# СРАВНЕНИЕ ТРЕХ СИСТЕМ СЕМАНТИЧЕСКОЙ БЛИЗОСТИ ДЛЯ РУССКОГО ЯЗЫКА

**Арефьев Н. В.** (narefjev@cs.msu.su)
Московский Государственный Университет им. Ломоносова и ООО «Лаборатория Цифрового Общества», Москва, Россия

**Панченко А. И.** (panchenko@lt.informatik.tu-darmstadt.de)
Дармштадский Технический Университет, Дармштадт, Германия

**Луканин А. В.** (artyom.lukanin@gmail.com)
ООО «СофтПлюс», Челябинск, Россия

**Лесота О. О.** (cheesemaid@gmail.com)
Московский Государственный Университет им. Ломоносова, Москва, Россия

**Романов П. В.** (romanov4400@gmail.com)
ОАО 1С, Москва, Россия

Статья представляет результаты участия в дорожке по семантической близости RUSSE. Мы сравниваем три подхода к оценке семантической близости слов. Данные подходы основаны на использовании корпусов текстов русского языка. В первом подходе используются лексико-синтаксические шаблоны для извлечения и разметки предложений, содержащих слова, находящихся в гипо-гиперонимических отношениях. Второй подход — это классический метод контекстного окна на данных Google N-Grams. В третьем подходе используется программа *word2vec* и большой корпус для создания векторов слов. Последний метод считается лучшим методом для английского языка. Наши эксперименты показывают, что он также является лучшим методом для русского языка. В данной статье мы анализируем, как изменение метапараметров *word2vec* и использование различных корпусов, на которых он обучается, влияет на качество получаемых векторов слов. Мы также предлагаем простую, но действенную, методику по учёту слов, отсутствующих в словаре.

**Ключевые слова:** семантическая близость, лексико-синтаксические шаблоны, Google N-Grams, контекстное окно, word2vec, RUSSE, русский язык

# EVALUATING THREE CORPUS-BASED SEMANTIC SIMILARITY SYSTEMS FOR RUSSIAN

**Arefyev N. V.** (narefjev@cs.msu.su)
Lomonosov Moscow State University &
Digital Society Laboratory, Moscow, Russia

**Panchenko A. I.** (panchenko@lt.informatik.tu-darmstadt.de)
TU Darmstadt, Darmstadt, Germany

**Lukanin A. V.** (artyom.lukanin@gmail.com)
LLC "SoftPlus", Chelyabinsk, Russia

**Lesota O. O.** (cheesemaid@gmail.com)
Lomonosov Moscow State University, Moscow, Russia

**Romanov P. V.** (romanov4400@gmail.com)
1C Company, Moscow, Russia

This paper reports results of our participation in the first shared task on Russian Semantic Similarity Evaluation (RUSSE). We compare three corpus-based systems that measure semantic similarity between words. The first one uses lexico-syntactic patterns to retrieve sentences indicating a particular semantic relation between words. The second one builds traditional context window approach on the top of Google N-Grams data to take advantage of the huge corpora it was collected on. The third system uses *word2vec* trained on a huge *lib.rus.ec* book collection. *word2vec* is one of the state-of-the-art methods for English. Our initial experiments showed that it yields the best results for Russian as well, comparing to other two systems considered in this paper. Therefore, we focus on study of *word2vec* meta-parameters and investigate how the training corpus affects quality of produced word vectors. Finally, we propose a simple but useful technique for dealing with out-of-vocabulary words.

**Keywords:** semantic similarity, lexico-syntactic patterns, skip-gram model, Google n-grams, context window, word2vec, RUSSE, Russian language

## 1. Introduction

A semantic similarity measure (SSM) outputs words with close meaning to an input word. For instance, such system can take as input the word "python" and return a list of related words, such as "perl", "ruby", "snake", "reptile" and "holy grail" (see serelex.org/#python). Similarity can be interpreted in many ways. In this paper, we consider words similar if they are synonyms, hypernyms or free (cognitive) associations, depending on the task. SSMs can be *global* and *contextual*. A global measure does not consider any context and therefore will return a mix of senses for ambiguous words, such as "python". On the other hand, contextualized SSMs take into account

context and therefore can filter out irrelevant results for a given word occurrence. Usually, a similarity measure returns a weighted (or ranked) list of results. However, most often, such a list contains a mix of synonyms, hyponyms, associations, co-hyponyms and other related words without explicit distinction between them.

