Evaluation of Vector Transformations for Russian Word2Vec and FastText Embeddings

Olga Korogodina[1] [0000-0003-3601-4677], Olesya Karpik[2] [0000-0002-0477-1502] and Eduard Klyshinsky[1] [0000-0002-4020-488X]

[1] National Research University Higher School of Economics, Moscow Myasnitskaya. 20, 101000, Russia
eklyshinsky@hse.ru
[2] Keldysh Institute of Applied Mathematics, Miusskaya sq., 4
Moscow, 125047, Russia
parlak@mail.ru

Abstract. Authors of Word2Vec claimed that their technology could solve the word analogy problem using the vector transformation in the introduced vector space. However, the practice demonstrates that it is not always true. In this paper, we investigate several Word2Vec and FastText model trained for the Russian language and find out reasons of such inconsistency. We found out that different types of words are demonstrating different behavior in the semantic space. FastText vectors are tending to find phonological analogies, while Word2Vec vectors are better in finding relations in geographical proper names. However, we found out that just four out of fifteen selected domains are demonstrating accuracy more that 0.8. We also draw a conclusion that in a common case, the task of word analogies could not be solved using a random word pair taken from two investigated categories. Our experiments have demonstrated that in some cases the length of the vectors could differ more than twice. Calculation of an average vector leads to a better solution here since it closer to more vectors.

Keywords: Word Embeddings · Vector Space · Vector Transformation · Word Analogies.

1 Introduction

The basic point for the semantic space of natural language words was the paper [1] published in 2003. It introduced fixed-size vectors (embeddings) generated by a neural network using statistical information about the word context. This concept was developed in [2] where the author demonstrated that such pre-trained vectors can be useful for solution of different problems of natural language processing.

The Word2Vec mode based on the distributive hypothesis was introduced in 2013 [3-5]. The Word2Vec model also uses neural networks reinforced by several new ideas.

First of all, the new approach [4] had less computational complexity compared to the previous systems. The next article [5] increases its learning rate and accuracy. And the greatest contribution of the authors was the publication of source codes and pre-trained language models for free use.

However, the authors of these papers processed texts written only in English. The transfer of these models into inflectional languages meets some problems because of the large variety of tokens of the same lemma. The statistical distribution of word forms of the given lemma is not always uniform. As a result, the model needs bigger corpora to collect good statistics for rare forms. That is why researchers use lemmatization for inflectional languages. However, some of the problems need information for individual word forms, what brings us back to the problem of under-tuned models.

In order to keep the important lexical information, the FastText model was introduced in 2017 [6]. The authors of this model keep the following idea. Both prefixes and postfixes of words carry semantic information as well as word roots. In this case, the meaning of a word can be composed from the meaning of its parts. Dividing a word into n-grams, the system collects more information about the same n-gram using contexts of different words.

The authors of [3-5] claimed that the new semantic space allows the vector arithmetic. Their example "Queen = King–man + woman" swiftly becomes very famous. However, it becomes clear in a short time that such operations do not always lead us to success. One of the proofs of this concept is the problem of words analogies. The early experiments demonstrated that another favorite example, countries and their capitals, does not work correctly for any case – country, capital and pre-trained language model. The accuracy of this analogy was pretty high but not enough to state that vector arithmetic works properly. The Word2Vec and FastText models correctly find the list of semantic neighbors for a given word, what makes it a crucial part of modern systems of natural language processing. However, the problem of words analogies does not work as well as it could be.

In this paper, we investigate the reasons of such deviations in accuracy. In the Section 2, we state the problem of word analogies as a vector transformation problem. The Section 3 gives a short review of existing vector transformation methods for the problem of word analogies. Sections 4 and 5 describe the used data set and the numerical evaluation of free language models for the Russian language. The Section 6 analyses the reasons of low accuracy for some categories of word analogies. The Section 7 concludes the article. In this paper, we do not consider systems of the BERT and ELMO families [7, 8]. The correct investigation of such systems needs a slight correction of our method and will be conducted in the nearest future. The description of contextualized words embedding could be found in paper [9].

