Schedae Informaticae Vol. 25 (2016): 127–138
doi: 10.4467/20838476SI.16.010.6191

tfml 2017
theoretical foundations
of machine learning, Kraków

Word Embeddings for Morphologically Complex Languages*

Grzegorz Jurdziński
Department of Theoretical Computer Science
Faculty of Mathematics and Computer Science of the Jagiellonian University
ul. prof. Stanisława Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: *grzegorz.jurdzinski@student.uj.edu.pl*

Abstract. Recent methods for learning word embeddings, like GloVe or Word2-Vec, succeeded in spatial representation of semantic and syntactic relations. We extend GloVe by introducing separate vectors for base form and grammatical form of a word, using morphosyntactic dictionary for this. This allows vectors to capture properties of words better. We also present model results for word analogy test and introduce a new test based on WordNet.

Keywords: machine learning, word embeddings, natural language processing, morphology

1. Introduction

Word embedding methods assign vectors in continuous n-dimensional space to words from a language. These can be used for various tasks, such as information retrieval [1], document classification [2], question answering [3], named entity recognition [4] and parsing [5].

Most word vector methods are supposed to cluster words that have similar meaning and their performance was evaluated based on experiments testing distance or

Received: 11 December 2016 / Accepted: 30 December 2016

* First version of this work was prepared as Bachelor Thesis at Institute of Computer Science of University of Wrocław. Its preparation was supervised by Jan Chorowski Ph.D., Institute of Computer Science, Faculty of Mathematics and Computer Science of the University of Wrocław, ul. Joliot-Curie 15, 50-383 Wrocław (e-mail: *jan.chorowski@cs.uni.wroc.pl*)

angle between pairs of words. [6] introduced a more complex evaluation scheme. It is based on word analogies that examine finer structure of word vector space on various dimensions of difference. For example, the analogy *"king is to queen as man is to woman"* should be encoded in vectors by equation *vector("king") – vector("queen") = vector("man") – vector("woman")*. Indeed many of mentioned word embedding methods produce representations encoding such relations well.

Models like Word2Vec [7] and GloVe [8] receive worse scores on syntactic part of this test. The idea of our solution is to produce separate vectors for the base form of a word and the set of tags describing its grammatical form (called tagset in further part of this work). For example for Polish word *"jabłek"* (genitive case of word *"apples"*) its base form is *"jabłko"* (*"apple"*) and its tagset is *"subst:pl:gen:n2"* (describing that it is a plural form of a noun in genitive case with neuter grammatical gender). Such decomposition of a word can be obtained using morphosyntactic dictionaries (e.g. Polimorfologik for Polish). Then, during learning, vector for each word is represented as sum of vectors of its base form and tagset. One of benefits of such approach is giving the model a possibility to gather more information about rare words. Models mentioned above treat occurrences of the same word in different grammatical forms as separate words. For example there is no direct connection between *"jabłkami"* and *"jabłku"*. Out model has a common base form vector for all forms of *"jabłko"* so it is able to make use out of each occurrence of word regardless its grammatical form.

1.1. Related work

Word embedding was analyzed and implemented in various works. We shortly describe some important examples below.

Feedforward Neural Net Language Model (NNLM) [9] uses word vectors as its parameters. The network itself models the language – that means than when fed with N words (where N is a fixed, chosen number) it produces a probability distribution over all words from the language. For each word it should be the probability of appearing after the N given words. Part of the neural network is a shared matrix of word vectors. NNLM consist of input, projection, hidden and output layers. In the input layer N previous words are encoded using 1-of-V coding (where V is size of the vocabulary), then it is projected to a projection layer P that has dimensionality $N \cdot D$ (where D is the dimensionality of word vectors), using a shared projection matrix. That means that each 1-of-V vector is replaced with a word vector from the shared matrix and then all of them create one big $N \cdot D$ vector. Between the projection and hidden layers there is a dense connection and results of the hidden layer are used by the output layer to compute a probability distribution over all words in the vocabulary using the softmax function.

Word2Vec [7, 10] implements two models – Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model (Skip-gram). Both models were based on NNLM. They consist of input, projection and output layers. CBOW, after projecting words to their vectors, averages them to one vector (NNLM was joining them to create bigger vector) and uses an output layer to predict the word. Also, instead of predicting next word, after given N words, it predicts middle word given words within certain range. Skip-gram model, instead of predicting a single word based on context, predicts the context itself. Authors of Word2Vec made many improvements with respect to the complexity of the model. They have replaced softmax with its efficient approximations – they have tested Hierarchical Softmax and Negative Sampling [7].