The main motivation for development of SSMs is the wide range of language processing applications, they can be applied in, ranging from lexical substitution and word sense disambiguation to query expansion and question answering. No wonder many researchers tried to propose SSMs during the last two decades. In particular, most of the methods rely on a text corpus in order to estimate word similarities, for instance the classical distributional models, such as the context window and the syntactic context techniques. However, there exist many other original approaches that are built upon the structure of a lexical network, counts of a web search engine or entries of a dictionary. One of the recent trends in this field is corpus-based models that use a neural network to train word vectors used for similarity computation. The skip-gram model used in our work is one of them (Mikolov et al., 2013). You will be able to find exhaustive references to the mentioned above techniques in multiple comparisons of SSMs, such as Lee (1999), Agirre et al. (2009), Ferret (2010), Panchenko (2013) and Baroni (2014).

While there exist many approaches to semantic similarity, most of them were tested only for English. On the other hand, the Russian language has several important features that make it quite different from English: a grammar system with complex morphological rules, very flexible word order, absense of articles and Cyrillic alphabet. It is therefore premature to take for granted that the approaches yielding good results for English are going to work as well in the context of Russian. A recent paper by Zervanou et al. (2014) provides a study of semantic similarity for morphologically rich languages, including German and Greek, however Russian is not considered in the experiment. Finally, several researchers already tried to apply distributional semantic models for the Russian language including Krizhanovski (2007), Turdakov (2010), Krukov et al. (2010), Sokirko (2012) and Kolb[1]. However, these experiments lack a systematic evaluation of semantic similarity measures for Russian. Indeed, the workshop on Russian Semantic Similarity Evaluation *RUSSE* (Panchenko et al., 2015) introduced the first large-scale publicly available evaluation framework tailored for the Russian language. In this work, we use this collection of novel benchmarks to assess performance of our approaches[2].

Main contribution of our work is a comparative study of three global corpus-based systems of semantic similarity for the Russian language that are based respectively on the lexico-syntactic patterns, the right side context window and the skip-gram model. To the best of our knowledge, this is the first public attempt to quantify performance of these three approaches in the context of the Russian language. We experimentally assess performance of these techniques in the context of a shared task on a Russian semantic similarity, where the proposed methods consistently score in the top 10 models in all tracks. Systems and models described in this paper are available online (see below). In particular, to the best of our knowledge, we are the first to release a large scale *word2vec* model for the Russian language.

---

[1] http://www.linguatools.de/disco/disco_en.html

[2] https://github.com/nlpub/russe-evaluation/tree/master/russe/evaluation, http://russe.nlpub.ru

## 2. The First System: Pattern-Based Similarity on Wikipedia and Web corpora

The *PatternSim* similarity measure was first introduced for English language by Panchenko et al. (2012). The method operates in two steps. First, it extracts a set of sentences, which contain similar words, and tags these words with a set of manually crafted lexico-syntactic patterns. Second, it calculates a semantic similarity between words based on several factors, such as a term frequency and the number of term co-occurrences within sentences. Implementation of the method is available online[3].

### 2.1. Corpora

We used two corpora in order to calculate the semantic similarity with the *PatternSim* measure: the Russian Wikipedia and a collection of Russian Web pages. The Wikipedia dump was downloaded in April 2014 and processed with the *WikpediaExtractor.py* script[4]. The corpus of Web pages was crawled from the pages of 2,736 web sites each belonging to one of 20 following topical categories: auto-moto, beauty, child wares, clothes, clubs-concerts-cinema, cookery, credits, eating-out, everyday wares, furniture, insurance, information technology, massage, medicine, politics, realty, religion, repair wares, sport, travel. The seed web sites and the corpus itself are available for download[5].

**Table 1.** Corpora used by the three similarity measures described in this paper

| Name | Description | Tokens | Documents | Size, Gb |
|---|---|---|---|---|
| wiki | Russian Wikipedia | 238,052,379 | 1,159,723 | 3 |
| web | Russian Web Pages | 567,914,057 | 890,551 | 7 |
| lib | Lib.rus.ec book collection | 12,902,854,351 | 233,876 | 149 |
| ngram | Russian Google N-Grams | 67,137,666,353 | 591,310 | — |

### 2.2. Lexico-syntactic patterns

Sabirova and Lukanin (2014) developed six lexico-syntactic patterns for extracting hypernyms and hyponyms from Russian texts. The patterns were encoded as a cascade of finite state transducers (FSTs) with the help of the corpus processing tool *Unitex*[6]. Our grammar relies on the full version (Nagel, 2002) of the standard Russian morphological dictionary shipped with the tool. We apply these FSTs to mark hypernyms and hyponyms

[3] https://github.com/cental/PatternSim

[4] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

[5] http://panchenko.me/data/dataset-2734.csv,
http://panchenko.me/data/webtopic-corpus-892233.csv.gz