2 Formal Statement of the Problem of Word Analogies

In common words, the main question of the problem of word analogies could be stated as "Is there a word c which relates to the word b as the word a' relates to the word a?" Answering this question, Word2Vec uses vector representation of the word. Let $v_{a'}$ and

v_a be vectors corresponding to the words *a*' and *a* respectively; in this case, the vector difference $v_{a'}$ - v_a expresses the semantic relation (or in other words, the semantic difference) between the words *a* and *a*'. Thus, in order to find an analogue, we should find the word *x* and its corresponding vector *y* such that $y - v_b = v_{a'} - v_a$, or

$$y = v_b + v_{a'} - v_a. \tag{1}$$

However, the probability of existence of a word having exactly the same vector as v_x is extremely small. That is why Word2Vec finds vector *y*' that is the closest word to the vector *y*:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ cos(v, v_b + v_{a'} - v_a) \tag{2}$$

We can reformulate the question for word groups. Let us consider a set of word pairs (w_{11}:w_{12}), (w_{21}:w_{22}), …, (w_{N1}:w_{N2}) that have the same semantic or lexical relation, and their corresponding vectors v_{11}, v_{11}, v_{21}, v_{21}, …, v_{N1}, v_{N1}. In this case, the task of word analogies could be formulated as following: if there is a vector *x* that makes an affine transformation of w_{11}, w_{21}, …, w_{N1} to w_{12}, w_{22}, …, w_{N2}, then *x* is such that

$$v_{i2} = \underset{v'}{argmax} cos(v', v_{i1} + x), \tag{3}$$

Let us denote the fact that the word *a* relates to the word *b* in the same sense as the word *c* relates to the word *d* by the following equation: (*a*:*b*) :: (*c*:*d*). For example, (*apple*:*fruit*) :: (*cucumber*:*vegetable*), (*apple*:*apples*) :: (*cucumber*:*cucumbers*), and, classical, (*king*:*queen*) :: (*man*:*woman*). In this case, a request to find an analogy looks like (*king*:*?*) :: (*man*:*woman*) or (*man*:*woman*) :: (*king*:*?*).

3 Review of Affine Transformation Methods for the Problem of Word Analogies

As it was mentioned above, the Word2Vec system uses an algorithm that finds the vector nearest to the calculated one [3]:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ cos(v, v_b + v_{a'} - v_a). \tag{4}$$

Such model is called **3CosAdd**. As it was demonstrated in [10-15], the 3CosAdd method has crucial drawbacks. The accuracy of this method varies depending on the word set or the trained model [10, 11, 15]. This method is not applicable for such tasks as analogies of synonyms and antonyms [11, 12]. Such variety could be explained by the drawbacks of trained models and text corpora; however, the paper [13] states that the reason is the relative ordering of vectors in the model space. Moreover, the 3CosAdd method returns just one vector while in such tasks as “whole-part” there could be a set of response vectors [14].

That is the why researchers have to find other methods to solve the problem of word analogies. One of them is the **Only-b** [16] method that takes a nearest neighbor of v_b:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ cos(v, v_b). \tag{5}$$

We suppose here that if v_a and v_b in (1) are very close, then v_a - v_b = 0 and $y = v_b$.

The next method is **Ignore-a** [16] that takes the vector closest to the sum of $v_{a'}$ and v_b, i.e., finds the vector located between $v_{a'}$ and v_b:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ cos(v, v_{a'} + v_b) \tag{6}$$

The **PairDirection** [10] mehod supposes that $v_{a'}$ - v_a and $v_{b'}$ - v_b are collinear:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ cos(v - v_b, v_{a'} - v_a).$$

The authors of **3CosMul** method show in the paper [10] that formula (2) is equal to

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ \big(cos(v, v_{a'}) - cos(v, v_a) + cos(v, v_b)\big). \tag{7}$$

The authors use an example (*London*:*England*)::(*Baghdad*:*?*). The component $cos(v, v_{England})$ responses to the similarity of the answer with the country, the component $cos(v, v_{Baghdad})$ responses to the cultural and geographical similarity, and the component $cos(v, v_{London})$ responses to the cultural and geographical difference. The correct answer is *Iraq*, but the component $cos(v, v_{Baghdad})$ dominates all other components and the resulting answer becomes *Mosul*. In order to eliminate such domination, the authors introduce a new formula that keeps balance among the various components:

$$y' = \underset{v \notin \{v_a, v_{a'}, v_b\}}{argmax}\ \frac{cos(v, v_{a'})cos(v, v_b)}{cos(v, v_a) + \varepsilon}, \tag{8}$$

where $\varepsilon = 0.001$ is a small constant eliminating division by zero.

