MultiVec: a Multilingual and Multilevel Representation Learning Toolkit for NLP

Alexandre Bérard, Christophe Servan, Olivier Pietquin, Laurent Besacier

MultiVec: a Multilingual and Multilevel Representation Learning Toolkit for NLP

Alexandre Bérard[†‡], Christophe Servan[‡], Olivier Pietquin[†*] and Laurent Besacier[‡*]
[†]Univ. Lille, CNRS, Centrale Lille, Inria UMR 9189 - CRIStAL, F-59000 Lille, France
alexandre.berard@ed.univ-lille1.fr
olivier.pietquin@univ-lille1.fr

[‡]LIG, Univ. Grenoble Alpes
Campus Saint-Martin d'Hères, Grenoble, France
christophe.servan@imag.fr
laurent.besacier@imag.fr

[*]Institut Universitaire de France, France

Abstract

We present MultiVec, a new toolkit for computing continuous representations for text at different granularity levels (word-level or sequences of words). MultiVec includes Mikolov et al. [2013b]'s word2vec features, Le and Mikolov [2014]'s paragraph vector (batch and online) and Luong et al. [2015]'s model for bilingual distributed representations. MultiVec also includes different distance measures between words and sequences of words. The toolkit is written in C++ and is aimed at being fast (in the same order of magnitude as word2vec), easy to use, and easy to extend. It has been evaluated on several NLP tasks: the analogical reasoning task, sentiment analysis, and crosslingual document classification.

Keywords: Word embeddings, paragraph vector, bilingual word embeddings, crosslingual document classification

1. Introduction

There has been a growing interest in distributed representations for text, largely due to Mikolov et al. [2013a] who propose simple models which can be trained on huge amounts of data. A number of contributions have extended this work to phrases [Mikolov et al., 2013b], text sequences [Le and Mikolov, 2014], bilingual distributed representations [Luong et al., 2015] [Gouws et al., 2015], or bilingual representations for text sequences [Pham et al., 2015].

Although most of these techniques have official or non-official implementations (word2vec, bivec, gensim [Řehůřek and Sojka, 2010], etc.), there has been no concerted effort to regroup all of these techniques in a single toolkit.

Contribution This paper presents MultiVec, a toolkit which enables the generation and manipulation of multilingual vector representations at several granularity levels (from word to any sequence of words). MultiVec combines several techniques of the literature: it includes most of word2vec's features [Mikolov et al., 2013a] for learning distributed word representations (also known as word embeddings); as well as an implementation of Le and Mikolov [2014]'s paragraph vector. In addition, MultiVec can compute bilingual representations on a parallel corpus using Luong et al. [2015]'s bivec model. The code (provided on *GitHub*) is written in C++, and is fast, easy to use and readable. The models can also be used from Python code thanks to a Python wrapper.

We provide the results and code for a number of comparisons and benchmarks that test the usability of this toolkit against other toolkits in the literature.

The MultiVec toolkit has two main components: the first component enables the generation of new models, while the second component uses those models to compute distances between words or sequences. It also includes a series of benchmarks that makes possible the evaluation of trained models on different NLP tasks.

Outline The rest of this paper goes simply as follows: we first describe the models that can be trained using MultiVec. Then, we describe the distance computation features. Finally, we present the benchmarks and their results.

2. Models

2.1. Word embeddings

Mikolov et al. [2013a] offer a simplified version of Bengio et al. [2003]'s neural language model, with a number of tricks to boost performance. They present two models: the continuous bag-of-words model (CBOW), and the skip-gram model.

Given a sequence of words $(w_1, \ldots, w_N)$, the CBOW model learns to predict all words w_i from their surrounding words $(w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k})$. The training objective is to find the parameters C_i and C_o that maximize the log-likelihood of the training corpus:

$$\sum_{i=1}^{N} \log \hat{P}(x = w_i | context = w_{i-k} \ldots w_{i+k}) \quad (1)$$

where the conditional probability $\hat{P}(x = w|context)$ of a word given its context is estimated using the softmax func-

tion over the entire vocabulary V:

$$\hat{P}(x = w|context) = \frac{e^{y^w}}{\sum_{\hat{w} \in V} e^{y^{\hat{w}}}} \quad (2)$$

with:

$$y^w = \frac{1}{2k}(\sum_{x \in context} C_i(x)) \cdot C_o(w) \quad (3)$$

Training is done with the stochastic gradient ascent algorithm, which updates the parameters $\theta = (C_i, C_o)$ after each word w:

$$\theta \leftarrow \theta + \alpha \frac{\partial \log \hat{P}(w|context)}{\partial \theta} \quad (4)$$

C_i and C_o are the input and output weight matrices, which map each vocabulary word w to a weight vector $C_{i/o}(w)$ of size D. As shown in [Mikolov et al., 2013a], after training on a large corpus of text, the embeddings of a word $C_i(w)$ and $C_o(w)$[1] exhibit very interesting linguistic properties.
The skip-gram model has a similar objective function. The key difference is that it uses the current word to predict the context words (reversed direction).

