Enriching Word Vectors with Subword Information

Piotr Bojanowski* and **Edouard Grave*** and **Armand Joulin** and **Tomas Mikolov**
Facebook AI Research
{bojanowski,egrave,ajoulin,tmikolov}@fb.com

Abstract

Continuous word representations, trained on large unlabeled corpora are useful for many natural language processing tasks. Popular models that learn such representations ignore the morphology of words, by assigning a distinct vector to each word. This is a limitation, especially for languages with large vocabularies and many rare words. In this paper, we propose a new approach based on the skipgram model, where each word is represented as a bag of character n-grams. A vector representation is associated to each character n-gram; words being represented as the sum of these representations. Our method is fast, allowing to train models on large corpora quickly and allows us to compute word representations for words that did not appear in the training data. We evaluate our word representations on nine different languages, both on word similarity and analogy tasks. By comparing to recently proposed morphological word representations, we show that our vectors achieve state-of-the-art performance on these tasks.

1 Introduction

Learning continuous representations of words has a long history in natural language processing (Rumelhart et al., 1988). These representations are typically derived from large unlabeled corpora using co-occurrence statistics (Deerwester et al., 1990; Schütze, 1992; Lund and Burgess, 1996). A large body of work, known as distributional semantics, has studied the properties of these methods (Turney et al., 2010; Baroni and Lenci, 2010). In the neural network community, Collobert and Weston (2008) proposed to learn word embeddings using a feed-forward neural network, by predicting a word based on the two words on the left and two words on the right. More recently, Mikolov et al. (2013b) proposed simple log-bilinear models to learn continuous representations of words on very large corpora efficiently.

Most of these techniques represent each word of the vocabulary by a distinct vector, without parameter sharing. In particular, they ignore the internal structure of words, which is an important limitation for morphologically rich languages, such as Turkish or Finnish. For example, in French or Spanish, most verbs have more than forty different inflected forms, while the Finnish language has fifteen cases for nouns. These languages contain many word forms that occur rarely (or not at all) in the training corpus, making it difficult to learn good word representations. Because many word formations follow rules, it is possible to improve vector representations for morphologically rich languages by using character level information.

In this paper, we propose to learn representations for character n-grams, and to represent words as the sum of the n-gram vectors. Our main contribution is to introduce an extension of the continuous skipgram model (Mikolov et al., 2013b), which takes into account subword information. We evaluate this model on nine languages exhibiting different morphologies, showing the benefit of our approach.

*The two first authors contributed equally.

Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017. Action Editor: Hinrich Schütze.
Submission batch: 9/2016; Revision batch: 12/2016; Published 6/2017.

2 Related work

Morphological word representations. In recent years, many methods have been proposed to incorporate morphological information into word representations. To model rare words better, Alexandrescu and Kirchhoff (2006) introduced factored neural language models, where words are represented as sets of features. These features might include morphological information, and this technique was succesfully applied to morphologically rich languages, such as Turkish (Sak et al., 2010). Recently, several works have proposed different composition functions to derive representations of words from morphemes (Lazaridou et al., 2013; Luong et al., 2013; Botha and Blunsom, 2014; Qiu et al., 2014). These different approaches rely on a morphological decomposition of words, while ours does not. Similarly, Chen et al. (2015) introduced a method to jointly learn embeddings for Chinese words and characters. Cui et al. (2015) proposed to constrain morphologically similar words to have similar representations. Soricut and Och (2015) described a method to learn vector representations of morphological transformations, allowing to obtain representations for unseen words by applying these rules. Word representations trained on morphologically annotated data were introduced by Cotterell and Schütze (2015). Closest to our approach, Schütze (1993) learned representations of character four-grams through singular value decomposition, and derived representations for words by summing the four-grams representations. Very recently, Wieting et al. (2016) also proposed to represent words using character n-gram count vectors. However, the objective function used to learn these representations is based on paraphrase pairs, while our model can be trained on any text corpus.

Character level features for NLP. Another area of research closely related to our work are character-level models for natural language processing. These models discard the segmentation into words and aim at learning language representations directly from characters. A first class of such models are recurrent neural networks, applied to language modeling (Mikolov et al., 2012; Sutskever et al., 2011; Graves, 2013; Bojanowski et al., 2015), text normalization (Chrupała, 2014), part-of-speech tagging (Ling et al., 2015) and parsing (Ballesteros et al., 2015). Another family of models are convolutional neural networks trained on characters, which were applied to part-of-speech tagging (dos Santos and Zadrozny, 2014), sentiment analysis (dos Santos and Gatti, 2014), text classification (Zhang et al., 2015) and language modeling (Kim et al., 2016). Sperr et al. (2013) introduced a language model based on restricted Boltzmann machines, in which words are encoded as a set of character n-grams. Finally, recent works in machine translation have proposed using subword units to obtain representations of rare words (Sennrich et al., 2016; Luong and Manning, 2016).

