Better Word Embeddings by Disentangling Contextual n-Gram Information

Prakhar Gupta*
EPFL
Switzerland
prakhar.gupta@epfl.ch

Matteo Pagliardini*
Iprova SA, Switzerland
mpagliardini@iprova.com

Martin Jaggi
EPFL
Switzerland
martin.jaggi@epfl.ch

Abstract

Pre-trained word vectors are ubiquitous in Natural Language Processing applications. In this paper, we show how training word embeddings jointly with bigram and even trigram embeddings, results in improved unigram embeddings. We claim that training word embeddings along with higher n-gram embeddings helps in the removal of the contextual information from the unigrams, resulting in better stand-alone word embeddings. We empirically show the validity of our hypothesis by outperforming other competing word representation models by a significant margin on a wide variety of tasks. We make our models publicly available.

1 Introduction

Distributed word representations are essential building blocks of modern NLP systems. Used as features in downstream applications, they often enhance generalization of models trained on a limited amount of data. They do so by capturing relevant distributional information about words from large volumes of unlabeled text.

Efficient methods to learn word vectors have been introduced in the past, most of them based on the distributional hypothesis of Harris (1954); Firth (1957): *"a word is characterized by the company it keeps"*. While a standard approach relies on global corpus statistics (Pennington et al., 2014) formulated as a matrix factorization using mean square reconstruction loss, other widely used methods are the bilinear word2vec architectures introduced by Mikolov et al. (2013a): While skip-gram aims to predict nearby words from a given word, CBOW predicts a target word from its set of context words.

Recently, significant improvements in the quality of the word embeddings were obtained by augmenting word-context pairs with sub-word information in the form of character n-grams (Bojanowski et al., 2017), especially for morphologically rich languages. Nevertheless, to the best of our knowledge, no method has been introduced leveraging collocations of words with higher order word n-grams such as bigrams or trigrams as well as character n-grams together.

In this paper, we show how using higher order word n-grams along with unigrams during training can significantly improve the quality of obtained word embeddings. The addition furthermore helps to disentangle contextual information present in the training data from the unigrams and results in overall better distributed word representations.

To validate our claim, we train two modifications of CBOW augmented with word-n-gram information during training. One is a recent sentence embedding method, Sent2Vec (Pagliardini et al., 2018), which we repurpose to obtain word vectors. The second method we propose is a modification of CBOW enriched with character n-gram information (Bojanowski et al., 2017) that we again augment with word n-gram information. In both cases, we compare the resulting vectors with the most widely used word embedding methods on word similarity and analogy tasks and show significant quality improvements. The code used to train the models presented in this paper as well as the models themselves are made available to the public[1].

2 Model Description

Before introducing our model, we recapitulate fundamental existing word embeddings methods.

CBOW and skip-gram models. Continuous bag-of-words (CBOW) and skip-gram models are

* indicates equal contribution

[1]publicly available on http://github.com/epfml/sent2vec

standard log-bilinear models for obtaining word embeddings based on word-context pair information (Mikolov et al., 2013a). Context here refers to a symmetric window centered on the target word w_t, containing the surrounding tokens at a distance less than some window size ws: $C_t = \{w_k \,|\, k \in [t - \mathsf{ws}, t + \mathsf{ws}]\}$. The CBOW model tries to predict the target word given its context, maximizing the likelihood $\prod_{t=1}^{T} p(w_t|C_t)$, whereas skip-gram learns by predicting the context for a given target word maximizing $\prod_{t=1}^{T} p(C_t|w_t)$. To model those probabilities, a softmax activation is used on top of the inner product between a target vector $\mathbf{u}_{w_t}$ and its context vector $\frac{1}{|C_t|}\sum_{w \in C_t} \mathbf{v}_w$.

To overcome the computational bottleneck of the softmax for large vocabulary, negative sampling or noise contrastive estimation are well-established (Mikolov et al., 2013b), with the idea of employing simpler pairwise binary classifier loss functions to differentiate between the valid context C_t and fake contexts N_{C_t} sampled at random. While generating target-context pairs, both CBOW and skip-gram also use input word subsampling, discarding higher-frequency words with higher probability during training, in order to prevent the model from overfitting the most frequent tokens. Standard CBOW also uses a dynamic context window size: for each subsampled target word w, the size of its associated context window is sampled uniformly between 1 and ws (Mikolov et al., 2013b).