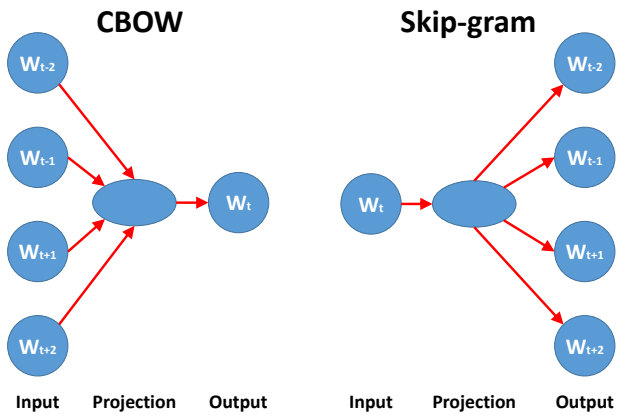

Figure 1: The *CBOW* model predicts w_t based on the context and the *skip-gram* model predicts the context (surrounding words) given w_t.

Evaluating the softmax function is very expensive, as it involves $D^2 \times |V|$ computation steps. To avoid this, Mikolov et al. [2013a] propose to use hierarchical softmax, which reduces the complexity to $D^2 \times \log(|V|)$.
Another way to reduce complexity is to use a different training objective, in which instead of predicting words, we predict whether a word is correct or not. This method is called negative sampling [Mikolov et al., 2013b], or noise-contrastive estimation [Mnih and Kavukcuoglu, 2013].
MultiVec includes both the CBOW and skip-gram model, as well as the hierarchical softmax and negative sampling training algorithms.

2.2. Paragraph vector

Paragraph vector was introduced by Le and Mikolov [2014]. Most machine learning algorithms require their input to be fixed-size vectors. Hence, in order to be processed by such algorithms, variable-size text sequences need to be transformed into a fixed-size representation. This is often done by using the so-called bag-of-words model, which sums the fixed-size representations of the individual components (words or n-grams) of a text sequence.
These representations are either one-hot vectors (whose dimension is the size of the vocabulary), or more compact vector representations (e.g. distributed representations like word2vec). Even though this method is widely used in NLP and IR – and is good enough in some cases – it presents some serious limitations, in particular the loss of any information about word order.
Paragraph vector is an alternative representation that alleviates some of these limitations. The architecture is very similar to the CBOW model in word2vec.[2] It only adds a weight bias vector to the projection layer (of the same size as the word vectors) for each sentence of the corpus. Once the model trained on the entire corpus, each sentence has its distributed representation, which is its corresponding weight vector.

2.2.1. Online paragraph vector

The above method only works for training paragraph vectors in a batch fashion (when the whole corpus is available at once). It is also possible to pre-train a model on a given corpus, and to infer paragraph vectors for new sentences that were not seen in the training corpus. This is done by doing gradient descent on a sentence as usual while freezing the word weights. Le and Mikolov [2014] refer to this method as *inference step*. To the best of our knowledge, ours is the only implementation of this feature.

2.3. Bilingual word embeddings

Word embeddings can be used in multilingual tasks (e.g. machine translation or crosslingual document classification) by training a model independently for each language. However, the resulting representations will be in a different vector space: similar words in different languages will likely have very different representations.
There exist several methods to solve this problem: it is possible to train both models independently and then learn a mapping from one representation to the other; one can also constrain the training to keep the representations of similar words close to each other; or the training can be performed jointly using a parallel corpus.
Luong et al. [2015]'s *bivec* falls into the latter category. This method is especially interesting because it stems directly from Mikolov et al. [2013a]'s *word2vec* and is thus very easy to implement into our architecture, while providing excellent results both on bilingual tasks and monolingual tasks.
For each pair of sentences in a parallel corpus, bivec tries to predict words in the same sentence like word2vec does, but also uses words in the source sentence to predict words in the target sentence (and conversely). Thus, for each update in word2vec, bivec performs 4 updates: source to source, source to target, target to target and target to source.

[1] word2vec only exports the input weights. Our toolkit lets the user export either of them or a sum or concatenation of both.