3 Model

In this section, we propose our model to learn word representations while taking into account morphology. We model morphology by considering subword units, and representing words by a sum of its character n-grams. We will begin by presenting the general framework that we use to train word vectors, then present our subword model and eventually describe how we handle the dictionary of character n-grams.

3.1 General model

We start by briefly reviewing the continuous skipgram model introduced by Mikolov et al. (2013b), from which our model is derived. Given a word vocabulary of size W, where a word is identified by its index $w \in \{1, ..., W\}$, the goal is to learn a vectorial representation for each word w. Inspired by the distributional hypothesis (Harris, 1954), word representations are trained to *predict well* words that appear in its context. More formally, given a large training corpus represented as a sequence of words $w_1, ..., w_T$, the objective of the skipgram model is to maximize the following log-likelihood:

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

where the context $\mathcal{C}_t$ is the set of indices of words surrounding word w_t. The probability of observing a context word w_c given w_t will be parameterized using the aforementioned word vectors. For now, let us consider that we are given a scoring function s which maps pairs of (word, context) to scores in $\mathbb{R}$.

One possible choice to define the probability of a context word is the softmax:

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

However, such a model is not adapted to our case as it implies that, given a word w_t, we only predict one context word w_c.

The problem of predicting context words can instead be framed as a set of independent binary classification tasks. Then the goal is to independently predict the presence (or absence) of context words. For the word at position t we consider all context words as positive examples and sample negatives at random from the dictionary. For a chosen context position c, using the binary logistic loss, we obtain the following negative log-likelihood:

$$\log\left(1 + e^{-s(w_t,\, w_c)}\right) + \sum_{n \in \mathcal{N}_{t,c}} \log\left(1 + e^{s(w_t,\, n)}\right),$$

where $\mathcal{N}_{t,c}$ is a set of negative examples sampled from the vocabulary. By denoting the logistic loss function $\ell : x \mapsto \log(1 + e^{-x})$, we can re-write the objective as:

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

A natural parameterization for the scoring function s between a word w_t and a context word w_c is to use word vectors. Let us define for each word w in the vocabulary two vectors u_w and v_w in $\mathbb{R}^d$. These two vectors are sometimes referred to as *input* and *output* vectors in the literature. In particular, we have vectors $\mathbf{u}_{w_t}$ and $\mathbf{v}_{w_c}$, corresponding, respectively, to words w_t and w_c. Then the score can be computed as the scalar product between word and context vectors as $s(w_t, w_c) = \mathbf{u}_{w_t}^\top \mathbf{v}_{w_c}$. The model described in this section is the skipgram model with negative sampling, introduced by Mikolov et al. (2013b).

3.2 Subword model

By using a distinct vector representation for each word, the skipgram model ignores the internal structure of words. In this section, we propose a different scoring function s, in order to take into account this information.

Each word w is represented as a bag of character n-gram. We add special boundary symbols < and > at the beginning and end of words, allowing to distinguish prefixes and suffixes from other character sequences. We also include the word w itself in the set of its n-grams, to learn a representation for each word (in addition to character n-grams). Taking the word *where* and $n = 3$ as an example, it will be represented by the character n-grams:

<wh, whe, her, ere, re>

and the special sequence

<where>.

Note that the sequence <her>, corresponding to the word *her* is different from the tri-gram her from the word *where*. In practice, we extract all the n-grams for n greater or equal to 3 and smaller or equal to 6. This is a very simple approach, and different sets of n-grams could be considered, for example taking all prefixes and suffixes.

Suppose that you are given a dictionary of n-grams of size G. Given a word w, let us denote by $\mathcal{G}_w \subset \{1, \ldots, G\}$ the set of n-grams appearing in w. We associate a vector representation $\mathbf{z}_g$ to each n-gram g. We represent a word by the sum of the vector representations of its n-grams. We thus obtain the scoring function:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

This simple model allows sharing the representations across words, thus allowing to learn reliable representation for rare words.

In order to bound the memory requirements of our model, we use a hashing function that maps n-grams to integers in 1 to K. We hash character sequences using the Fowler-Noll-Vo hashing function (specifically the FNV-1a variant).[1] We set $K = 2.10^6$ below. Ultimately, a word is represented by its index in the word dictionary and the set of hashed n-grams it contains.