Problem of representing morphology in word embeddings was already tackled before. One of the examples are morphoRNN [11] and [12]. Both of these works split words into morphemes and learn separate vectors for theme. For example word *"unfortunately"* would be split into *"un"*, *"fortunate"* and *"ly"*. To get a vector for a word [12] sum up vectors of its morphemes. They also learn a separate vector for each word and add it to its morphemes vectors, so for example vector for word *"greenhouse"* would be *vector("greenhouse") + vector("green") + vector("house")* (final vector for the word *"greenhouse"* and the vector used to compute it are two different vectors). [11] present more complex way of combining morpheme's vectors. To produce a word vector a small neural network is used. At each step one affix and word stem are combined. A new vector is produced from stem vector ($x_{\texttt{stem}}$) and affix vector($x_{\texttt{affix}}$) as follow: $p = f(W_m[x_{\texttt{stem}}; x_{\texttt{affix}}] + b_m)$. Vectors for stem and affix are combined into bigger vector and multiplied by the matrix of parameters (W_m) and bias vector (b_m) is added. f is an element-wise activation function (tanh for example). So for example a vector for a word *("unfortunately")* would be computed in two steps. First vector for *"unfortunate"* would be computed by combining vectors for *"un"* and *"fortunate"*. Then vectors for *"unfortunate"* and *"ly"* would be combined to give vector for *"unfortunately"*.

The main difference between these two works and the approach we present is the way of splitting the words – instead of looking at the morphemes the word consists of we take its base form and the set of the tags describing its grammatical form.

GloVe was introduced by [8]. Since our work is based on it, we will describe it precisely in next section.

2. GloVe model

GloVe model, introduced by [8], utilizes two approaches to the problem – matrix factorization methods and shallow window-based methods. First uses a large matrix that captures statistical information about a corpus, e.g. rows can correspond to terms, columns to documents in the corpus and cells are numbers of occurrences of term in document. Such approach is used by [13] in latent semantic analysis (LSA). Examples of shallow window-based methods are NNLMs and Word2Vec. Their approach is to learn word representation which aim is to predict word given a local context window of a few, typically 5–15 words.

Before learning vectors a co-occurrence matrix X is created by counting word co-occurrences in a corpus. For that the context window size is being chosen – let's denote it as ws. We will say that a word j occurs in context of a word i if it occurs in the corpus within ws distance from i. $X \in \mathbb{N}^{V \times V}$ (where V is size of vocabulary

– number of different words in corpus) is a word-word co-occurrence matrix, that means X_{ij} is number of occurrences of word j in context of word i. We will call word j a *context word.*

Word vectors are learned based on the matrix X. For each word there are two separate vectors in GloVe – the *word vector* and the *context vector*. Let w_i be the word vector of word i and $\tilde{w}_i$ be the context vector of word i. Analogically there are two biases for each word – b_i and $\tilde{b}_i$.

Authors of GloVe claim that word vectors should satisfy the following equation ([8] give full justification for this formula):

$$w_i^T \tilde{w}_j + b_i + \tilde{b}_j = \log(X_{ij}) \tag{1}$$

For learning vectors equation 1 is treated as least squares problem. This results in a cost function defined as following:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik}\right)^2 \tag{2}$$

where $f(X_{ij})$ is a weighting function. Main reason for introducing it is to prevent rare co-occurrences from influencing the model too strongly. According to Pennington et al. weighting function should satisfy the following properties:

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \rightarrow 0$ fast enough that the $\lim_{x\rightarrow 0} f(x)\log^2 x$ is finite.

2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.

3. $f(x)$ should be relatively small for large values, so that frequent co-occurrences are not overweighted.

Authors of GloVe were using the following weighting function:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where $x_{\max}$ and α are left as free parameters. They have discovered that $x_{\max}$ does not influence the model strongly and were using $x_{\max} = 100$ for all their tests. They have also found that $\alpha = 3/4$ gives slight improvement over $\alpha = 1$.