[6] http://www-igm.univ-mlv.fr/~unitex/

with special tags (HYPER and HYPO). For example, the first FST of six, which corresponds to pattern "такие/таких/таким HYPER, как HYPO[, HYPO] и/или HYPO" (such HYPER as HYPO[, HYPO], and/or HYPO), will produce the following tagged sentence:

> В Индии зародились такие {[религии]=HYPER} как {[индуизм]=HYPO}, {[буддизм]=HYPO}, {[сикхизм]=HYPO} и {[джайнизм]=HYPO}.
>
> In India such {[religions]=HYPER} as {[Hinduism]=HYPO}, {[Buddhism]=HYPO}, {[Sikhism]=HYPO}, and {[Jainism]=HYPO} were born.

Such tagged sentences are used in order to estimate similarity between words. In this case, the words *religion*, *Hinduism, Buddhism, Sikhism,* and *Jainism* will be considered to be semantically similar (see the next section).

### 2.3. Calculation of semantic similarity

We experimented with different ranking formula and the metric *Efreq-Rnum-Cfreq-Pnum* proved to work best for English and French languages (Panchenko et al., 2012). This metric relies on several factors:

- the number of term co-occurrences within a set of concordances;
- frequencies of related terms;
- the "hubness" of related terms; the similarity with the terms that are related to many other terms is reduced;
- the number of distinct patterns which extracted a relation; relations extracted independently by several patterns are more robust than those extracted only by one pattern.

## 3. The Second System: Right-Context Window on the Google N-Grams

The right-context window distributional model represents each word as a vector in a vector-space built using Google N-grams corpus. Its dimensionality is equal to the number of unique words in the corpus, called contexts (or context words), thus each dimension is associated with exactly one context. In the model each element of a vector of any word contains information about its co-occurrence with a certain context word. Semantic similarity calculation is based on an assumption that two words similarity correlates with the distance between their vectors.

### 3.1. Corpus

The Google N-grams project aims at collecting statistical data of all ever published books, using Google Books corpus[7]. The Russian section of this corpus consists

[7] http://storage.googleapis.com/books/ngrams/books/datasetsv2.html, https://books.google.com

of 591 thousand volumes and contains over 67 billion tokens. Every section includes a subsection per each type of N-grams, for N from 1 to 5. Each subsection presents information about N-grams passed certain occurrence thresholds. The full N-gram list is not publicly available. The information is formatted as such lines: "*NG*, *Y*, *P*, *V*, *C*", where *C* is the number of times N-gram *NG* appeared in the corpus in the year *Y*, while *P* and *V* display numbers of pages and volumes containing the N-gram respectively.

The main advantage of the Google N-gram corpus is its size. In our model, we use data from the year 1900 to the present time for preserving language integrity, which is about 560 thousand volumes and over 64 billion tokens. However, in this corpus, the information is sorted by the first word of N-gram, thus preventing the researchers from conducting an experiment for symmetrical context window in reasonable time, which is a more common approach (Patel et al., 1997). In addition it turned out that due to occurrence thresholds many words do not get enough contexts to represent their meanings.

### 3.2. Calculation of semantic similarity

In the experiment, semantic similarity between two words is modeled by a cosine distance between two corresponding PPMI (Positive Pointwise Mutual Information) (Bullinaria and Levy, 2007) vectors. Component $a_i$ (corresponding to a context word $cw_i$) of a PPMI vector of a word $w$ is calculated as follows:

$$a_i = max(0, log(\frac{P(w, cw_i)}{P(w) \cdot P(cw_i)})) = max(0, log(\frac{count(w, cw_i) \cdot count(.)}{count(w) \cdot count(cw_i)})),$$

where $P(w, cw_i)$ is the probability of the occurrence of $cw_i$ within the distance of five words to the right of $w$ (since right-context window of width 5 was used), $P(w)$ and $P(cw_i)$ are probabilities of words $w$ and $cw_i$, correspondingly; $count(w, cw_i)$, $count(w)$, $count(cw_i)$ are corresponding frequencies and $count(.)$ is the size of the corpus.

## 4. The Third System: Skip-Gram Model on the LibRusEc corpus

*word2vec* is a piece of software developed by Mikolov et al. (2013) for learning vector representations for words and phrases[8]. These representations are learnt as the result of parameter optimization for a probabilistic language model. *word2vec* supports several language models. Here we will briefly describe just the one we used to obtain the best results, namely the *skip-gram model*, that was trained using the negative sampling method. Like the widely used bigram model, the skip-gram model estimates probability for a pair of words to be close to each other in the text. But unlike the bigram model these words do not have to occupy adjacent positions, instead they can be separated by other words.