4 Experimental setup

4.1 Baseline

In most experiments (except in Sec. 5.3), we compare our model to the C implementation

[1] http://www.isthe.com/chongo/tech/comp/fnv

of the skipgram and cbow models from the word2vec[2] package.

4.2 Optimization

We solve our optimization problem by performing stochastic gradient descent on the negative log likelihood presented before. As in the baseline skipgram model, we use a linear decay of the step size. Given a training set containing T words and a number of passes over the data equal to P, the step size at time t is equal to $\gamma_0(1 - \frac{t}{TP})$, where γ_0 is a fixed parameter. We carry out the optimization in parallel, by resorting to Hogwild (Recht et al., 2011). All threads share parameters and update vectors in an asynchronous manner.

4.3 Implementation details

For both our model and the baseline experiments, we use the following parameters: the word vectors have dimension 300. For each positive example, we sample 5 negatives at random, with probability proportional to the square root of the uni-gram frequency. We use a context window of size c, and uniformly sample the size c between 1 and 5. In order to subsample the most frequent words, we use a rejection threshold of 10^{-4} (for more details, see (Mikolov et al., 2013b)). When building the word dictionary, we keep the words that appear at least 5 times in the training set. The step size γ_0 is set to 0.025 for the skipgram baseline and to 0.05 for both our model and the cbow baseline. These are the default values in the word2vec package and work well for our model too.

Using this setting on English data, our model with character n-grams is approximately $1.5\times$ slower to train than the skipgram baseline. Indeed, we process 105k words/second/thread versus 145k words/second/thread for the baseline. Our model is implemented in C++, and is publicly available.[3]

4.4 Datasets

Except for the comparison to previous work (Sec. 5.3), we train our models on Wikipedia data.[4] We downloaded Wikipedia dumps in nine languages: Arabic, Czech, German, English, Spanish, French, Italian, Romanian and Russian. We normalize the raw Wikipedia data using Matt Mahoney's pre-processing perl script.[5] All the datasets are shuffled, and we train our models by doing five passes over them.

5 Results

We evaluate our model in five experiments: an evaluation of word similarity and word analogies, a comparison to state-of-the-art methods, an analysis of the effect of the size of training data and of the size of character n-grams that we consider. We will describe these experiments in detail in the following sections.

5.1 Human similarity judgement

We first evaluate the quality of our representations on the task of word similarity / relatedness. We do so by computing Spearman's rank correlation coefficient (Spearman, 1904) between human judgement and the cosine similarity between the vector representations. For German, we compare the different models on three datasets: GUR65, GUR350 and ZG222 (Gurevych, 2005; Zesch and Gurevych, 2006). For English, we use the WS353 dataset introduced by Finkelstein et al. (2001) and the rare word dataset (RW), introduced by Luong et al. (2013). We evaluate the French word vectors on the translated dataset RG65 (Joubarne and Inkpen, 2011). Spanish, Arabic and Romanian word vectors are evaluated using the datasets described in (Hassan and Mihalcea, 2009). Russian word vectors are evaluated using the HJ dataset introduced by Panchenko et al. (2016).

We report results for our method and baselines for all datasets in Table 1. Some words from these datasets do not appear in our training data, and thus, we cannot obtain word representation for these words using the cbow and skipgram baselines. In order to provide comparable results, we propose by default to use null vectors for these words. Since our model exploits subword information, we can also compute valid representations for out-of-vocabulary words. We do so by taking the sum of its n-gram vectors. When OOV words are represented using

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

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

[4] https://dumps.wikimedia.org

[5] http://mattmahoney.net/dc/textdata

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Table 1: Correlation between human judgement and similarity scores on word similarity datasets. We train both our model and the `word2vec` baseline on normalized Wikipedia dumps. Evaluation datasets contain words that are not part of the training set, so we represent them using null vectors (`sisg-`). With our model, we also compute vectors for unseen words by summing the n-gram vectors (`sisg`).

null vectors we refer to our method as `sisg-` and `sisg` otherwise (Subword Information Skip Gram).

First, by looking at Table 1, we notice that the proposed model (`sisg`), which uses subword information, outperforms the baselines on all datasets except the English WS353 dataset. Moreover, computing vectors for out-of-vocabulary words (`sisg`) is always at least as good as not doing so (`sisg-`). This proves the advantage of using subword information in the form of character n-grams.

Second, we observe that the effect of using character n-grams is more important for Arabic, German and Russian than for English, French or Spanish. German and Russian exhibit grammatical declensions with four cases for German and six for Russian. Also, many German words are compound words; for instance the nominal phrase "table tennis" is written in a single word as "Tischtennis". By exploiting the character-level similarities between "Tischtennis" and "Tennis", our model does not represent the two words as completely different words.