The task of word analogies is very sensitive to the noise in the input data. A word can be homonymous; this means that it should be presented as two or more separate vectors representing different meanings of this word. In case of Word2Vec, such a word will be represented only by a vector that will be a superposition of all its meanings. Moreover, the resulting vectors of similar entities could express differences in their occurrence with other words. For example, a dog and a cow are both animals, but dog is a carnivore and a human's friend, while a cow is an herbivore and gives milk; thus, the analogy is not complete here. In order to eliminate such influence, the authors of the **3CosAvg** method [17] introduce a new formula that takes into account not just a pair of words but whole two groups having the same analogy:

$$y' = \underset{v \in V \setminus \{v_b\}}{argmaxcos}\left(v, v_b + \frac{\sum_{i=1}^{n} v_{a'_i}}{n} - \frac{\sum_{i=1}^{n} v_{a_i}}{n}\right) \tag{9}$$

As it was shown in [10, 16], methods Only-b, Ignore-a, and PairDirection give unsatisfactory results. In the next part of our article we will compare the results of two methods (3CosAdd and 3CosAvg) shown on different models trained in Russian.

4 Used Data Sets

We used several pre-trained models from the site RusVectōrēs (http://rusvectores.org/ru/models/): Araneum Upos Skipgram 2018, Ruwikiruscorpora Upos Skipgram 2018, Ruwikiruscorpora Upos Skipgram 2019, Tayga Upos Skipgram 2019, News Upos Skipgram 2019, Ruscorpora Upos CBOW 2019, Araneum Fasttext Skipgram 2018, Facebook FastText CBOW 2018 [29, 30]. The first models were trained using Word2Vec, the two later ones were trained using FastText.

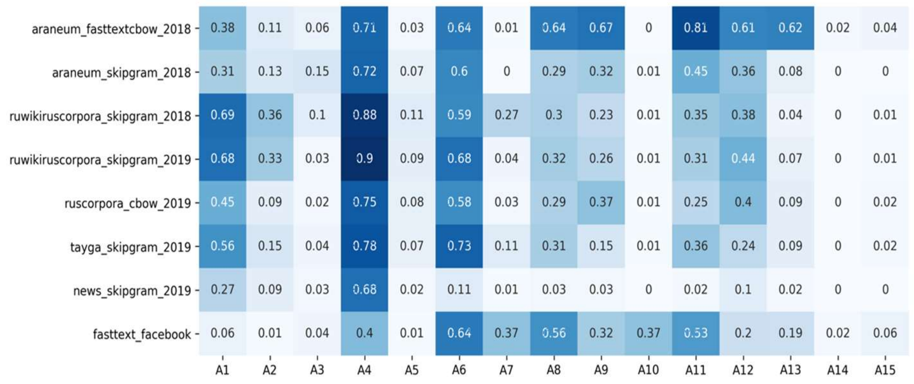
For semantic analogies, we used the Russian versions of Google analogy test set [4] and BATS (The Bigger Analogy Test Set) [15]. For grammatical analogies we used morphological dictionary of the Russian language. The list of used categories is presented in Table 1.