[8] https://code.google.com/p/word2vec

Assume $P(D = 1|w, c; \theta)$ is the probability of the event that a word $w$ appears in some context $c$. Then $P(D = 0|w, c; \theta) = 1 - P(D = 1|w, c; \theta)$ is the probability of the opposite event. Originally, the set of word's contexts is just the set of words occurring within some predefined distance (window size, *win*) from the target word $w$. However, the model generalizes to other context types; for instance, in (Levy and Goldberg, 2014) syntactically dependent words were used as contexts. We want the model to assign high probability to $(c, w)$ pairs which can appear in texts and low probability to the ones which cannot. So the authors of the skip-gram model defined the following optimization problem:

$$\theta^* = arg\ max \prod_{(c,w)\in corp} P(D = 1|w, c; \theta) \prod_{(c,w)\in rand} (1 - P(D = 1|w, c; \theta))$$

Here *corp* contains $(c, w)$ pairs extracted from corpus and *rand* contains randomly generated $(c, w)$ pairs. The probability is calculated the following way:

$$P(D = 1|w, c; \theta) = (1 + e^{-V_c W_w})^{-1},$$

where $\theta = (V, W)$ are two matrices which columns $V_c$ and $W_w$ contain vectors of context $c$ and the word $w$ of some predefined length (vector dimensionality, *dim*). Thus optimization process gives us context vectors and word vectors. We ignore the former and use the latter to calculate semantic similarity. It was shown that simple algebraic operations with such word vectors can be used to model different semantic relations between corresponding words (Mikolov et al., 2013). For instance, synonyms will have very similar word vectors in the terms of cosine measure.

There exist several implementations of the skip-gram model. We used the original C implementation provided by the authors of the method to build word vectors and the Python implementation which is a part of the *GenSim* library[9] to calculate semantic similarity between words given their vectors.

### 4.1. Corpus

Lib.rus.ec is a large collection of Russian books in machine-readable XML-based format FB2. Each FB2 file contains meta information about a particular book (title, language, author, etc.) and its text. Using this meta information we selected books written in Russian. Texts of these books were saved as a single 149G text file containing 12.9 billion tokens[10].

Along with Lib.rus.ec we tried vectors trained on non-lemmatized and lemmatized versions of Russian Wikipedia (see Table 1) and also a version where each token was a concatenation of the lemma and the POS tag e.g. "*российский#JJ империя#NN*" (russian#JJ empire#NN).

---

[9] http://radimrehurek.com/gensim/

[10] as reported by the Unix command *wc*

### 4.2. Calculation of semantic similarity

**Preprocessing.** Each corpus, used to build word vectors, was preprocessed with a slightly modified script from *word2vec* C distribution. The script converts text to lowercase, inserts space character before punctuation marks (otherwise they are considered a part of the previous word), removes digits, several special characters, etc. We also added some preprocessing, that is specific for Russian (replaced all occurrences of "ё" to "е", for instance).

**Building word vectors.** To build word vectors an appropriate utility from *word2vec* C distribution was executed on the preprocessed corpus. We specified the following parameters:

- *cbow*: train CBOW (context bag of words) or skip-gram model. As our preliminary experiments showed, the skip-gram model always gives better results than CBOW, so we did not use CBOW for our submissions and do not describe it here.
- *dim*: word vectors dimensionality; we tried values from 100 to 1,000.
- *window*: maximum distance between a target word and words counted as its contexts; we tried values from 2 to 30.
- *iter*: number of passes over the whole corpus; to solve optimization problem described earlier, *word2vec* uses stochastic gradient descent—an iterative method which can benefit from processing the same training examples many times.
- *min-cnt:* discard words which appear less than this number of times in the corpus. We specified *min-cnt* = 5.

  All other parameters were not specified, so the default values were used.

**Calculating distance.** To calculate a semantic similarity between words we calculated cosine between the corresponding vectors. To deal with out-of-vocabulary words, i.e. the words which didn't occur in our corpus or occurred less than *min-cnt* times, we tried the following technique denoted as "*oov*" in the results table. If a vector is missed for one or both words from a particular word pair we used a set of vectors of its parts instead. First, we tried to split out-of-vocabulary words by a dash and for each in-vocabulary part added its vector to the set. If such set was still empty we tried to remove prefixes from such a word and if the derived words had vectors, then we added their vectors to the set. For instance, the word "*авиамотосообщение*"—a composite noun meaning flight or automobile connection—was represented with vectors of "*мотосообщение*" (automobile connection) and "*сообщение*" (transport connection). We defined similarity between two sets of vectors as similarity between the most similar vectors from these sets. The following examples illustrate the described technique:

sim(актриса, актер-статист) = sim(актриса, [актер, статист]) =
sim(актриса, актер) = 0.75
sim(автотехника, автомототехника) = sim(автотехника,
[мототехника, техника]) = sim(автотехника, мототехника) = 0.64

sim(actress, dummy-actor) = sim(actress, [dummy, actor]) = sim(actress,
actor) = 0.75
sim(auto-vehicles, auto-motor-vehicles) = sim(auto-vehicles,
[motor-vehicles, vehicles]) =
sim(auto-vehicles, motor-vehicles) = 0.64