The cost function (2) is minimized using the AdaGrad algorithm [14] – variation of regular Gradient Descent. At each step a pair of words i and j is being chosen and the cost of approximating X_{ij} is being computed. Every parameter (vectors and biases) is then being updated using gradient. A full iteration of the algorithm goes over all pairs of words. The program is being run for a fixed number of iterations.

Resulting vectors create many clusters – words with similar meaning or grammatical form are grouped together. Unfortunately often syntactics of words is not represented by these clusters so words considering similar subject but with completely different grammatical form are close.

3. Extentded GloVe model

The main idea of our extension of GloVe model is to replace word vectors with vectors for base forms and tagsets – for each word its vector will be the sum of vectors of its base and grammatical form. It requires finding the base form and the tagset for each word in the vocabulary and changing the model itself. Such approach gives less freedom to the model – regular GloVe could easily place word vectors in most suitable place, here it is not always possible since base forms vectors and tagsets vectors are shared by various words – but it also gives extra information that model can use.

3.1. Obtaining base forms and tagsets for words

For obtaining base forms and tagsets for words we have used the morphosyntactic dictionary Polimorfologik [15]. For each word it defines its base form and a set of tags describing its syntactic form. If its ambiguous, several tagsets, separated with plus sign, are defined. An example line of Polimorfologik looks like this:

```
kot;kotami;subst:pl:inst:m1+subst:pl:inst:m2
```

The example word is *"kotami"* (*"cats"* in instrumental case). The first column is the base form – `kot` (*"cat"*) here. The second column is the word itself – `kotami`. The third column – `subst:pl:inst:m1+subst:pl:inst:m2` – is the set of tags describing grammatical form of the word. Two tags (`subst:pl:inst:m1` and `subst:pl:inst:m2`) are separated by the `+` sign due to the ambiguouity of the grammatical form of the word. Both contain parts describing that it is a noun (`subst`), in plural form (`pl`), in instrumental case (`inst`). The difference between them is the part responsible for grammatical gender of the word (`m1` and `m2`). The word *"kotami"* in Polish can be interpreted as as in two of three possible musculine grammatical genders – personal and animate.

We have prepared scripts extracting base forms and tagsets for words from vocabulary, putting them in separate files and creating a file that for each word contains line with its base form and tagset. For words that were not found in the Polimorfologik we have the word itself to be its base form and assigned an empty grammatical tag. Therefore all unknown words are assigned to the same grammatical tagset. Please note that this doesn't preclude learning correct embeddings for words not found in Polimorfologik – in fact, since the base form of such words is unique their embedding is not shared with other words and the model is free to place them whenever is appropriate in the embedding space so the results for these words should be similar to the regular GloVe.

3.2. Learning separate vectors

Introducing separate vectors for base forms and tagsets required changes in model. First denote J_{ij} as single element of a sum from equation 2 for words i and j. Let $\text{fdiff} = f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)$ and $w_{i,k}$ be k^{th} element of vector w_i. Then we have

$$\begin{aligned}\frac{\partial J_{ij}}{\partial w_{i,k}} &= \frac{\partial}{\partial w_{i,k}} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j + \log X_{ij})^2 \\ &= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} w_{i,l}\tilde{w}_{j,l} + b_i + \tilde{b}_j - \log X_{ij} \right) \\ &= 2 \cdot \text{fdiff} \cdot \tilde{w}_{j,k}\end{aligned} \tag{4}$$

where n is dimensionality of vectors.

For learning separate vectors for base forms and tagsets we replaced word vectors with sums of vectors of its forms, resulting with following cost function:

$$J^{(2)} = \sum_{i,j=1}^{V} f(X_{ij})((w_i + v_i)^T(\tilde{w}_j + \tilde{v}_j) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij})^2 \tag{5}$$

where w_i and v_i are vectors of base form and tagset of word i (analogically for context word and biases).