Table 1. Used semantic and grammatical categories

Category	ID	Example		Number of Pairs
Famous capital → Country	A1	Афины	Греция	23
All capitals → Country	A2	Канберра	Австралия	115
Country → currency	A3	Ангола	кванза	30
Country → Adjective	A4	Австралия	австралийский	41
Country → Language	A5	Аргентина	испанский	36
Masculine → Feminine	A6	наследник	наследница	67
Singular → Plural	A7	улыбка	улыбки	100
Antonyms with *не-* (*non-*, *ir-*)	A8	определенный	неопределенный	27
Adjective → Adverb	A9	спокойный	спокойно	30
Possessive Adjective → Comparative Adjective	A10	яркий	ярче	24
Verb → Corresponding Noun with *-ация* (*-ation*)	A11	консультировать	консультация	55
Verb → Corresponding Noun with *-ение* (*-ment*, *-ion*)	A12	назначать	назначение	55
Verb → Corresponding Noun with *-тель* (*-er*, *-or*)	A13	слушать	слушатель	56
Verb → Reflexive Verb	A14	откопаю	откопаюсь	400
Verb → Verb with *при-*	A15	вязать	привязать	376

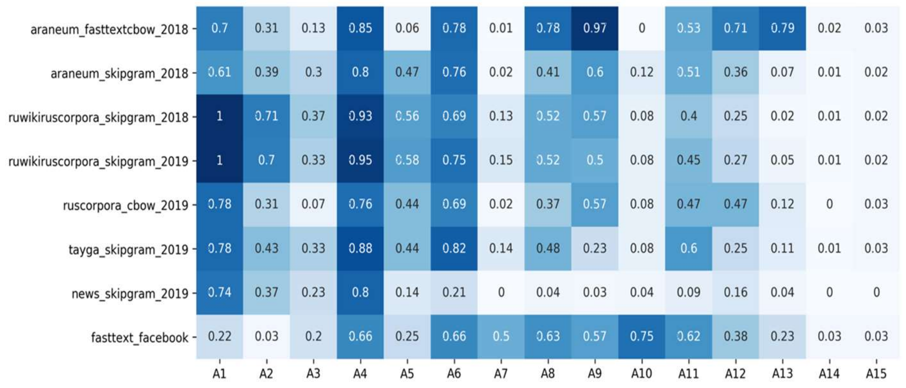
5 Evaluation

For the purpose of evaluation, we calculated the accuracy metrics for all category examples for the given language model. Fig.1 demonstrates the results for the 3CosAdd method; Fig.2 demonstrates the results for the 3CosAvg method. Dark blue shows results with higher accuracy, up to 1; light blue shows results closer to zero.



	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
araneum_fasttextcbow_2018	0.38	0.11	0.06	0.71	0.03	0.64	0.01	0.64	0.67	0	0.81	0.61	0.62	0.02	0.04
araneum_skipgram_2018	0.31	0.13	0.15	0.72	0.07	0.6	0	0.29	0.32	0.01	0.45	0.36	0.08	0	0
ruwikiruscorpora_skipgram_2018	0.69	0.36	0.1	0.88	0.11	0.59	0.27	0.3	0.23	0.01	0.35	0.38	0.04	0	0.01
ruwikiruscorpora_skipgram_2019	0.68	0.33	0.03	0.9	0.09	0.68	0.04	0.32	0.26	0.01	0.31	0.44	0.07	0	0.01
ruscorpora_cbow_2019	0.45	0.09	0.02	0.75	0.08	0.58	0.03	0.29	0.37	0.01	0.25	0.4	0.09	0	0.02
tayga_skipgram_2019	0.56	0.15	0.04	0.78	0.07	0.73	0.11	0.31	0.15	0.01	0.36	0.24	0.09	0	0.02
news_skipgram_2019	0.27	0.09	0.03	0.68	0.02	0.11	0.01	0.03	0.03	0	0.02	0.1	0.02	0	0
fasttext_facebook	0.06	0.01	0.04	0.4	0.01	0.64	0.37	0.56	0.32	0.37	0.53	0.2	0.19	0.02	0.06

