the start and end of the word.

Note that the original FastText (FastText$_{vanilla}$) does not have the hybrid training scheme. As later shown in our experiments, FastText$_{word}$ from the hybrid training scheme works better than FastText$_{vanilla}$ in some semantic relatedness datasets. Moreover, hybrid training scheme is essential for other types of composition functions.

For fair comparison, in the following models, we use the same padding and the same length of character n-grams’ vector.

3.2.3 Convolutional Neural Network

Despite its simplicity and efficiency, there is no clear evidence that simple summation as in FastText is the best choice for composing subword information.

This paper investigate two neural network. We first consider the Convolutional Neural Network (CNN) as composition function $f(w)$. CNN (LeCun, 1998) is able to capture local features automatically, and has been applied to a wide range of NLP tasks (Kim et al., 2016; Zhang et al., 2015; Luong and Manning, 2016).

The CNN architecture introduced in this paper is inspired by the model used for language modeling in Kim et al. (2016). As illustrated in Figure 1-c, similarly to FastText, the vectors of characters are first extracted from a lookup table. Those vectors form a matrix with size $N * L$, where L is the number of characters. The 1D convolution filters are used to extract local features. We apply 1D convolutions of size ranging from 1 to 7 in parallel, perform max-pooling and concatenate the output. Each of the convolutions uses 200 filters. The output of this model is a fully-connected layer with the number of units corresponding to the desired size of embeddings. The resulting vector is used for predicting contextual words using negative sampling, the same as the negative sampling in Skip-Gram.

3.2.4 Recurrent Neural Network

Another neural network that is worth considering is Recurrent Neural Network (RNN). It takes a sequence of arbitrary length as an input, and outputs a vector that represents this sequence. Among all the different variations, the Long Short-Term Memory based recurrent neural network (LSTM) (Hochreiter and Schmidhuber, 1997) and its bi-direction version (Schuster and Paliwal, 1997) are easier to train, and better capture long-distance information. As illustrated in Figure 1-d, for each direction, an LSTM runs over all the vectors of characters in the word w. The hidden layers at each position are then summed together, and the resulting vector is fed into a fully connected layer to form the final vector w. We empirically set the hidden layer size of LSTM to $N * 2$.

Skip-Gram	FastText$_{subword}$	RNN$_{subword}$	CNN$_{subword}$
geneticists cosmologists	musicologists classicists	psychologists **physiologist**	protists **physicians**
logicians historians	biophysicist **physicians**	trombonists pharmacists	cryonicists **physician**
humanists geographers	**physicalism physicist**	aerodynamicists **physiocrats**	artists **physicality**
zoologists philosophers	mathematicians publicists	**physicist** microeconomists	**physiocrats** publicists
astrologers astronomers	ethicists eugenicists	**physicality physicians**	**physics physicist**

Table 1: Illustration of the top-10 most similar words of target word "physicists".

4 Experiments

4.1 Implementation Details

We implemented all models described in section 3 using Chainer deep learning framework (Tokui et al., 2015). Since the proposed CNN and RNN architectures significantly increase computational requirements for training, we choose a relatively small TEXT8[1] corpus for this evaluation. This corpus contains the first 10^9 bytes of the English Wikipedia dump from Mar. 3, 2006. The word embedding size N is set to 300. The batch size is fixed to 1000. The negative sampling size is set to 5, and the window size is set to 2. Following Chainer's original word2vec implementation, we use Adam (Kingma and Ba, 2014) as the optimization function. Words which appear fewer than five times are directly discarded, which results in vocabulary size of 71290. For character ngrams, we follow the FastText's best configuration and use 5-grams for FastText. We also discard character ngrams which appear fewer than five times, which results in character ngrams vocabulary size of 143207. Word embedding models are trained for 5 epochs on Nvidia Tesla K80 or P100 GPU.

We also download the state-of-the-art FastText embeddings (denoted as FastText$_{external}$) [2], which are trained on a much larger full Wikipedia corpus. Note that since CNN and RNN require approximate 45 and 65 days of training on K80, we didn't train on this corpus.