Let $J_{ij}^{(2)}$ be analogical to J_{ij} for equation 5 and fdiff to the previous definition. Than we have:

$$\begin{aligned}\frac{\partial J_{ij}^{(2)}}{\partial w_{i,k}} &= \frac{\partial}{\partial w_{i,k}} f(X_{ij})((w_i + v_i)^T(\tilde{w}_j + \tilde{v}_j) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij})^2 \\ &= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} (w_{i,l} + v_{i,l})(\tilde{w}_{j,l}\tilde{v}_{j,l}) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij} \right) \\ &= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} (w_{i,l} + v_{i,l})(\tilde{w}_{j,l}\tilde{v}_{j,l}) \right) \\ &= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} (w_{i,k}\tilde{w}_{j,k} + w_{i,k}\tilde{v}_{j,k} + v_{i,k}\tilde{w}_{j,k} + v_{i,k}\tilde{v}_{j,k}) \\ &= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} (w_{i,k}\tilde{w}_{j,k} + w_{i,k}\tilde{v}_{j,k}) = 2 \cdot \text{fdiff} \cdot (\tilde{w}_{j,k} + \tilde{v}_{j,k})\end{aligned} \tag{6}$$

(analogically for other vectors and biases).

We have set $x_{\max} = 100$ and $\alpha = 3/4$ since these were the values that gave best results for GloVe.

By separating vectors responsible for meaning and grammar of words the model can easily group one type of vectors by semantics of words and second one by syntactics.

4. Experiments

4.1. Corpora and training details

For Polish language we have trained my model on a 2016 Wikipedia dump with over 350 billion words. We lowercased the corpus and removed punctuation marks with simple script, built a vocabulary of words appearing in the corpus at least 20 times (resulting with over 400 million words vocabulary). Then we built the co-occurrence matrix X using a window size 10.

For all experiments we set $x_{\max} = 100$ and $\alpha = 3/4$. The model was trained using asynchronous AdaGrad, stochastically sampling non-zero elements from X (co-occurrences are randomly shuffled after counting and before starting learning), with initial learning rate set to 0.05. We have trained our model for vectors of size 50, 100, 200 and 300.

4.2. Word analogies test

It has been observed that words with the same grammatical form or similar meaning tend to spread among subspace of original vector space. This suggests that some part of vectors might be responsible for certain features of words. Following this observation [6] has proposed new technique to measure quality of word vector representations. It intends not only to check whether similar words are close to each other but also to investigate multiple degrees of similarity.

Similarities of word vector representations apply not only to syntactic properties of words but also to semantic. [10] describe word offset technique that uses simple algebraic operations applied to vectors . For example, as we have mentioned in the introduction, *vector("king") – vector("man") + vector("woman")* results with a vector that is closest to the vector representation of word *queen*.

To examine these regularities Mikolov et al. has introduced the *word analogy test*. It consists both of semantic and syntactic tests divided into categories. Each file consists of one type of regularity. In each line there are two pairs of words

For our needs we have translated files with these tests to Polish and extended with tests for regularities that are absent in English. Examples of each categories are presented in Table 1. We have avoided multi-word entities (like *New York*). Overall there are 8869 semantic and 10000 syntactic questions.

We evaluate vectors accuracy for all question types and for each question type separately (semantic, syntactic). Question is assumed to be answered correctly if and only if the nearest word to the vector computed suing method described above is exactly the word from question.

GloVe model vectors score better results in the semantic part of the test. Learning separate vectors for base and grammatical forms of words gives the model extra

Table 1. Examples of word analogies divided into types.

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Ateny	Grecja	Oslo	Norwegia
All capital cities	Astana	Kazachstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	Kalifornia
Man-Woman	brat	siostra	wnuk	wnuczka
Adjective to adverb	niezwykły	niezwykle	pozorny	pozornie
Opposite	stały	niestały	świadomy	nieświadomy
Comparative	zły	gorszy	miękki	miększy
Superlative	duży	największy	bliski	najbliższy
Nationality adjective	albania	albański	japonia	japoński
Past tense	tańczy	tańczył	zmniejsza	zmniejszał
Plural nouns	ptak	ptaki	butelka	butelki
Grammatical gender	jadł	jadła	znalazł	znalazła
Nominative-genitive	kalafior	kalafiora	pierwiastek	pierwiastka

information about morphology. It results with much better scores for syntactic part of the word analogy test but it also lowers scores for semantic part.

We have experimented with two ways of computing vectors for each word – adding vectors of its base form and tagset (BF + TS) and adding also vectors of the word itself (the one computed for the regular GloVe; BF + TS + word). Similar technique to the second one was used in [12]. Results are presented in Table 2.

Table 2. Accuracy for word analogy test.