## 5. Results and Discussion

We performed evaluation of the systems described above on the shared task on a Russian semantic similarity *RUSSE*. This shared task provided us with four benchmarks:

1. **HJ.** Correlations with human judgements in terms of Spearman's rank correlation. This test set was composed of 333 word pairs.
2. **RT**. The quality of a semantic relation classification in terms of an average precision. This test set was composed of 9,548 word pairs (4,774 unrelated pairs and 4,774 synonyms and hypernyms from the *RuThes Lite* thesaurus[11]).
3. **AE**. The quality of a semantic relation classification in terms of an average precision. This test set was composed of 1,952 word pairs (976 unrelated pairs and 976 cognitive associations from the Russian Associative Thesaurus[12]).
4. **AE2.** The quality of a semantic relation classification in terms of an average precision. This test set was composed of 3,002 word pairs (1,501 unrelated pairs and 1,501 cognitive associations from a web-scale associative experiment[13]).

Table 2 presents the results of the three methods on the shared task. As one can observe, the similarity measure *PatternSim* based on lexico-syntactic patterns yields the best results on the concatenation of Wikipedia and Web corpora. However, the *PatternSim* measures provide one of the lowest results among the three considered approaches in terms of correlations with human judgements (*HJ*). Average precision of this method on synonyms and hypernyms (*RT*) and free associations (*AE2*) is also rather low as compared to top system in our study and other best submission to the RUSSE shared task.

**Table 2.** Comparisons of the the HJ, RT, AE and AE2 datasets

| Method | Corpus | HJ | RT | AE | AE2 |
|---|---|---|---|---|---|
| patternsim | web+wiki | 0.372 | 0.754 | 0.708 | 0.797 |
| patternsim | wiki | 0.322 | 0.755 | 0.724 | 0.784 |
| patternsim | web | 0.322 | 0.745 | 0.696 | 0.775 |
| skipgram-dim100-win10-iter1 | lib | 0.621 | 0.847 | 0.912 | **0.967** |
| **skipgram-dim500-win20-iter1 + oov** | **lib** | **0.677** | **0.905** | 0.907 | **0.965** |
| skipgram-dim300-win20-iter1 | lib (20%) | 0.651 | 0.856 | **0.917** | **0.965** |
| skipgram-dim500-win5-iter3 | lib | 0.654 | **0.903** | 0.912 | **0.965** |
| *skipgram-dim500-win5-iter3* | *wiki_nonlem.* | *0.532* | *0.731* | *0.881* | *0.914* |
| *skipgram-dim500-win5-iter3* | *wiki* | *0.601* | *0.803* | *0.771* | *0.928* |
| *skipgram-sim500-win10-iter3* | *lib* | *0.674* | *0.903* | *0.925* | *0.972* |
| ***skipgram-sim500-win10-iter3 + oov*** | ***lib*** | ***0.699*** | ***0.918*** | ***0.928*** | ***0.975*** |
| right-context-window | ngram | 0.303 | 0.612 | 0.734 | 0.676 |

[11] http://www.labinform.ru/pub/ruthes/index.htm

[12] http://it-claim.ru/asis

[13] http://sociation.org

A more close inspection of the results of the pattern-based measures shows that a low performance is caused by a low recall of this approach. The method yields high precision, but is not able to assess similarity between some word pairs. Indeed, this model was able to assess similarity of 5–30% of word pairs, depending on the dataset. For instance, the method PatternSim on the web+wiki corpus was able to model only 98 of 333 word pairs. Therefore, sparsity of this representation is the main problem of the current version of this system.

According to our experiments, the right-context-window approach showed lowest scores among the three considered systems, despite the fact that Google N-gram corpus is 5 times bigger than the Lib.rus.ec. We think the main reason is the frequency threshold which ngrams must pass to be included in the corpus. We investigated occurrences of several less frequent words in Google N-gram corpus and found that there are too few contexts to build an adequate vector representations for these words. Probably, the threshold should not be constant, but should instead depend on the frequency of a particular word.