For the fair comparison, we ensure that all embeddings used in evaluations use exactly same vocabulary. Unlike most of the benchmarks, where only embeddings of encountered words affect resulting accuracy, analogical reasoning benchmark is sensitive to the entire vocabulary in terms of size and embeddings of individual words. For example it is hard to make a mistake when looking for "Paris" as the pair for "France" if the whole vocabulary contains only these two words. Furthermore, accuracy depends on how close the target word is to the source words (Rogers et al., 2017), which could also be affected by a larger vocabulary.

This issue is especially pronounced for embeddings with dynamic vocabulary, such as subword-level models evaluated in this study. In our pilot experiments, models with large vocabulary like FastText$_{external}$ result in poor performance on word analogy tasks since large vocabulary increases the number of options.

Skip-Gram	FastText$_{vanilla}$	FastText$_{external}$
85.6M	171.9M	8493.6M

CNN$_{vanilla}$	RNN$_{vanilla}$
7.5M	19.1M

Table 2: Memory footprint of different models (For Skip-Gram and FastText, it will increase on larger vocabulary. For CNN- and RNN-based models, it is constant.

Note that the sizes of these models are different, as shown in Table 2. Due to the large number of character ngrams, FastText requires the most data among these models, while CNN needs only a few megabytes of data.

4.2 Qualitative Analysis

Before looking into the performance of models on specific tasks, we first conduct qualitative analysis. We choose several target words and analyze their nearest neighbors in Skip-Gram, FastText$_{subword}$, CNN$_{subword}$, and RNN$_{subword}$. We find that subword-level models, especially CNN- and RNN-based models, tend to cluster words with the same morpheme together. Taking target word "physicists" as an example (Table 1), the word "physicist" is within the top-10 nearest neighbors in subword-level models, but not in

[1] `http://mattmahoney.net/dc/textdata.html`

[2] `https://github.com/facebookresearch/fastText`

Model	Word Similarity						Word Analogy					
	Rare Words	WS sem.	WS rel.	Sim 999	MEN	Mech Turk	BATS inf.	BATS der.	BATS enc.	BATS lex.	Google sem.	Google syn.
Skip-Gram	.059	.658	.586	.285	.555	.577	.652	.096	.280	.098	**.427**	.534
FastText$_{word}$	.065	.708	**.619**	.285	.610	.596	.622	.102	.274	.078	.424	.530
CNN$_{word}$	.058	.670	.584	.245	.554	.584	.628	.108	.250	.074	.416	.489
LSTM$_{word}$	.063	.697	.623	.284	.600	.599	.666	.134	.264	.080	.427	.597
Concat$_{word}$							.712	.184	.316	**.130**	.492	.649
FastText$_{subword}$	.085	.707	.566	.273	.620	.582	.790	.580	**.290**	.082	.374	**.812**
CNN$_{subword}$	.051	.581	.417	.190	.478	.509	.758	.682	.076	.028	.019	.789
RNN$_{subword}$	.063	.664	.532	.282	.566	.579	.798	.672	.114	.028	.061	.786
Concat$_{subword}$							.846	.696	.326	.078	.290	.914
FastText$_{word+subword}$							.792	.578	**.350**	.104	**.439**	.856
CNN$_{word+subword}$							.832	.638	.242	.084	.160	.859
RNN$_{word+subword}$							.832	.606	.244	.086	.240	.875
FastText$_{subword+OOV}$	.344	.715	.575	.277	.619	.599	.842	.824	.254	.096	.340	.776
CNN$_{subword+OOV}$	.234	.564	.409	.224	.490	.497	.880	**.948**	.094	.044	.023	.786
RNN$_{subword+OOV}$	.299	.670	.540	**.286**	.566	.587	**.908**	.906	.134	.040	.061	.796
Concat$_{subword+OOV}$							**.962**	**.960**	.328	.116	.298	**.917**
FastText$_{vanilla+OOV}$	.348	.717	.579	.283	.630	.624	.840	.834	.252	**.116**	.347	.799
CNN$_{vanilla+OOV}$	.212	.535	.400	.185	.474	.556	.874	.918	.104	.042	.015	.773
RNN$_{vanilla+OOV}$	.273	.638	.542	.250	.576	.568	.856	.866	.112	.034	.050	.748
Concat$_{vanilla+OOV}$							.958	.958	.314	.126	.275	.895
FastText$_{external}$	.096	.674	.604	.332	.600	.574	.746	.446	.528	.194	.779	.872
FastText$_{external+OOV}$	.431	.682	.607	.341	.600	.587	.834	.672	.562	.214	.798	.879