Fig. 1. Accuracy by 3CosAdd method



	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
araneum_fasttextcbow_2018	0.7	0.31	0.13	0.85	0.06	0.78	0.01	0.78	0.97	0	0.53	0.71	0.79	0.02	0.03
araneum_skipgram_2018	0.61	0.39	0.3	0.8	0.47	0.76	0.02	0.41	0.6	0.12	0.51	0.36	0.07	0.01	0.02
ruwikiruscorpora_skipgram_2018	1	0.71	0.37	0.93	0.56	0.69	0.13	0.52	0.57	0.08	0.4	0.25	0.02	0.01	0.02
ruwikiruscorpora_skipgram_2019	1	0.7	0.33	0.95	0.58	0.75	0.15	0.52	0.5	0.08	0.45	0.27	0.05	0.01	0.02
ruscorpora_cbow_2019	0.78	0.31	0.07	0.76	0.44	0.69	0.02	0.37	0.57	0.08	0.47	0.47	0.12	0	0.03
tayga_skipgram_2019	0.78	0.43	0.33	0.88	0.44	0.82	0.14	0.48	0.23	0.08	0.6	0.25	0.11	0.01	0.03
news_skipgram_2019	0.74	0.37	0.23	0.8	0.14	0.21	0	0.04	0.03	0.04	0.09	0.16	0.04	0	0
fasttext_facebook	0.22	0.03	0.2	0.66	0.25	0.66	0.5	0.63	0.57	0.75	0.62	0.38	0.23	0.03	0.03

Fig. 2. Accuracy by 3CosAvg method

The best results for the 3CosAdd method were 0.9 for the category Country → Adjective taught at the Russian Wikipedia. Note that this category demonstrates the best results for any used language model. It is not surprising that Country categories achieve better results on Wikipedia text model since there is more information on this topic.

The worst results are demonstrated at reflective verbs (A14) and verbs with prefixes (A15). In our experiments, we included not only the initial forms of the verb but a variety of forms: the past tense, the imperative mood, etc. That is why the resulting affine transfer vector expresses variety of characteristics, but not the only one, and failed to find one preferential direction.

Some words were rare in the learning corpus, and the model failed to represent a solid vector representation for these words. That is true, for example, for the category A10 Possessive Adjective → Comparative Adjective.

The FastText model works better for categories with flexies: A8, A9, A11, A12, A13. Originally, the FastText model was created to work with character n-grams and learn grammatical features of the language. So, FastText is better situated to learn the 'sense' of prefixes and affixes.

The results for 3CosAvg are much better. The category A1 Famous capital → Country demonstrates 100% accuracy on Wikipedia data. However, the results for some categories are not so impressive. The categories A14 and A15 still demonstrate the same results by the same reasons. The category A3 Country → currency cannot be solved better than 37%. We can suppose that the reason is that different countries have the same name for their currency, but their vectors are different; thus, their transition vectors should be different as well. Finally, the category A7 Singular → Plural faces to the variety of Russian homonymous postfixes denoting several tags.

6 Data Analysis

In order to find out the reasons of success and fail, we conducted a visual analysis of the Word2Vec and FastText vectors. First of all, we projected 300-dimensional vectors into 2D-space using Principal Component Analysis (PCA). Instead of t-SNE and UMAP, PCA does not create areas with non-linear skewing. As a result, parallel vectors keep their parallelism. On the other hand, there is a non-zero probability that two vectors on parallel planes could become parallel on the projection. The later case makes some distortion in the data, but not as critical as the former one.

For our experiments, we used two pre-trained language models: Araneum Upos Skipgram 2018 and Araneum Fasttext Skipgram 2018. They were trained on the same Araneum corpus. We randomly selected five word pairs for several categories presented in Fig. 3-8. All vectors are directed from a blue to a red point.

Fig. 3-8 helps us make the following conclusion. The main reason of low accuracy in the task of word analogies is the bias between vectors. It is easy to see that Word2Vec vectors in Fig. 3 are mostly parallel, excluding slightly bias for *Рим-Италия* (*Rome-Italy*). The length of the vectors is also almost the same. Averaging among the beginning and ending points of vectors helps adjusting these small biases in the Word2Vec model. However, the FastText model is oriented mostly on word parts; that is why its vectors are almost randomly oriented and have no preferential direction. The same is true for Fig. 4. However, in Fig. 6 the situation is opposite. The difference between the starting and finishing points here is the prefix *не-* (*non-, ir-*). The FastText model, which vectors are parallel and have quite the same length, processes such situation better than Word2Vec. Quite the same is true for the category A13 Verb → Noun with *-тель* (*-er*, *-or*) (Fig. 8) where the FastText model achieves much better results.