Finally, the skip-gram model yielded the best results according to the *RUSSE* evaluation. Even when trained on a non-lemmatized Wikipedia, it gives better results than the other two systems, except for the *RT* metric, where it performs almost the same as *PatternSim*. Training on a lemmatized Wikipedia improves the model even further. Finally, the model trained on non-lemmatized Lib.rus.ec showed even better results as this corpus is 50 times bigger than the Russian Wikipedia. It would be interesting to use a lemmatized and POS-tagged version of Lib.rus.ec but we leave this experiment for the future. Increasing corpus size gives significant improvements which are especially notable on the *RT* metric. In the shared task, our skip-gram system ranks among the top 10 submissions (out of 105 other systems), or in the top 5 participants (out of 19 other participants) according to all metrics[14]. The best skip-gram models for Russian language and scripts required to train and use them are available online[15].

To gain more insights on how *word2vec* meta-parameters influence performance, we evaluated models trained with different parameters and on different corpora. We display the most interesting results in Table 2, the full results table is available online[16]. In table 2 we also include the results which were not submitted because they were obtained after the submission was closed. These results are included for comparison and are displayed in *italics*. Several conclusions can be made from these results.

First of all, some preprocessing of the corpus is necessary, otherwise the results could be 3–12% worse than they could be. Probably this is because *word2vec* treates punctuation marks as a part of a previous word if they are not separated by a white space. The lemmatized version of Wikipedia gives about 10% improvement on *HJ* and *RT* metrics compared to the non-lemmatized version, however on *AE2* metric the improvement is only 3% and on *AE* metric the non-lemmatized version is 7% better. Probably this happens because an association word often agrees with a stimulus word in gender and number, so it is not lemmatized.

---

[14] http://russe.nlpub.ru/results

[15] https://github.com/nlpub/russe-evaluation/tree/master/russe/measures/word2vec

[16] http://goo.gl/xPL7DT

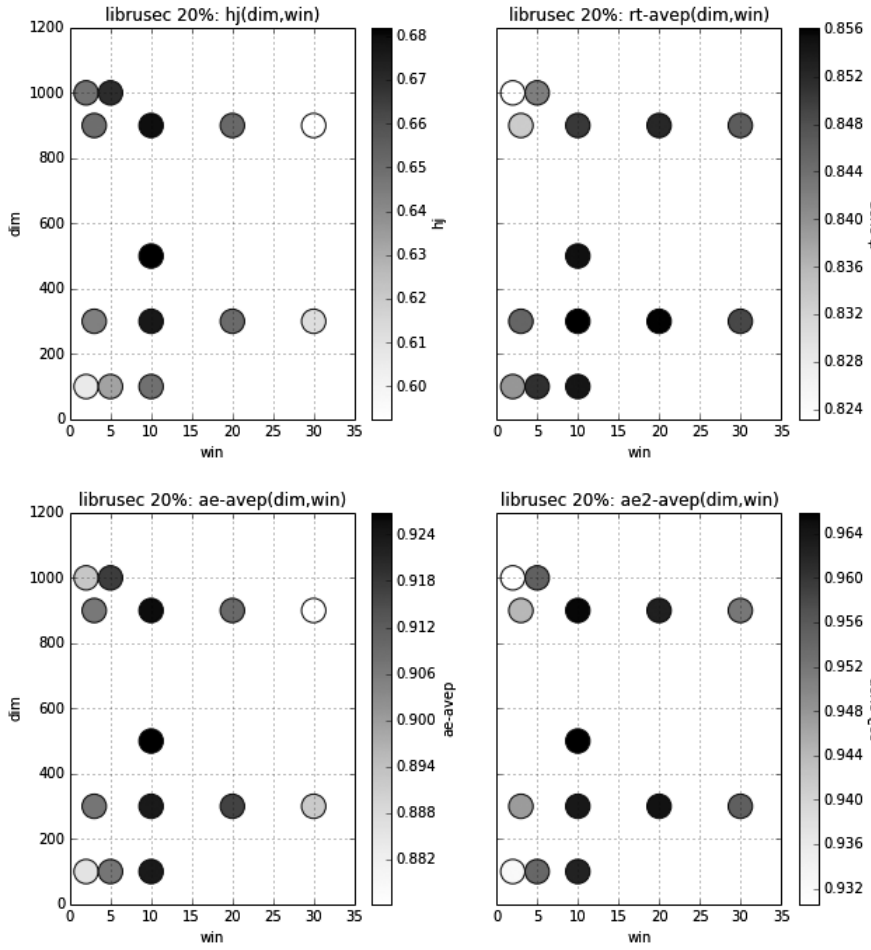


**Fig. 1.** Dependence of *word2vec* vectors' performance from the window size and vectors dimensionality. Vectors were trained on 20% of librusec corpus (30G)

We investigated how word vector dimensionality and context window size affect the results. We did it on 20% of Lib.rus.ec to be able to try many parameter combinations while reducing the computation time. However, it seems that on 100% librusec the results are similar. Fig. 1 clearly shows that performance declines when the window size is less than 5 or more than 20, window size 10 seems to be optimal among the window sizes we tried. The vectors dimensionality does not affect performance as much as the window size, dimensions between 300 and 900 give close results.