Table 3: Results on word similarity and word analogy datasets. For hybrid training scheme, we denote the embeddings that come from word vector lookup table as "Model$_{word}$", and the embeddings which come from the composition function as "Model$_{subword}$". We denote the vanilla (non-hybrid) models as "Model$_{vanilla}$". The "FastText$_{external}$" is the public available FastText embeddings, which are trained on the full Wikipedia corpus. We also test the version where OOV words are expanded, and denote as "Model$_{+OOV}$". Model combinations are denoted as gray rows, and best results among them are marked **bold**. Rare words dataset in blue column have 43.3% OOV rate, while other word similarity datasets have maximum 4.6% OOV rate. Morphology related categories are denoted as almond columns. The results of model combination for word similarity task are simply the average of results from each single models, which are not listed in this table.

Skip-Gram. Moreover, subword models tend to cluster words with the same morphology form (affix) together, especially for RNN and CNN.

4.3 Quantitative Analysis

4.3.1 Word Similarity

Word similarity task aims at producing semantic similarity scores of word pairs, which are compared with the human scores using Spearman's correlation. The cosine distance is used for generating similarity scores between two word vectors. In order to test the effectiveness of capturing word similarity for rare word, we choose the Rare Words dataset (Luong et al., 2013). For systematical comparison, we also test our models on the WordSim353 (WS) (Finkelstein et al., 2001) dataset, divided into similarity (sem.) and relatedness (rel.) categories (Zesch et al., 2008; Agirre et al., 2009), Sim 999 dataset (Hill et al., 2016), MEN dataset (Bruni et al., 2012), and Mech Turk dataset (Radinsky et al., 2011).

Table 3 shows that on word similarity tasks FastText models perform the best in all datasets except Sim 999. CNN$_{subword}$ and RNN$_{subword}$ are more focused on word morphology, and thus do not perform well on word similarity task.

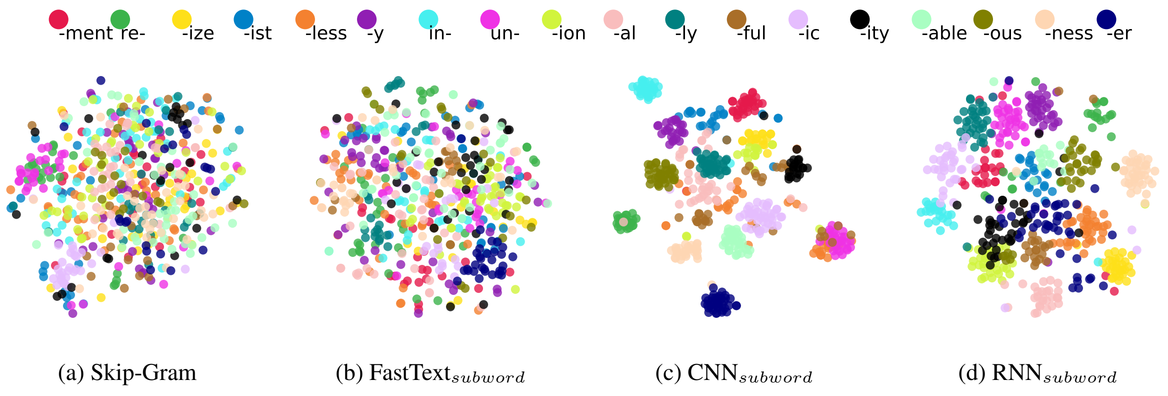

Figure 3: Visualization of learned word embeddings, each dot represents a word, different colors represent different affixes.