Adding character n-grams. Bojanowski et al. (2017) have augmented CBOW and skip-gram by adding character n-grams to the context representations. Word vectors are expressed as the sum of its unigram and average of its character n-gram embeddings W_w:

$$\mathbf{v} := \mathbf{v}_w + \frac{1}{|W_w|} \sum_{c \in W_w} \mathbf{v}_c$$

Character n-grams are hashed to an index in the embedding matrix . The training remains the same as for CBOW and skip-gram. This approach greatly improves the performances of CBOW and skip-gram on morpho-syntactic tasks. For the rest of the paper, we will refer to the CBOW and skip-gram methods enriched with subword-information as *CBOW-char* and *skip-gram-char* respectively.

GloVe. Instead of training online on local window contexts, GloVe vectors (Pennington et al., 2014) are trained using global co-occurrence statistics by factorizing the word-context co-occurrence matrix.

Ngram2vec. In order to leverage the performance of word vectors, training of word vectors using the skip-gram objective function with negative sampling is augmented with n-gram co-occurrence information (Zhao et al., 2017).

2.1 Improving unigram embeddings by adding higher order word-n-grams to contexts

CBOW-char with word n-grams. We propose to augment CBOW-char to additionally use word n-gram context vectors (in addition to char n-grams and the context word itself). More precisely, during training, the context vector for a given word w_t is given by the average of all word-n-grams N_t, all char-n-grams, and all unigrams in the span of the current context window C_t:

$$\mathbf{v} := \frac{\sum_{w \in C_t} \mathbf{v}_w + \sum_{n \in N_t} \mathbf{v}_n + \sum_{w \in C_t} \sum_{c \in W_w} \mathbf{v}_c}{|C_t| + |N_t| + \sum_{w \in C_t} |W_w|} \tag{1}$$

For a given sentence, we apply input subsampling and a sliding context window as for standard CBOW. In addition, we keep the mapping from the subsampled sentence to the original sentence for the purpose of extracting word n-grams from the original sequence of words, within the span of the context window. Word n-grams are added to the context using the hashing trick in the same way char-n-grams are handled. We use two different hashing index ranges to ensure there is no collision between char n-gram and word n-gram representations.

Sent2Vec for word embeddings. Initially implemented for sentence embeddings, Sent2Vec (Pagliardini et al., 2018) can be seen as a derivative of word2vec's CBOW. The key differences between CBOW and Sent2Vec are the removal of the input subsampling, considering the entire sentence as context, as well as the addition of word-n-grams.

Here, word and n-grams embeddings from an entire sentence are averaged to form the corresponding sentence (context) embedding.

For both proposed CBOW-char and Sent2Vec models, we employ dropout on word n-grams during training. For both models, word embeddings are obtained by simply discarding the higher order n-gram embeddings after training.

Model	WS 353	WS 353 Relatedness	WS 353 Similarity
CBOW-char	.709 ± .006	.626 ± .009	.783 ± .004
CBOW-char + bi.	.719 ± .007	.652 ± .010	.778 ± .007
CBOW-char + bi. + tri.	.727 ± .008	.664 ± .008	.783 ± .004
Sent2Vec uni.	.705 ± .004	.593 ± .005	.793 ± .006
Sent2Vec uni. + bi.	.755 ± .005	.683 ± .008	.817 ± .007
Sent2Vec uni. + bi. + tri.	**.780 ± .003**	**.721 ± .006**	**.828 ± .003**