Fig. 2 shows how the results depend on the corpus and the number of iterations. As we said before, using even 20% of the non-lemmatized librusec (30G) instead of the lemmatized Wikipedia (3G) gives huge improvements (about 10% on hj and

rt metrics, 20% on *AE* and 3% on *AE2*). However using the whole librusec (150G) gives little improvement on *RT* and *AE2* metrics and even degradation on *HJ* and *AE* metrics compared to 20% of librusec. We have also compared the results on Wikipedia and 2% of librusec, which is almost the same size as Wikipedia (3G). The results on *HJ* and *AE2* are comparable with the lemmatized wiki, on *AE* 2% of librusec give better results which are comparable to the non-lemmatized wiki and on *RT* the lemmatized wiki beats both its non-lemmatized version and 2% of librusec with a huge gap.

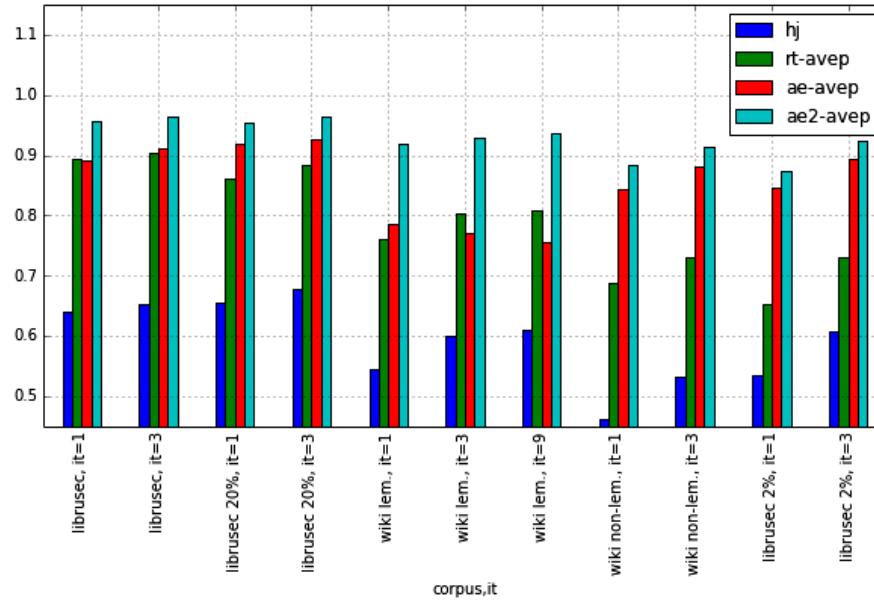


**Fig. 2.** Dependence of *word2vec* vectors' performance from the corpus and number of iterations. All vectors are 500d, window size is 5

We found that increasing the number of iterations over the whole corpus (*iter* parameter) gives great improvements on small corpora, such as Wikipedia, and little, but uniform improvements on large corpora; however the training time increases proportionally to the number of iterations, so it is very expensive to use large values of this parameter on large corpora. Finally, as one can see in Table 2, our technique for dealing with out-of-vocabulary words improves the results a little, but uniformly across all metrics.

## 6. Conclusion and Future Work

Our experiments clearly indicate that it is hard to compete with word vectors that are trained using *word2vec* and such a simple metric as the cosine distance between these vectors. Even when trained on a relatively small Russian Wikipedia this system performs better than the two other systems considered in this paper. When it is trained

on larger corpora and good meta-parameters are selected it ranks in the top 10 submissions (among other 105 submissions), or in the top 5 participants (among 19 other participants) according to all metrics of the *RUSSE* shared task. It worth to notice that these results were reached relatively easy by using freely available implementations of the *word2vec* method and small modifications of the preprocessing scripts to better handle Russian. Most time was spent on the selection of meta-parameters and corpora conversions. We also proposed a simple technique for dealing with compositional out-of-vocabulary words which gave a small but uniform improvement.