However, compared to Skip-Gram, CNN$_{word}$ and RNN$_{word}$ (the versions with word vector lookup table) achieve comparable or even better results. Their word vector lookup tables in these models are affected by the composition function, which results in better performance.

Note that the Rare Words dataset has 43.3% of words which are OOV. In this dataset, the vocabulary expanded models (FastText$_{subword+OOV}$, CNN$_{subword+OOV}$, and RNN$_{subword+OOV}$) perform a lot better than others. This highlights the necessity of expanding vocabulary and the effectiveness of subword-level models.

4.3.2 Word Analogy

The word analogy task aims at answering questions generalize as “a is to a’ as b is to __ ?”, such as “London is to Britain as Tokyo is to Japan”. We follow the evaluation protocol by Drozd et al. (2016) who proposed the LRCos method of solving word analogies, which significantly improves on the traditional vector offset method. We use Google analogy dataset (Mikolov et al., 2013) along with a much bigger and balanced BATS analogy dataset (Gladkova et al., 2016).

On word analogy datasets (Table 3), the inflectional and derivational morphology categories demonstrate the effectiveness of subword-level word models. It is especially obvious on derivation morphology category, where Skip-Gram only achieves 9.6% accuracy and subword-level models achieve minimal 57.8% accuracy (excluding the lookup table versions). Furthermore, when the vocabulary is expanded, the minimal accuracy of subword-level models reaches 82.4%.

Morphology-related analogy questions also benefit a lot from the model combination. Notably, Concat$_{subowrd+OOV}$ achieves an accuracy of 96.2% and 96.0% on inflection and derivation morphology, which is by far the highest accuracies on this two categories. We also observe that CNN is less sensitive to semantic word analogy, while performing the best on derivational word analogy.

4.3.3 Affix Prediction

In this section we test the ability of subword-level embeddings to predict what affix is present in a morphologically complex word. We use the dataset gathered by (Lazaridou et al., 2013), which contains 6549 stems and derived word pairs, such as “name”-“rename” and “sparse”-“sparsity”. There are 18 affixes, such as “re-” and “-ity”, and the task is to predict which one is present in a given word. We use the embeddings of derived words as input, and feed them to a logistic regression classifier for predicting their affixes. The accuracy, recall, and F_1-score are used for measurement. We also follow Lazaridou et al. (2013) in using the default training/test data split.

Figure 3 shows a t-SNE projection of the words with different affixes in the dataset. It is clear that both CNN and RNN are able to distinguish different derivation types, with the advantage of the former. This also confirms the good performance of CNN on derivational analogy task. Note that FastText does not fare much better than Skip-Gram, although it is a subword-level model. This partially explains its low accuracy compared to other subword-level models on morphological analogy categories.

The prediction results in Table 4 reflect the cluster visualization in Figure 3. Moreover, as in the word analogy task, the concatenation (especially with the expanded vocabulary version) performs