[2] We describe only the distributed memory model (DM), as the distributed bag-of-word (DBOW) model is not yet implemented in MultiVec.

3. Distance measures

Word embeddings can be used to detect near matches between words. A near match is when two words differ only in terms of morphology or inflection or when they are synonyms or closely related semantically. Near matches are useful to a number of NLP domains, including Information Retrieval and Machine Translation.

The usual way to detect near matches is by using linguistic resources, like WordNet [Princeton University, 2012], BabelNet [Navigli and Ponzetto, 2012] or Dbnary [Sérasset, 2012]. As an alternative to these linguistic databases, our toolkit can detect near matches by measuring the cosine similarity or cosine distance between word representations.

3.1. N-gram comparison

There exist several ways to compare two sequences to each other. As text sequences differ in length, a common way is to use the bag-of-words model which sums the representations of each word of the sequence. This method can be applied to any size of sequence.

We propose a similarity measure for sequences of identical length, typically n-grams. As shown in equation 5, the similarity between the two n-grams $s = (w_1, w_2, \ldots, w_n)$, $s' = (w'_1, w'_2, \ldots, w'_n)$ is obtained by comparing their vector representations $(v_1, \ldots, v_n)$ and $(v'_1, \ldots, v'_n)$ elementwise. Contrary to the bag-of-words model, this method is sensitive to word order.

$$S(s, s') = \frac{S_{cos}(v_1, v'_1) + S_{cos}(v_2, v'_2) + \cdots + S_{cos}(v_n, v'_n)}{n} \quad (5)$$

where $S_{cos}(v, v') = \frac{v^T v'}{\|v\|\|v'\|}$ is the cosine similarity between vector v and vector v'.

4. Benchmarks

This section reports the results of a series of experiments that compares MultiVec to other existing toolkits in the literature.

The main goal is to show that the techniques are correctly implemented by comparing our results with their official implementations (when they exist). We performed experiments on three different tasks: the analogical reasoning task for evaluating the standard (monolingual) word embeddings, the sentiment analysis task for paragraph vector, and the crosslingual document classification (CLDC) task for bilingual representations and paragraph vector.

4.1. Analogical reasoning task

We evaluate our toolkit on the *analogical reasoning task* as described in [Mikolov et al., 2013a]. The authors provide a dataset containing five different types of semantic questions, and nine types of syntactic questions, with a total of 19,558 questions. A question is a tuple $(word_1, word_2, word_3, word_4)$ in which $word_4$ is related (semantically or syntactically) to $word_3$, in the same way that $word_2$ is related to $word_1$. A famous example is $(king, man, queen, woman)$. It has been observed that $C(king) - C(man) \approx C(queen) - C(woman)$, corresponding to some sort of $royalty$ concept. This task evaluates the ability of the model to capture several kinds of linguistic regularities. Other types of questions include for example *state-city* relationships or *adjective-adverb* relationships.

Method	Model	Dim	Synt.	Sem.	Total	Time
word2vec	CBOW	100	35	11.9	28.4	4
		300	**38.6**	16	**32.1**	13
	SG	100	34.4	16.8	29.3	16
		300	30.6	**18.2**	27.1	49
MultiVec	CBOW	100	36.3	11.9	29.3	10
		300	**39.6**	16	**32.8**	17
	SG	100	33.5	17.1	28.8	28
		300	30.8	**20.6**	27.9	60
MV-bi	CBOW	300	**44**	**18.6**	**36.7**	26

Table 1: Results (precision) of the analogical reasoning task, on word2vec's `questions-words.txt`. The models were trained on English Europarl for 20 iterations, with negative sampling (5 samples), with a subsampling rate of 10^{-4}, a window size of 5 and initial training rate of 0.05. *MV-bi* is our bilingual implementation, trained on English-German Europarl for 10 iterations. Training time is given in minutes.

The precision as measured in this task is the percentage of questions for which the closest word in the vocabulary to $word_3 - word_1 + word_2$ according to the cosine similarity is exactly $word_4$.

As shown in table 1, word2vec and MultiVec with the same settings get very similar results.

Interestingly, bilingual models seem to perform significantly better, even on a monolingual task. The number of epochs was intentionally halved in the bilingual case, to make sure that this result is not simply due to a higher number of updates.

4.2. Sentiment analysis task

We evaluate our implementation of paragraph vector on the sentiment analysis task. The same experimental protocol as [Le and Mikolov, 2014] is used[3]. The IMDb dataset contains 100,000 documents. 50,000 of those are labeled with a positive or negative label, and 50,000 are unlabeled. The representations of 25,000 labeled documents are used as training examples for an SVM classifier. The remaining 25,000 labeled documents are used as test examples.