We showed that usage of the lemmatized version of Wikipedia instead of the non-lemmatized one gives better performing word vectors according to all metrics except one. We used a non-lemmatized version of Lib.rus.ec and leave experments with its lemmatization for the future. Another promising direction is training *word2vec* on Google N-Grams data which was collected on 5x larger corpora than Lib.rus.ec. However, usage of only Google N-Grams limits the window size to 2 (because only n-grams with n from 1 to 5 are available) which we found to be too small. So it is better to use a combination of Google N-Grams with other corpora. Two problems *word2vec* does not handle are words with multiple meanings and out-of-vocabulary words. These problems should be thoroughly considered in the future.

## Acknowledgements

We thank Digital Society Laboratory LLC provided us with computational platform for most of the experiments described in this paper. Also we would like to acknowledge work of Kristina Sabirova developed the first version of the extraction patterns for *PatternSim* similarity measure.

## References

1. *Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A.* (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In Proceedings of NAACL-HLT 2009, pages 19–27.
2. *Baroni, M., Dinu, G., & Kruszewski, G.* (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 1).
3. *Bullinaria, J., Levy, J.* (2007). Extracting Semantic Representations from Word Co-occurrence Statistics: A Computational Study. In Behavior research methods 39 (3), pages 510–526.
4. *Van de Cruys, T.* (2010). Mining for Meaning: The Extraction of Lexicosemantic Knowledge from Text. PhD thesis, University of Groningen, The Netherlands.
5. *Curran, J. R.* (2004). From distributional to semantic similarity. PhD thesis. University of Edinburgh, UK.
6. *Ferret, O.* (2010). Testing semantic similarity measures for extracting synonyms from a corpus. In Proceeding of LREC.

7. *Krizhanovski A. A.* (2007), Evaluation experiments on related terms search in Wikipedia, SPIIRAS Proceedings, Vol. 5, pp. 113–116.
8. *Krukov K. V., Pankova L. A., Pronina V. S., Sukhoverov V. S., Shiplina L. B.* (2010), Semantic similarity measures in ontology, Control Sciences, Vol. 5, pp. 2–14.
9. *Lee, L.* (1999). Measures of distributional similarity. In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, pages 25–32. Association for Computational Linguistics.
10. *Levy, O., Goldberg, Y.* (2014). Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Vol. 2, pp. 302–308).
11. *Mikolov, T., Chen, K., Corrado, G., Dean, J.* (2013). Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR.
12. *Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.* (2013). Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS.
13. *Nagel, S.* (2002). Formenbildung im Russischen. Formale Beschreibung und Automatisierung für das CISLEX-Wörterbuchsystem.
14. *Panchenko, A., Morozova, O., Naets, H.* (2012). A Semantic Similarity Measure Based on Lexico-Syntactic Patterns. In Conference on Natural Language Processing (KONVENS 2012), — Vienna (Austria), pp. 174–178.
15. *Panchenko, A.* (2013). Similarity measures for semantic relation extraction. PhD thesis. Université catholique de Louvain, 194 pages, Louvain-la-Neuve, Belgium.
16. *Panchenko A., Loukachevitch N. V., Ustalov D., Paperno D., Meyer C. M., Konstantinova N.* (2015) "RUSSE: The First Workshop on Russian Semantic Similarity". In Proceeding of the Dialogue 2015 conference. Moscow, Russia
17. *Patel, M., Bullinaria, J. A., Levy, J. P.* (1997). Extracting Semantic Representations from Large Text Corpora. 4th Neural Computation and Psychology Workshop, London, 9–11 April 1997, 199–212.
18. *Sabirova, K., Lukanin, A.* (2014). Automatic Extraction of Hypernyms and Hyponyms from Russian Texts. In Supplementary Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST 2014) / Ed. by D. I. Ignatov, M. Y. Khachay, A. Panchenko, N. Konstantinova, R. Yavorsky, D. Ustalov. Vol. 1197: Supplementary Proceedings of AIST 2014. CEUR-WS.org, 2014. Pp. 35–40.
19. *Sahlgren, M.* (2006). The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. PhD thesis.
20. *Sokirko A.* (2013), Mining semantically similar language expressions for the Yandex information retrieval system (through to 2012) [Mayning blizkikh po smyslu vyrazheniy dlya poiskovoy sistemy Yandex (do 2012 goda)], available at: http://www.aot.ru/docs/MiningQueryExpan.pdf
21. *Turdakov D. Y.* (2010), Methods and software for term sense disambiguation based on document networks [Metody i programmnye sredstva razresheniya leksicheskoy mnogoznachnosti terminov na osnove setey dokumentov], PhD thesis, Lomonosov Moscow State University, Moscow, Russia.
22. *Zervanou, K., Iosif, E., Potamianos, A.* (2014). Word Semantic Similarity for Morphologically Rich Languages. LREC