Finally, we observe that on the English Rare Words dataset (RW), our approach outperforms the baselines while it does not on the English WS353 dataset. This is due to the fact that words in the English WS353 dataset are common words for which good vectors can be obtained without exploiting subword information. When evaluating on less frequent words, we see that using similarities at the character level between words can help learning good word vectors.

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

Table 2: Accuracy of our model and baselines on word analogy tasks for Czech, German, English and Italian. We report results for semantic and syntactic analogies separately.

5.2 Word analogy tasks

We now evaluate our approach on word analogy questions, of the form A is to B as C is to D, where D must be predicted by the models. We use the datasets introduced by Mikolov et al. (2013a) for English, by Svoboda and Brychcin (2016) for Czech, by Köper et al. (2015) for German and by Berardi et al. (2015) for Italian. Some questions contain words that do not appear in our training corpus, and we thus excluded these questions from the evaluation.

We report accuracy for the different models in Table 2. We observe that morphological information significantly improves the syntactic tasks; our approach outperforms the baselines. In contrast, it does not help for semantic questions, and even degrades the performance for German and Italian. Note that this is tightly related to the choice of the length of character n-grams that we consider. We show in Sec. 5.5 that when the size of the n-grams is chosen optimally, the semantic analogies degrade

	DE		EN		ES	FR
	GUR350	ZG222	WS353	RW	WS353	RG65
Luong et al. (2013)	-	-	64	34	-	-
Qiu et al. (2014)	-	-	65	33	-	-
Soricut and Och (2015)	64	22	71	42	47	67
sisg	73	43	73	48	54	69
Botha and Blunsom (2014)	56	25	39	30	28	45
sisg	66	34	54	41	49	52

Table 3: Spearman's rank correlation coefficient between human judgement and model scores for different methods using morphology to learn word representations. We keep all the word pairs of the evaluation set and obtain representations for out-of-vocabulary words with our model by summing the vectors of character n-grams. Our model was trained on the same datasets as the methods we are comparing to (hence the two lines of results for our approach).

less. Another interesting observation is that, as expected, the improvement over the baselines is more important for morphologically rich languages, such as Czech and German.

5.3 Comparison with morphological representations

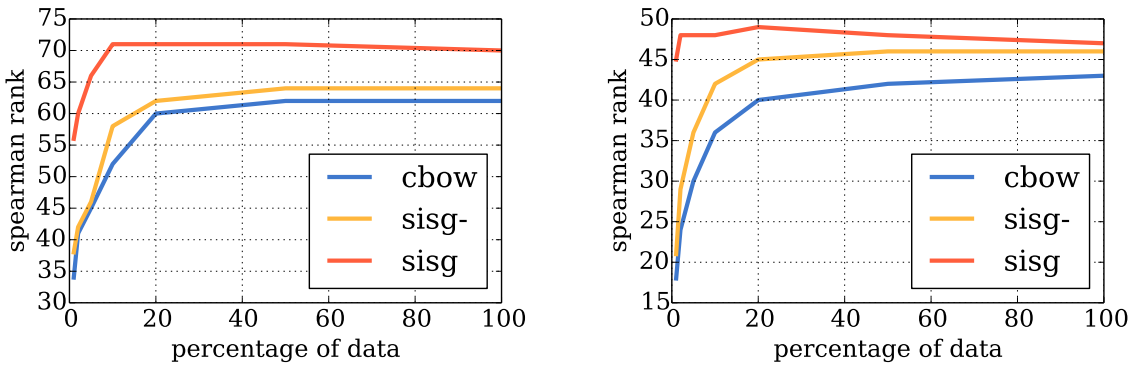
We also compare our approach to previous work on word vectors incorporating subword information on word similarity tasks. The methods used are: the recursive neural network of Luong et al. (2013), the morpheme cbow of Qiu et al. (2014) and the morphological transformations of Soricut and Och (2015). In order to make the results comparable, we trained our model on the same datasets as the methods we are comparing to: the English Wikipedia data released by Shaoul and Westbury (2010), and the news crawl data from the 2013 WMT shared task for German, Spanish and French. We also compare our approach to the log-bilinear language model introduced by Botha and Blunsom (2014), which was trained on the Europarl and news commentary corpora. Again, we trained our model on the same data to make the results comparable. Using our model, we obtain representations of out-of-vocabulary words by summing the representations of character n-grams. We report results in Table 3. We observe that our simple approach performs well relative to techniques based on subword information obtained from morphological segmentors. We also observe that our approach outperforms the Soricut and Och (2015) method, which is based on prefix and suffix analysis. The large improvement for German is due to the fact that their approach does not model noun compounding, contrary to ours.