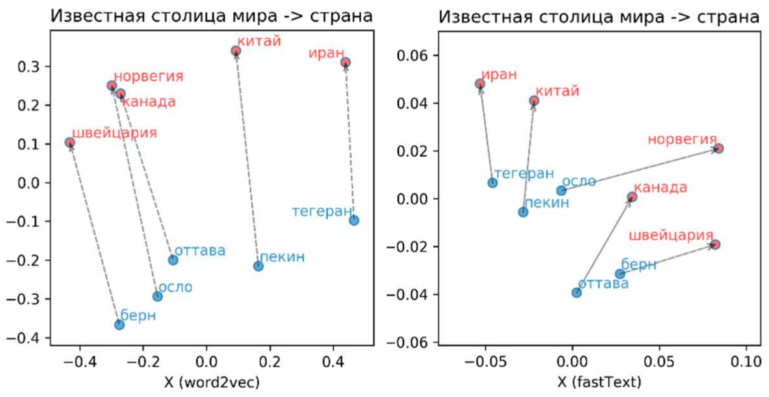


Fig. 3. Word2Vec and FastText vectors for the category A1 Famous capital → Country

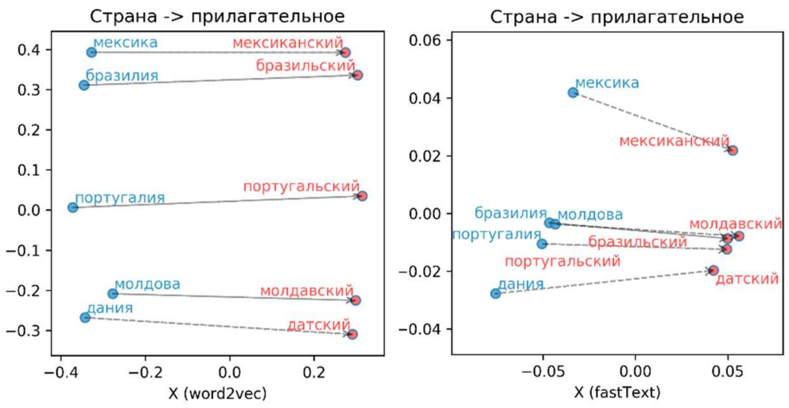


Fig. 4. Word2Vec and FastText vectors for the category A14 Country → Adjective

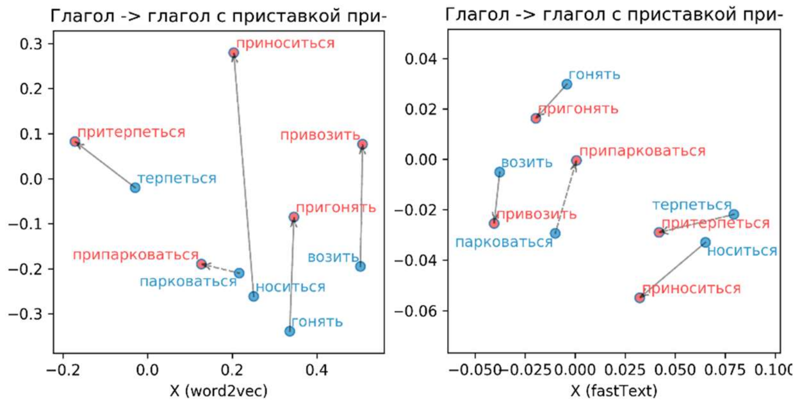


Fig. 5. Word2Vec and FastText vectors for the category A15 Verb → Verb with *при-*