Model	SimLex-999	MEN	Rare Words	Mechanical Turk
CBOW-char	.424 ± .004	.769 ± .002	.497 ± .002	.675 ± .007
CBOW-char + bi.	.436 ± .004	.786 ± .002	.506 ± .001	.671 ± .007
CBOW-char + bi. + tri.	.441 ± .003	.788 ± .002	**.509 ± .003**	**.678 ± .010**
Sent2Vec uni.	.450 ± .003	.765 ± .002	.444 ± .001	.625 ± .005
Sent2Vec uni. + bi.	.440 ± .002	.791 ± .002	.430 ± .002	.661 ± .005
Sent2Vec uni. + bi. + tri.	**.464 ± .003**	**.798 ± .001**	.432 ± .003	.658 ± .006

Model	Google (Syntactic Analogies)	Google (Semantic Analogies)	MSR
CBOW-char	.920 ± .001	.799 ± .004	.842 ± .002
CBOW-char + bi.	.928 ± .003	.798 ± .006	.856 ± .004
CBOW-char + bi. + tri.	**.929 ± .001**	.794 ± .005	**.857 ± .002**
Sent2Vec uni.	.826 ± .003	.847 ± .003	.734 ± .003
Sent2Vec uni. + bi.	.843 ± .004	.844 ± .002	.754 ± .004
Sent2Vec uni. + bi. + tri.	.837 ± .003	**.853 ± .003**	.745 ± .001

Table 1: Impact of using word n-grams: Models are compared using Spearman correlation measures for word similarity tasks and accuracy for word analogy tasks. Top performances on each dataset are shown in bold. An underline shows the best model(s) restricted to each architecture type. The abbreviations uni., bi., and tri. stand for unigrams, bigrams, and trigrams respectively.

3 Experimental Setup

3.1 Training

We train all competing models on a wikipedia dump of 69 million sentences containing 1.7 billion words, following (Pagliardini et al., 2018).

Sentences are tokenized using the Stanford NLP library (Manning et al., 2014). All algorithms are implemented using a modified version of the fast-text (Bojanowski et al., 2017; Joulin et al., 2017) and sent2vec (Pagliardini et al., 2018) libraries respectively. Detailed training hyperparameters for all models included in the comparison are provided in Table 3 in the supplementary material. During training, we save models checkpoints at 20 equidistant intervals and found out that the best performance for CBOW models occurs around $60-80\%$ of the total training. As a result, we also indicate the checkpoint at which we stop training the CBOW models. We use 300-dimension vectors for all our word embedding models. For the Ngram2vec model, learning source and target embeddings for all the n-grams upto bigrams was the best performing model and is included in the comparison.

For each method, we extensively tuned hyperparameters starting from the recommended values. For each model, we select the parameters which give the best averaged results on our word-similarity and analogy tasks. After selecting the best hyperparameters, we train 5 models for each method, using a different random seed. The reported results are given as mean and standard deviation for those five models.

3.2 Evaluation

In order to evaluate our model, we use six datasets covering pair-wise word-similarity tasks and two datasets covering word-analogy tasks.

Word-similarity tasks. Word-similarity tasks consist of word pairs along with their human annotated similarity scores. To evaluate the performance of our models on pair-wise word-similarity tasks, we use *WordSim353* (353 word-pairs) (Finkelstein et al., 2002) divided into two datasets, *WordSim Similarity* (203 word-pairs) and

Model	WS 353	WS 353 Relatedness	WS 353 Similarity
CBOW-char + bi. + tri.	.727 ± .008	.664 ± .008	.783 ± .004
Sent2Vec uni. + bi. + tri.	**.780 ± .003**	**.721 ± .006**	**.828 ± .003**
CBOW-char	.709 ± .006	.626 ± .009	.783 ± .004
CBOW	.722 ± .008	.634 ± .008	.796 ± .005
Skip-gram-char	.724 ± .007	.655 ± .008	.789 ± .004
Skip-gram	.736 ± .004	.672 ± .007	.796 ± .005
GloVe	.559 ± .002	.484 ± .005	.665 ± .008
Ngram2vec bi. - bi.	.745 ± .003	.687 ± .003	.797 ± .004