5.4 Effect of the size of the training data

Since we exploit character-level similarities between words, we are able to better model infrequent words. Therefore, we should also be more robust to the size of the training data that we use. In order to assess that, we propose to evaluate the performance of our word vectors on the similarity task as a function of the training data size. To this end, we train our model and the cbow baseline on portions of Wikipedia of increasing size. We use the Wikipedia corpus described above and isolate the first 1, 2, 5, 10, 20, and 50 percent of the data. Since we don't reshuffle the dataset, they are all subsets of each other. We report results in Fig. 1.

As in the experiment presented in Sec. 5.1, not all words from the evaluation set are present in the Wikipedia data. Again, by default, we use a null vector for these words (sisg-) or compute a vector by summing the n-gram representations (sisg). The out-of-vocabulary rate is growing as the dataset shrinks, and therefore the performance of sisg- and cbow necessarily degrades. However, the proposed model (sisg) assigns non-trivial vectors to previously unseen words.

First, we notice that for all datasets, and all sizes, the proposed approach (sisg) performs better than



(a) DE-GUR350 (b) EN-RW

Figure 1: Influence of size of the training data on performance. We compute word vectors following the proposed model using datasets of increasing size. In this experiment, we train models on a fraction of the full Wikipedia dump.

the baseline. However, the performance of the baseline `cbow` model gets better as more and more data is available. Our model, on the other hand, seems to quickly saturate and adding more data does not always lead to improved results.

Second, and most importantly, we notice that the proposed approach provides very good word vectors even when using very small training datasets. For instance, on the German GUR350 dataset, our model (`sisg`) trained on 5% of the data achieves better performance (66) than the `cbow` baseline trained on the full dataset (62). On the other hand, on the English RW dataset, using 1% of the Wikipedia corpus we achieve a correlation coefficient of 45 which is better than the performance of `cbow` trained on the full dataset (43). This has a very important practical implication: well performing word vectors can be computed on datasets of a restricted size and still work well on previously unseen words. In general, when using vectorial word representations in specific applications, it is recommended to retrain the model on textual data relevant for the application. However, this kind of relevant task-specific data is often very scarce and learning from a reduced amount of training data is a great advantage.

5.5 Effect of the size of n-grams

The proposed model relies on the use of character n-grams to represent words as vectors. As mentioned in Sec. 3.2, we decided to use n-grams ranging from 3 to 6 characters. This choice was arbitrary, motivated by the fact that n-grams of these lengths will cover a wide range of information. They would include short suffixes (corresponding to conjugations and declensions for instance) as well as longer roots. In this experiment, we empirically check for the influence of the range of n-grams that we use on performance. We report our results in Table 4 for English and German on word similarity and analogy datasets.

We observe that for both English and German, our arbitrary choice of 3-6 was a reasonable decision, as it provides satisfactory performance across languages. The optimal choice of length ranges depends on the considered task and language and should be tuned appropriately. However, due to the scarcity of test data, we did not implement any proper validation procedure to automatically select the best parameters. Nonetheless, taking a large range such as $3 - 6$ provides a reasonable amount of subword information.

This experiment also shows that it is important to include long n-grams, as columns corresponding to $n \leq 5$ and $n \leq 6$ work best. This is especially true for German, as many nouns are compounds made up from several units that can only be captured by longer character sequences. On analogy tasks, we observe that using larger n-grams helps for semantic analogies. However, results are always improved by taking $n \geq 3$ rather than $n \geq 2$, which shows that character 2-grams are not informative for that task. As described in Sec. 3.2, before computing

	2	3	4	5	6
2	57	64	67	69	69
3		65	68	70	70
4			70	70	**71**
5				69	**71**
6					70

(a) DE-GUR350

	2	3	4	5	6
2	59	55	56	59	60
3		60	58	60	62
4			62	62	63
5				64	64
6					**65**

(b) DE Semantic

	2	3	4	5	6
2	45	50	53	54	55
3		51	55	55	**56**
4			54	**56**	**56**
5				**56**	**56**
6					54

(c) DE Syntactic

	2	3	4	5	6
2	41	42	46	47	**48**
3		44	46	**48**	**48**
4			47	**48**	**48**
5				**48**	**48**
6					**48**

(d) EN-RW

	2	3	4	5	6
2	78	76	75	76	76
3		78	77	78	77
4			79	79	79
5				**80**	79
6					**80**