Model	AP			SL		
	P	R	F_1	POS	Chunk	NER
Skip-Gram	.324	.270	.251	.878	.881	.915
FastText$_{word}$	.346	.267	.250	.880	.883	.917
CNN$_{word}$	.344	.289	.270	.877	.882	.912
LSTM$_{word}$	.354	.298	.280	.878	.882	.915
Concat$_{word}$	.359	.301	.298	.890	.891	.921
FastText$_{subword}$	.527	.430	.433	.830	.822	.910
CNN$_{subword}$	.878	.622	.694	.845	.850	.870
RNN$_{subword}$	.864	.614	.684	.866	.872	.897
Concat$_{subword}$	.900	.628	.705	.892	.890	.919
FastText$_{word+subword}$	.520	.426	.428	.886	.887	.920
CNN$_{word+subword}$	.886	.621	.696	.890	.888	.913
RNN$_{word+subword}$	.839	.607	.670	.890	.888	.919
FastText$_{subword+OOV}$	.564	.481	.493	.834	.804	**.929**
CNN$_{subword+OOV}$	.912	**.687**	.765	.909	.895	.909
RNN$_{subword+OOV}$	.890	.674	.751	.925	**.912**	**$\underline{.929}$**
Concat$_{subword+OOV}$	**$\underline{.928}$**	.694	**$\underline{.777}$**	**$\underline{.945}$**	.925	**$\underline{.948}$**
FastText$_{vanilla+OOV}$	.574	.489	.502	.833	.803	$\underline{.929}$
CNN$_{vanilla+OOV}$	$\underline{.920}$	.689	.770	.907	.894	.906
RNN$_{vanilla+OOV}$	.897	.683	**.757**	.924	$\underline{.912}$	.926
Concat$_{vanilla+OOV}$	.914	**$\underline{.701}$**	.768	**$\underline{.945}$**	**$\underline{.926}$**	**$\underline{.948}$**
FastText$_{external}$	.521	.411	.414	.888	.882	.929
FastText$_{external}$	.636	.707	.659	.941	.919	.940

Table 4: Results on affix prediction (AP) and sequence labeling (SL) tasks. Sequence labeling tasks have 16.5%, 27.1%, 28.5% OOV rate respectively.

the best among all the other models.

4.3.4 Sequence Labeling

Sequence labeling task consists in assigning labels to elements of texts. We evaluate word embedding models on Part-of-Speech Tagging (POS), Chunking[3] and Named Entity Recognition (NER) tasks [4]. Following the evaluation protocol used in Kiros et al. (2015); Li et al. (2017), we restrict the predicting model to Logistic Regression Classifier [5]. The classifier's input for predicting the label of word w_i is simply the concatenation of word vectors $\vec{w_{i-2}}$, $\vec{w_{i-1}}$, $\vec{w_i}$, $\vec{w_{i+1}}$, $\vec{w_{i+2}}$. This ensures that the quality of the embedding models is directly evaluated, and their strengths and weaknesses are easily observed.

Subword-level models on sequence labeling tasks clearly demonstrate the effectiveness of expanding OOV words. As shown in Table 4, expanding vocabulary boosts the performance by a large margin. For example, on NER task, FastText$_{subword+OOV}$, CNN$_{subword+OOV}$, and RNN$_{subword+OOV}$ achieve 1.9%, 2.1%, 3.2% absolute gains over the versions of the same models without expanded vocabulary.

[3]CoNLL 2000 shared task `http://www.cnts.ua.ac.be/conll2000/chunking`

[4]CoNLL 2003 shared task `http://www.cnts.ua.ac.be/conll2003/ner`

[5]`http://scikit-learn.org/`

4.3.5 Text Classification

For text classification task, we choose the movie review sentiment (MR) (Pang and Lee, 2005), customer product reviews (CR) (Nakagawa et al., 2010), subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004), and IMDB movie review (IMDB) (Maas et al., 2011) datasets. The classification is performed by Logistic Regression Classifier. The input of this classifier is the sum of word embeddings that belonging to the text.

As shown in Table 5, the input word embeddings do not considerably affect the final accuracy. This is especially obvious when comparing *subword* and *subword + OOV* models. It's hard to draw any insightful conclusion from this experiment. This is in line with the observations of Li et al. (2017), who showed that Skip-Gram, CBOW, and GloVe trained with different context types perform similarly on text classification task.

5 Discussion

The evaluation showed that despite being trained on a relatively small corpus, CNN- and RNN-based (model-based) approaches outperform conventional (trained on word-level) embeddings as well as FastText embeddings, which are claimed to better capture morphological information (Bojanowski et al., 2017). In some cases, the performance of model-based embeddings is even higher than that of FastText embeddings that were trained on much larger corpus.