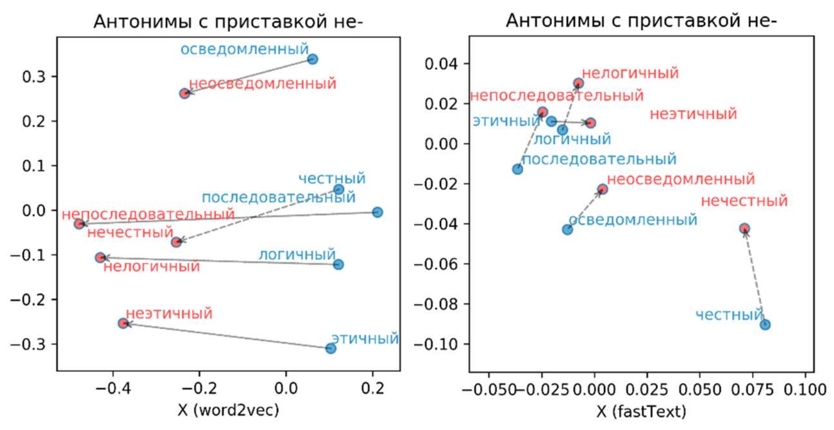


Fig. 6. Word2Vec and FastText vectors for the category A8 Antonyms with *не-* (*non-*, *ir-*)

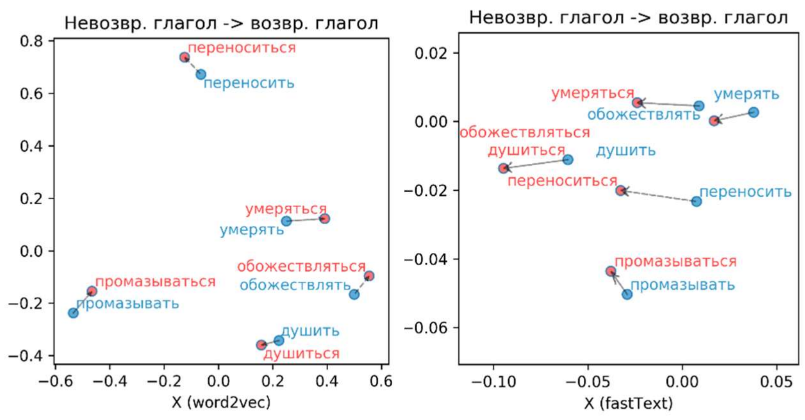


Fig. 7. Word2Vec and FastText vectors for category A14 Verb → Reflexive Verb

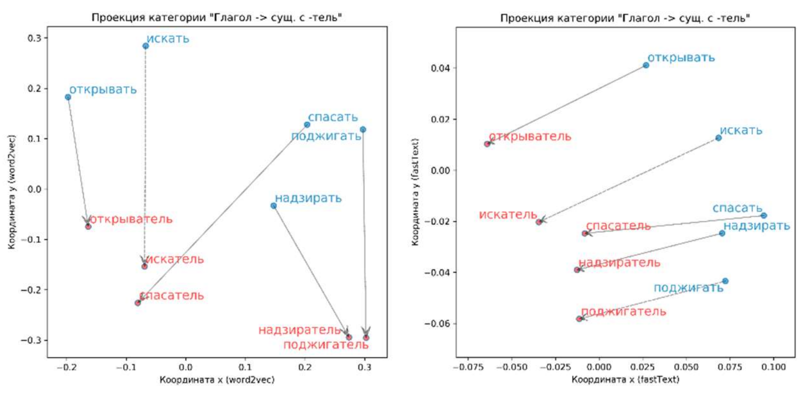


Fig. 8. Word2Vec and FastText vectors for the category A13 Verb → Noun with *-тель* (*-er*, *-or*)

The worst situation is shown in Fig. 5 and Fig. 7 for prefix *при-* and reflexive verbs, which differ from the main form by the suffix *-ся* or *-сь*. These parts of word are very homonymous. So, prefix *при-* could mean approaching, connection, proximity, partiality of an action, finalization of an action, etc. According to these meanings, the corresponding vectors could be perpendicular or antiparallel. This results in very low accuracy for all used models. We used just initial forms for Fig. 5 and Fig. 7 to eliminate the influence of grammatical features.

7 Discussion and Conclusion

In this article, we found several reasons why the vector transformation does not work on some categories of word analogies.