Model	SimLex-999	MEN	Rare Words	Mechanical Turk
CBOW-char + bi. + tri.	.441 ± .003	.788 ± .002	**.509 ± .003**	.678 ± .010
Sent2Vec uni. + bi. + tri.	**.464 ± .003**	**.798 ± .001**	.432 ± .003	.658 ± .006
CBOW-char	.424 ± .004	.769 ± .002	.497 ± .002	.675 ± .007
CBOW	.432 ± .004	.757 ± .002	.454 ± .002	.674 ± .006
Skip-gram-char	.395 ± .003	.762 ± .001	.487 ± .002	**.684 ± .003**
Skip-gram	.405 ± .001	.770 ± .001	.468 ± .002	**.684 ± .005**
GloVe	.375 ± .002	.690 ± .001	.327 ± .002	.622 ± .004
Ngram2vec bi. - bi.	.424 ± .000	.756 ± .001	.462 ± .002	.681 ± .004

Model	Google (Syntactic Analogies)	Google (Semantic Analogies)	MSR
CBOW-char + bi. + tri.	**.929 ± .001**	.794 ± .005	**.857 ± .002**
Sent2Vec uni. + bi. + tri.	.837 ± .003	**.853 ± .003**	.745 ± .001
CBOW-char	.920 ± .001	.799 ± .004	.842 ± .002
CBOW	.816 ± .002	.805 ± .005	.713 ± .004
Skip-gram-char	.860 ± .001	.828 ± .005	.796 ± .003
Skip-gram	.829 ± .002	.837 ± .002	.753 ± .005
GloVe	.767 ± .002	.697 ± .007	.678 ± .003
Ngram2vec bi. - bi.	.834 ± .001	.812 ± .003	.761 ± .001

Table 2: Improvement over existing methods: Models are compared using Spearman correlation measures for word similarity tasks and accuracy for word analogy tasks. Top performance(s) on each dataset is(are) shown in bold. The abbreviations uni., bi., and tri. stand for unigrams, bigrams, and trigrams respectively.

WordSim Relatedness (252 word-pairs) (Agirre et al., 2009); *MEN* (3000 word-pairs) (Bruni et al., 2012); *Mechanical Turk* dataset (Radinsky et al., 2011) (287 word-pairs); *Rare words dataset* (2034 word-pairs) (Luong et al., 2013); and *SimLex-999* (999 word-pairs) (Hill et al., 2015) dataset.

To calculate the similarity between two words, we use the cosine similarity between their word representations. The similarity scores then, are compared to the human ratings using Spearman's ρ (Spearman, 1904) correlation scores.

Word-analogy tasks. Word analogy tasks pose analogy relations of the form "x is to y as $x^\star$ is to $y^\star$", where y is hidden and must be guessed from the dataset vocabulary.

We use the *MSR* (Mikolov et al., 2013c) and the *Google* (Mikolov et al., 2013a) analogy datasets. The *MSR* dataset contains 8000 syntactic analogy quadruplets while the *Google* set has 8869 semantic and 10675 syntactic relations.

To calculate the missing word in the relation, we use the 3CosMul method (Levy and Goldberg, 2014):

$$y^\star := \arg \max_{z \in \mathcal{V} \setminus \{x, y, x^\star\}} \frac{cos(\mathbf{v}_z, \mathbf{v}_y) \cdot cos(\mathbf{v}_z, \mathbf{v}_{x^\star})}{cos(\mathbf{v}_z, \mathbf{v}_x) + \varepsilon} \tag{2}$$

where $\varepsilon = 0.0001$ is used to prevent division by 0 and $\mathcal{V}$ is the dataset vocabulary.

We remove all the out of vocabulary words and are left with 6946 syntactic relations for the *MSR* dataset and 1959 word-pairs for the *Rare Words* dataset. All other datasets do not have any out of vocabulary words.

4 Results

Impact of word n-grams. In Table 1, we evaluate the impact of adding contextual word n-grams to two CBOW variations: CBOW-char and Sent2Vec. By adding n-gram information, we consistently observe a boost in the Spearman correlation on the word similarity tasks. On the few datasets where we do not observe an improvement, the results for word-n-gram augmented methods are within standard deviation reach. The *Rare Words* dataset for Sent2Vec is the only exception, despite getting some improvement for CBOW-char based methods. This observation can be attributed to the fact that character ngrams are shared between unigrams, enhancing generalization to infrequent words. Without char n-grams, the model might underfit those rare words, even more so with word n-grams.