(e) EN Semantic

	2	3	4	5	6
2	70	71	73	74	73
3		72	74	**75**	74
4			74	**75**	**75**
5				74	74
6					72

(f) EN Syntactic

Table 4: Study of the effect of sizes of n-grams considered on performance. We compute word vectors by using character n-grams with n in $\{i, \ldots, j\}$ and report performance for various values of i and j. We evaluate this effect on German and English, and represent out-of-vocabulary words using subword information.

character n-grams, we prepend and append special positional characters to represent the beginning and end of word. Therefore, 2-grams will not be enough to properly capture suffixes that correspond to conjugations or declensions, since they are composed of a single proper character and a positional one.

5.6 Language modeling

In this section, we describe an evaluation of the word vectors obtained with our method on a language modeling task. We evaluate our language model on five languages (CS, DE, ES, FR, RU) using the datasets introduced by Botha and Blunsom (2014). Each dataset contains roughly one million training tokens, and we use the same preprocessing and data splits as Botha and Blunsom (2014).

Our model is a recurrent neural network with 650 LSTM units, regularized with dropout (with probability of 0.5) and weight decay (regularization parameter of 10^{-5}). We learn the parameters using the Adagrad algorithm with a learning rate of 0.1, clipping the gradients which have a norm larger than 1.0. We initialize the weight of the network in the range $[-0.05, 0.05]$, and use a batch size of 20. Two baselines are considered: we compare our approach to the log-bilinear language model of Botha and Blunsom (2014) and the character aware language model of Kim et al. (2016). We trained word vectors with character n-grams on the training set of the language modeling task and use them to initialize the lookup table of our language model. We report the test perplexity of our model without using pre-trained word vectors (`LSTM`), with word vectors pre-trained without subword information (`sg`) and with our vectors (`sisg`). The results are presented in Table 5.

We observe that initializing the lookup table of the language model with pre-trained word representations improves the test perplexity over the baseline LSTM. The most important observation is that using word representations trained with subword information outperforms the plain skipgram model. We observe that this improvement is most significant for morphologically rich Slavic languages such as Czech (8% reduction of perplexity over `sg`) and Russian (13% reduction). The improvement is less significant for Roman languages such as Spanish (3% reduction) or French (2% reduction). This shows the importance of subword information on the language modeling task and exhibits the usefulness

	CS	DE	ES	FR	RU
Vocab. size	46k	37k	27k	25k	63k
CLBL	465	296	200	225	304
CANLM	371	239	165	184	261
LSTM	366	222	157	173	262
sg	339	216	150	162	237
sisg	**312**	**206**	**145**	**159**	**206**

Table 5: Test perplexity on the language modeling task, for 5 different languages. We compare to two state of the art approaches: CLBL refers to the work of Botha and Blunsom (2014) and CANLM refers to the work of Kim et al. (2016).

of the vectors that we propose for morphologically rich languages.

6 Qualitative analysis

6.1 Nearest neighbors.

We report sample qualitative results in Table 7. For selected words, we show nearest neighbors according to cosine similarity for vectors trained using the proposed approach and for the skipgram baseline. As expected, the nearest neighbors for complex, technical and infrequent words using our approach are better than the ones obtained using the baseline model.

6.2 Character n-grams and morphemes

We want to qualitatively evaluate whether or not the most important n-grams in a word correspond to morphemes. To this end, we take a word vector that we construct as the sum of n-grams. As described in Sec. 3.2, each word w is represented as the sum of its n-grams: $u_w = \sum_{g \in \mathcal{G}_w} z_g$. For each n-gram g, we propose to compute the restricted representation $u_{w \backslash g}$ obtained by omitting g:

$$u_{w \backslash g} = \sum_{g' \in \mathcal{G} - \{g\}} z_{g'}.$$

We then rank n-grams by increasing value of cosine between u_w and $u_{w \backslash g}$. We show ranked n-grams for selected words in three languages in Table 6.

For German, which has a lot of compound nouns, we observe that the most important n-grams correspond to valid morphemes. Good examples include *Autofahrer* (car driver) whose most important n-grams are *Auto* (car) and *Fahrer* (driver). We also observe the separation of compound nouns into morphemes in English, with words such as *lifetime* or *starfish*. However, for English, we also observe that n-grams can correspond to affixes in words such as *kindness* or *unlucky*. Interestingly, for French we observe the inflections of verbs with endings such as *ais>*, *ent>* or *ions>*.

	word	n-grams		
DE	autofahrer	fahr	fahrer	auto
	freundeskreis	kreis	kreis>	<freun
	grundwort	wort	wort>	grund
	sprachschule	schul	hschul	sprach
	tageslicht	licht	gesl	tages
EN	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
FR	finirais	ais>	nir	fini
	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

Table 6: Illustration of most important character n-grams for selected words in three languages. For each word, we show the n-grams that, when removed, result in the most different representation.