Moreover, such performance is achieved with very compact representations: it is possible to generate an embedding for any given word on demand with only network weights, and these weights require an order of mere kilobytes of data (Table 2). Naturally, these compact representations do not have enough capacity to capture semantic information well. However, besides merely indicating a possibility for improvement/limitations of more "heavy-weight" models (like original Skip-Gram or FastText), model-based embeddings can be used together with other approaches to improve their sensitivity to morphological information.

One such approach is to combine embeddings from different models after training, as demonstrated in our experiments ("concat" lines in Table 3). Simple concatenation of lookup-table-based and model-based embeddings maintain high accuracy on morphology-related benchmarks, while elevating performance on semantic tasks to comparable level.

Additionally, subword-level models can be trained *jointly* with models based on lookup-tables, which improves their performance on different tasks. After training, either part can be used independently or jointly (e.g. by concatenation) in down-stream tasks.

6 Conclusion

We have implemented and evaluated several types of composition functions for subword-level elements (characters and character n-grams) in the context of training word embeddings in Skip-Gram-like model.

We have shown that morphological information can be captured efficiently by extremely compact models. Embeddings generated dynamically from just a few megabytes of parameters significantly outperform conventional (word2vec and FastText) models on morphology related tasks. Additionally, this indicates the vast limitation of the ability of conventional models to capture morphological information.

To model both morphological and semantic information, we propose two methods for combining strength of compact subword-level- and lookup-table based models: merging trained embeddings and training jointly. The resulting embeddings achieved high accuracy on a range of benchmarks and are particularly promising for datasets with high OOV rate.

The source code of those models, along with the pretrained word embeddings, has been integrated into an open-source project, and will be publicly available.

Model	Text Classification			
	MR	CR	SUBJ	IMDB
Skip-Gram	.688	.765	.881	.797
FastText$_{word}$	.690	.756	.878	.796
CNN$_{word}$	.690	.764	.876	.796
LSTM$_{word}$	.687	.752	.881	.794
FastText$_{subword}$	.691	.758	.867	.798
CNN$_{subword}$	.670	.759	.857	.784
RNN$_{subword}$	.689	.758	.872	.796
FastText$_{subword+OOV}$	.693	.759	.869	.797
CNN$_{subword+OOV}$	.675	.763	.858	.785
RNN$_{subword+OOV}$	.692	.760	.872	.795
FastText$_{word+subword}$	.693	.759	.869	.797
CNN$_{word+subword}$	.675	.763	.858	.785
RNN$_{word+subword}$	.692	.760	.872	.795

Table 5: Results on text classification datasets.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant No. JP17K12739, JST CREST Grant No. JPMJCR1687 and National Natural Science Foundation of China Grant No.61472428.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27. Association for Computational Linguistics.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *HLT-NAACL*.

R Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. The celex lexical database (release 2). *Distributed by the Linguistic Data Consortium, University of Pennsylvania*.

Marco Baroni. 2009. Distributions in text.

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.

Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. 2015. Alternative structures for character-level rnns. *CoRR*, abs/1511.06303.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, pages 136–145. Association for Computational Linguistics.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *ACL 2016*, page 18.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *HLT-NAACL*, pages 1287–1292.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2015. Discovering aspectual classes of russian verbs in untagged large corpora. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 61–68.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *COLING*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't. In *NAACL-HLT*, pages 8–15.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Yoon Kim, Yacine Jernite, David A Sontag, and Alexander M. Rush. 2016. Character-aware neural language models.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *ACL (1)*, pages 1517–1526.

Yann LeCun. 1998. Gradient-based learning applied to document recognition.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

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 *EMNLP*, pages 2411–2421.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, abs/1604.00788.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*, pages 641–648.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *NAACL*, pages 786–794. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pages 271–278. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *EMNLP*.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *COLING*, pages 141–150.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *WWW*, pages 337–346. ACM.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (Too Many) Problems of Analogical Reasoning with Word Vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

Cicero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642. Citeseer.

Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 30–39.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, volume 5.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, pages 861–866.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 244–253.