We also see that the boost obtained by adding n-grams on word-similarity tasks is much larger for Sent2Vec models as compared to the CBOW-char ones possibly due to the fact that during training, Sent2Vec models use a much larger context and hence can use much more n-gram information for obtaining a better context representation.

For analogy tasks, we see an improvement in the augmented CBOW-char methods for morpho-syntactic analogy datasets with little or no gain for semantic analogy datasets. Yet, for Sent2Vec models, the gain is the other way around. This observation indicates the strong role played by character n-grams in boosting the performance on the syntactic tasks as well as restricting the word n-grams from improving the performance on semantic analogies.

Comparison with competing methods. In Table 2, we compare word n-gram augmented methods with the most prominent word embedding models. We obtain state-of-the-art results for standalone unigram embeddings on most of the datasets confirming our hypothesis. The *Mechanical Turk* dataset is the only exception.

We notice that Sent2Vec trigrams model dominates the word-similarity tasks as well as the semantic analogy tasks. However, character n-grams are quite helpful when it comes to syntactic analogy tasks underlining the importance of subword information. We also note that the Ngram2vec model outperforms our augmented CBOW-char model in some of the tasks but is always inferior to Sent2Vec in those cases.

5 Conclusion and Future Work

We empirically show how augmenting the context representations using higher-order word n-grams improves the quality of word representations. The empirical success also calls for a new theoretical model on the composite effect of training higher order n-grams simultaneously with unigrams. Also, the success of Sent2Vec on word-level tasks, a method originally geared towards obtaining general purposed sentence embeddings, hints towards the additional benefits of using compositional methods for obtaining sentence/phrase representations.

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 *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Z Harris. 1954. Distributional structure.(j. katz, ed.) word journal of the international linguistic association, 10 (23), 146-162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

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

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS - Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

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

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.

A Training parameters for selected models

Training parameters for all models except GloVe and Ngram2vec are provided in Table 3. For the GloVe model , the minimum word count is set to 10; the window size is set to 10; we use 10 epochs for training; X_{max}, the weighting parameter for the word-context pairs is set to 100; all other parameters are set to default. For Ngram2vec, the minimum word count is set to 10; the window size is set to 5; both source and target vectors are trained for unigrams and bigrams; overlap between the target word and source n-grams is allowed. All other features are set to default. To train the Ngram2vec models, we use the library provided by (Zhao et al., 2017)[2].

[2] https://github.com/zhezhaoa/ngram2vec

Model	Sent2Vec uni.	Sent2Vec uni.+bi.	Sent2Vec uni.+bi+tri.	CBOW (char.)	CBOW (char.)+bi.	CBOW (char.)+bi.+tri.	CBOW	Skip-gram (char.)	Skip-gram
Embedding Dimensions	300	300	300	300	300	300	300	300	300
Max Vocab. Size	750k	750k	750k	750k	750k	750k	750k	750k	750k
Minimum Word Count	5	5	5	5	5	5	5	5	5
Initial Learning Rate	0.2	0.2	0.2	0.05	0.05	0.05	0.05	0.05	0.05
Epochs	9	9	9	9	9	9	5	15	15
Subsampling hyper-param.	1×10^{-5}	5×10^{-5}	5×10^{-6}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}
Word-Ngrams Bucket Size	-	2M	4M	-	2M	4M	-	-	-
Char-Ngrams Bucket Size	-	-	-	2M	2M	2M	-	2M	-
Word-Ngrams Dropped per context	-	4	4	-	2	2	-	-	-
Window Size	-	-	-	10	10	10	10	5	5
Number of negatives sampled	10	10	10	5	5	5	5	5	5
Max Char-Ngram Size	-	-	-	6	6	6	-	6	-
Min Char-Ngram Size	-	-	-	3	3	3	-	3	-
Percentage at which training is halted (For CBOW models only)	-	-	-	75%	80%	80%	60%	-	-

Table 3: Training parameters for all non-GloVe models