6.3 Word similarity for OOV words

As described in Sec. 3.2, our model is capable of building word vectors for words that do not appear in the training set. For such words, we simply average the vector representation of its n-grams. In order to assess the quality of these representations, we analyze which of the n-grams match best for OOV words by selecting a few word pairs from the English RW similarity dataset. We select pairs such that one of the two words is not in the training vocabulary and is hence only represented by its n-grams. For each pair of words, we display the cosine similarity between each pair of n-grams that appear

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
`sisg`	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
`sg`	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Table 7: Nearest neighbors of rare words using our representations and `skipgram`. These hand picked examples are for illustration.

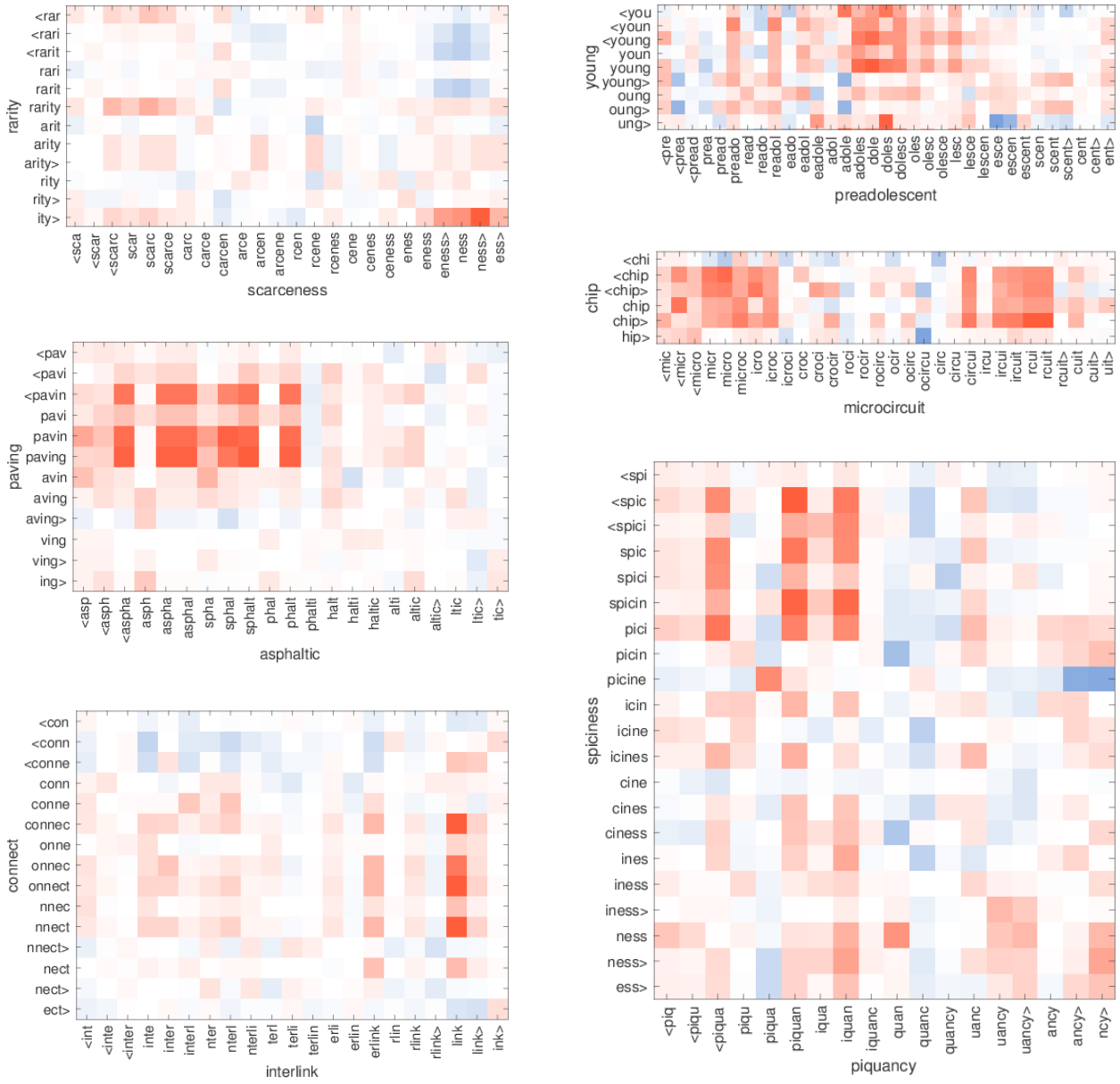


Figure 2: Illustration of the similarity between character n-grams in out-of-vocabulary words. For each pair, only one word is OOV, and is shown on the x axis. Red indicates positive cosine, while blue negative.