Model	Dim.	Sem. [%]	Syn. [%]	Tot. [%]
GloVe	50	25.57	18.97	22.36
BF+TS	50	13.92	37.17	25.23
BF+TS+word	50	17.46	24.61	20.94
GloVe	100	46.47	25.36	36.20
BF+TS	100	22.10	41.46	31.59
BF+TS+word	100	31.57	37.77	34.59
GloVe	200	56.96	28.43	43.09
BF+TS	200	21.69	44.21	32.64
BF+TS+word	200	42.83	46.24	**44.49**
GloVe	300	**57.64**	28.64	43.54
BF+TS	300	20.15	43.46	31.49
BF+TS+word	300	40.97	**48.01**	44.39

These results show that when splitting a word into its base and grammatical form the model tends to loose information about semantics of the word for its syntactic. Possible reason for that is lower freedom of placing vectors for the extended model than in the regular one. Both base forms vectors and tagsets vectors are shared by many various words what makes it much harder for the model to place word vector in most convenient location. Grammatical properties represented by tagsets are always

Table 3. Weighted vector sums accuracy for dimensionality 300.

BF weight	TS weight	word weight	Sem. [%]	Syn. [%]	Tot. [%]
Regular GloVe			**57.64**	28.64	43.54
0.6	0.4	0	14.53	59.27	36.29
0.7	0.3	0	11.55	55.31	32.83
0.8	0.2	0	9.42	51.18	29.73
0.4	0.3	0.3	39.22	57.51	48.11
0.3	0.3	0.4	51.87	50.53	**51.22**
0.4	0.2	0.4	46.68	49.35	47.98
0.4	0.4	0.2	30.60	**61.57**	45.66

strictly defined and base forms can be ambiguous what might make tagsets vectors „dominate" base forms vectors. For this reason we have tested assigning different weights to base form, tagset and word vectors when summing these vectors. Results for vectors of dimensionality 300 are presented in Table 3. For comparison we have also included results for the regular GloVe in the first row.

Our model outperforms regular GloVe in the syntactic part of the test for any weights. Even though regular GloVe is still better for the semantic part of the test, our model receives almost equal score for some weights. What is also worth noticing is the fact that for weights 0.3, 0.3, 0.4 our model get best total result out of all combinations we have tested and also very good scores for both semantic and syntactic parts of the test. That gives us a model that balances between semantic and syntactic performance well.

4.3. Wordnet

For this test we have used the polish wordnet – *Słowosieć* [16]. Wordnet is a semantic dictionary reflecting lexical system of polish language. Entries of wordnet are connected by relations creating net. For example *car* is a type of *vehicle*, consists of *wheels*, *engine* and others and its synonyms include *automobile* and *truck*. *Słowosieć* has been created together by computer scientists and linguists.

For our tests we have used the shortest path distance in wordnet graph as word similarity measure. We have defined test set, consisting of 72 popular words and 64 rare words. For each of them we have found 5–20 closest word vectors. Then we have computed the average distance of their base forms (wordnet contains only base forms of words) in wordnet graph from starting word.

Like in the word analogy test, we have tested regular GloVe vectors, sums of base and grammatical forms and sums of all three. Since the wordnet test examines only semantic features of word vectors we have also tested it for base form vectors (for each word we have assigned vector of its base form as its vector). Results are presented in Table 4. We have tested it for 10 closest words.

As it was expected, best results were scored for base form vectors – this test examines only semantics of words so tagset vectors only "disturb". The model we

Table 4. Accuracy for wordnet test for 10 closest words.

Model	Dim.	Popular words	Rare words	Total
GloVe	100	2.612	1.824	2.272
BF+TS	100	2.904	2.476	2.674
BF+TS+word	100	2.907	2.215	2.563
BF	100	0.593	0.605	0.599
GloVe	200	2.559	1.471	2.088
BF+TS	200	2.730	2.390	2.550
BF+TS+word	200	2.696	2.453	2.583
BF	200	0.434	**0.546**	**0.491**
GloVe	300	2.541	1.686	2.175
BF+TS	300	2.701	2.273	2.473
BF+TS+word	300	2.797	2.416	2.617
BF	300	**0.396**	0.595	0.498

have implemented has the advantage of gaining information about the meaning of word regardless of its grammatical form so no matter in what case the word occurs, it contributes to learning of the meaning of all of its forms.