Table 2 reports the results of the different models. We compare the batch paragraph vector implementation provided by Le and Mikolov [2014] with our batch and online implementations. We also report results obtained by simply averaging word embeddings.

As Mesnil et al. [2014] remarked, the results in the original paper were obtained on unshuffled data. This explains why the results reported here are much lower for MultiVec as well as word2vec.

[3] A training script and a modified version of word2vec was provided by the authors on the word2vec Google group.

Toolkit	Method	Training data	Accuracy
word2vec	batch par. vector	train+test	87.5
MultiVec	batch par. vector	train+test	87.8
	online par. vector	train	86.2
	online par. vector	europarl	78
	bag-of-words	train	88.3
	bag-of-words	europarl	77.7

Table 2: Results of the sentiment analysis task on the IMDb dataset. The batch models were trained on training and test data. The online models were trained on either the training data or English Europarl. The settings are: CBOW on 40 iterations, with 15 negative samples, a dimension of 100, window size of 10, learning rate of 0.05 and subsampling of 10^{-4}.

Dim	Toolkit	Method	Accuracy [%] en→de	Accuracy [%] de→en
40	bivec	bag-of-words	86.1	74.4
	MultiVec	bag-of-words	88.1	75.3
		par. vector	88.4	77.6
128	bivec	bag-of-words	89.0	78.6
	MultiVec	bag-of-words	88.9	76.4
		par. vector	88.2	79.1

Table 3: Results obtained within the framework of the CLDC task using the RCV corpus. $en \rightarrow de$ signifies training on English data and testing on German data; $de \rightarrow en$ is the reverse. The settings are the same as those in [Luong et al., 2015]: skip-gram model, 30 negative samples, 10 epochs.

4.3. Crosslingual document classification task

To evaluate the quality of our bilingual word embeddings, we reproduce Klementiev et al. [2012]'s experiments on the crosslingual document classification task. This task consists in classifying documents in a language using a model that was trained with documents from another language. Like Klementiev et al. [2012] and Luong et al. [2015], 1000 documents from the RCV corpus are used for training, and 5000 documents for testing. Each document belongs to one of 4 categories. Document representations are computed by doing a weighted sum of word embeddings, according to pre-defined word frequencies (TF-IDF). A perceptron classifier is then trained on the source-language documents and evaluated on target-language documents.[4]

We compare bilingual models trained with bivec and MultiVec. We also show results obtained by computing document representations with online paragraph vector. To do so, we export the previously trained bilingual model to source and target models, which are then used to compute paragraph vector representations for source and target documents.

Table 3 shows similar results for both MultiVec and bivec. Paragraph vector does no better than the bag-of-words representation, but the results confirm that our approach for computing bilingual paragraph vectors is sound.

[4]The data splits and training scripts were obtained from the authors.

5. Conclusion

In this paper we presented MultiVec, a toolkit which aggregates a number of techniques in the literature that compute distributed representations of text. It includes word2vec, paragraph vector (batch and online), and bivec. All these techniques fit nicely and are interoperable with each other. The toolkit is designed for being easy to set-up and use, while also being easy to dive into. The project is fully open to future contributions.

The code is provided on the project webpage[5] with installation instructions and command-line usage examples.

As future work, we plan on implementing a number of features, including but not only (see project webpage) bivec's *UnsupAlign* model, which uses word alignment information (from GIZA++); and the distributed bag-of-word (DBOW) model for paragraph vector.

Acknowledgments

This work was supported by the KEHATH project funded by the French National Agency for Research (ANR) under the grant number ANR-14-CE24-0016-03 .

References

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research (JMLR)*, 3:1137–1155, 2003.

S. Gouws, Y. Bengio, and G. Corrado. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2012.

Q. V. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014.

T. Luong, H. Pham, and C. D. Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015.

G. Mesnil, M. Ranzato, T. Mikolov, and Y. Bengio. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv:1412.5335 [cs]*, 2014.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *The Workshop Proceedings of the International Conference on Learning Representations (ICLR)*, May 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, 2013b.

[5]`https://github.com/eske/multivec`

A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

R. Navigli and S. P. Ponzetto. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250, 2012.

H. Pham, M.-T. Luong, and C. D. Manning. Learning Distributed Representations for Multilingual Text Sequences. In *Proceedings of NAACL-HLT*, 2015.

Princeton University. About WordNet. Technical report, Princeton University, 2012.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

G. Sérasset. Dbnary: Wiktionary as a LMF based Multilingual RDF network. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, 2012.