in the words. In order to simulate a setup with a larger number of OOV words, we use models trained on 1% of the Wikipedia data as in Sec. 5.4. The results are presented in Fig. 2.

We observe interesting patterns, showing that subwords match correctly. Indeed, for the word *chip*, we clearly see that there are two groups of n-grams in *microcircuit* that match well. These roughly correspond to *micro* and *circuit*, and n-grams in between don't match well. Another interesting example is the pair *rarity* and *scarceness*. Indeed, *scarce* roughly matches *rarity* while the suffix *-ness* matches *-ity* very well. Finally, the word *preadolescent* matches *young* well thanks to the *-adolesc-* subword. This shows that we build robust word representations where prefixes and suffixes can be ignored if the grammatical form is not found in the dictionary.

7 Conclusion

In this paper, we investigate a simple method to learn word representations by taking into account subword information. Our approach, which incorporates character n-grams into the skipgram model, is related to an idea that was introduced by Schütze (1993). Because of its simplicity, our model trains fast and does not require any preprocessing or supervision. We show that our model outperforms baselines that do not take into account subword information, as well as methods relying on morphological analysis. We will open source the implementation of our model, in order to facilitate comparison of future work on learning subword representations.

Acknowledgements

We thank Marco Baroni, Hinrich Schütze and the anonymous reviewers for their insightful comments.

References

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

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. EMNLP*.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Giacomo Berardi, Andrea Esuli, and Diego Marcheggiani. 2015. Word embeddings go to Italy: a comparison of models and training datasets. *Italian Information Retrieval Workshop*.

Piotr Bojanowski, Armand Joulin, and Tomáš Mikolov. 2015. Alternative structures for character-level RNNs. In *Proc. ICLR*.

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

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Proc. IJCAI*.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proc. ACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proc. NAACL*.

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 Transactions on Information Systems*, 34(1):4:1–4:25.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Cicero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proc. COLING*.

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

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Iryna Gurevych. 2005. Using the structure of a conceptual network in computing semantic relatedness. In *Proc. IJCNLP*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *Proc. EMNLP*.

Colette Joubarne and Diana Inkpen. 2011. Comparison of semantic similarity for different languages using the google n-gram corpus and second-order co-occurrence measures. In *Proc. Canadian Conference on Artificial Intelligence*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proc. AAAI*.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and "semantic" structure of continuous word spaces. *Proc. IWCS 2015*.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositionally derived representations of morphologically complex words in distributional semantics. In *Proc. ACL*.

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

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proc. ACL*.

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

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Černocký. 2012. Subword language modeling with neural networks. Technical report, Faculty of Information Technology, Brno University of Technology.

Tomáš Mikolov, Kai Chen, Greg D. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Adv. NIPS*.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. 2016. Human and machine judgements for russian semantic relatedness. In *Proc. AIST*.

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

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Adv. NIPS*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press.

Haşim Sak, Murat Saraclar, and Tunga Gungör. 2010. Morphology-based and sub-word language modeling for turkish speech recognition. In *Proc. ICASSP*.

Hinrich Schütze. 1992. Dimensions of meaning. In *Proc. IEEE Conference on Supercomputing*.

Hinrich Schütze. 1993. Word space. In *Adv. NIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. ACL*.

Cyrus Shaoul and Chris Westbury. 2010. The Westbury lab Wikipedia corpus.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proc. NAACL*.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

Henning Sperr, Jan Niehues, and Alexander Waibel. 2013. Letter n-gram-based input encoding for continuous space language models. In *Proc. of the Workshop on Continuous Vector Space Models and their Compositionality at ACL*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proc. ICML*.

Lukáš Svoboda and Tomáš Brychcin. 2016. New word analogy corpus for exploring embeddings of Czech words. In *Proc. CICLING*.

Peter D. Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. CHARAGRAM: Embedding words and sentences via character n-grams. In *Proc. EMNLP*.

Torsten Zesch and Iryna Gurevych. 2006. Automatically creating datasets for measures of semantic relatedness. In *Proc. Workshop on Linguistic Distances*.

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