5. Conclusion

In our work we have presented, implemented and tested methods to enhance word vectors performance. Feeding model with additional information about base and grammatical forms of words allows it to extract semantic and syntactic information better. It also allows the model to use the corpus more efficiently – occurrences of rare words in different forms are connected and we are able to gather more information about them. We have also translated to Polish set of questions for words analogy test prepared by [10] for testing my model and we have introduced new test basing on *Słowosieć* [16].

5.1. Future work

Several improvements can still be tested. One of them can be a syntactic tagging of corpus for word disambiguation. In our work for each word we have chosen its base and grammatical form not regarding its context. If its grammatical form was ambiguous a tagset was created.

Testing another models, like Skip-gram and CBOW [10], might also help to investigate capabilities of splitting word vectors. Although models implemented in Word2Vec and GloVe are somehow similar, the first ones are more complex what might give the opportunity to capture language structure better.

Another methods of constructing word vectors from vectors of its base forms and tagsets could be tested too. [11] construct word vectors with small neural network, what gives the model more flexibility and makes the process of combining vectors learnable. Another, simpler method could be concatenating two vectors into one bigger (so if vectors of base forms and tagsets would be from $\mathbb{R}^n$ the resulting vector would be from $\mathbb{R}^{2n}$).

Word embeddings are useful as a way of preprocessing data. When used in bigger model it is hard to tell how the vectors affect the model exactly and which of its parts should be improved. Due to this fact the performance of our vectors on downstream tasks could be also tested and discussed in future. [17] present more methods for evaluating word embeddings.

Acknowledgements

The authors would like to acknowledge the support of the National Science Center (Poland) grant Sonata 8 2014/15/D/ST6/04402 and thank the Wroclaw Center for Networking and Supercomputing for donating computer time.

6. References

[1] Manning C.D., Raghavan P., Schütze H., *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[2] Sebastiani F., *Machine learning in automated text categorization.* ACM computing surveys (CSUR), 2002, 34 (1), pp. 1–47.

[3] Tellex S., Katz B., Lin J., Fernandes A., Marton G., *Quantitative evaluation of passage retrieval algorithms for question answering.* In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 41–47.

[4] Turian J., Ratinov L., Bengio Y., *Word representations: a simple and general method for semi-supervised learning.* In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 384–394.

[5] Socher R., Bauer J., Manning C.D., Ng A.Y., *Parsing with compositional vector grammars.* In: *ACL (1)*, 2013, pp. 455–465.

[6] Mikolov T., Yih W.t., Zweig G., *Linguistic regularities in continuous space word representations.* In: *HLT-NAACL.* vol. 13., 2013, pp. 746–751.

[7] Mikolov T., Sutskever I., Chen K., Corrado G.S., Dean J., *Distributed representations of words and phrases and their compositionality.* In: *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

[8] Pennington J., Socher R., Manning C.D., *Glove: Global vectors for word representation.* In: *EMNLP.* vol. 14., 2014, pp. 1532–43.

[9] Bengio Y., Ducharme R., Vincent P., Jauvin C., *A neural probabilistic language model.* Journal of Machine Learning Research, 2003, 3 (Feb), pp. 1137–1155.

[10] Mikolov T., Chen K., Corrado G., Dean J., *Efficient estimation of word representations in vector space.* CoRR, 2013, abs/1301.3781.

[11] Luong T., Socher R., Manning C.D., *Better word representations with recursive neural networks for morphology.* In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, 2013, pp. 104–113.

[12] Botha J.A., Blunsom P., *Compositional morphology for word representations and language modelling.* In: *ICML*, 2014, pp. 1899–1907.

[13] Deerwester S., Dumais S.T., Furnas G.W., Landauer T.K., Harshman R., *Indexing by latent semantic analysis.* Journal of the American Society for Information Science, 1990, 41 (6), pp. 391.

[14] Duchi J., Hazan E., Singer Y., *Adaptive subgradient methods for online learning and stochastic optimization.* Journal of Machine Learning Research, 2011, 12 (Jul), pp. 2121–2159.

[15] Miłkowski M., *Polimorfologik.* `https://github.com/morfologik/polimorfologik` 2016.

[16] Maziarz M., Piasecki M., Szpakowicz S., *Approaching plWordNet 2.0.* In: *Proceedings of the 6th Global Wordnet Conference*, January 2012.

[17] Schnabel T., Labutov I., Mimno D.M., Joachims T.,*Evaluation methods for unsupervised word embeddings.* In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17–21, 2015*, 2015, pp. 298–307.