1. A used language model should be taught on texts that have enough occurrences of words for which the task of word analogies is solved. As we can see in Fig. 1 and 2, categories including countries, their capitals, and other related words are better analyzed using corpora like Wikipedia, since such corpora have enough information for inference of logical relations among these words. Moreover, these words are allocated in the same context in the Wikipedia text; thus, their vectors become more similar. Other model, which was trained on fiction or news texts, does not have the same context. That is why categories A1-A5 demonstrate the best results on the Wikipedia corpus. Note that this conclusion needs to be proved in an independent investigation.
2. In a common case, the task of word analogies could not be solved using a random word pair taken from two investigated categories. At least, some words are homonymous, and their vectors are out of the common systems for words of the selected category. If these words are chosen as the basis in such methods as Word2Vec, then such basis will be shifted and word analogies will be found incorrectly. For example, Moscow and Berlin are respectful representatives of their countries in case of international or cultural affairs; these cities are used as synonyms of Russian and German government and culture. However, Bogota and Kampala are rather a capital city than a government. Moreover, Russian prefixes and suffixes could have different meanings. Thus, there will be several preferential directions for different prefix or suffix values. If someone misuses just one word in a pair, he or she will get a wrong transition vector.

   Averaging of the starting and finishing points helps to eliminate this problem. This helps more in the case of factual information (about 30% increment for accuracy), but less for grammatical information (less than 10% for Masculine → Feminine transition and several percent for other categories in the case of Word2Vec, 1-30% depending the category for FastText). Moreover, averaging over extremely homonymous prefixes and suffixes makes the results worse (categories A11, A15 for FastText over the Araneum corpus).

   Averaging over a word list has some drawbacks. One should have a list of words in a given category to average their vectors; this is not always possible. On the other hand, sometime such a list is available, and one could use it to tune vectors in order to predict out-of-list words. But previously he or she should be sure that this list does not contain several preferred semantic directions.

This list could be used separately, but not for calculating pairs only. Any group will have its own center coordinates. If we have a pair *a*:*b*, we can use the coordinates of the nearest centers instead of the vectors of these words.

3. The main idea of an affine transition is that there is one vector that could be added to the word *a* to find its analogy *b*. That means that all vectors for *a*:*b* should be equal, i.e. have approximately equal length and angles of orientation. At least, the value of the bias between this transition vector and the vector of the correct answer should be less than half the distance to the nearest neighbor. As we have found out, it is not always true. In case of homonymous prefixes and suffixes, some vector groups could be oppositely directed. This means that the word analogies task could not be solved using only one transition vector. Our experiments have demonstrated that in some cases the length of the vectors could differ more than twice. Such biases lead to the situation, when the software module has to generate several outputs, and the user should find some extra methods to find the correct answer.

In our future research, we are going to investigate these drawbacks in more detail. One of the solutions here is to construct an interpretable vector space where each axis corresponds to a semantic feature. LSA-based methods successfully mark domains building a hierarchical representation of language semantics. However, these methods are not able to construct a vector space. In our opinion, incorporation of these two methods (LSA and vector space) could help achieve reasonable results. On the one hand, the vector space of Word2Vec-inherited methods allows the vector arithmetic in a continuous semantic space; on the other hand, the application of a hierarchy of terms to such a continuum makes it more interpretable.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, P. A.: Neural Probabilistic Language Model. Journal of Machine Learning Research 3, 1137–1155 (2003).
2. Collobert, R., Weston, J.: A unified architecture for natural language processing. In: Proceedings of the 25th International Conference on Machine Learning, vol. 20, pp. 160–167. (2008).
3. Mikolov, T., Yih, W.-T., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: Proc. of HLT-NAACL, pp. 746-751 (2013).
4. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proc. of International Conference on Learning Representations (ICLR), (2013).
5. Mikolov, T., Chen K., Corrado, G., Dean J.: Distributed Representations of Words and Phrases and their Compositionality. In: Proc. of 27th Annual Conference on Neural Information Processing Systems, pp. 3111-3119. (2013).
6. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics 5, 135-146. (2017).

7. Devlin, J., Chang, M., Lee, K., Toutanova, K., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv:1810.04805, https://arxiv.org/pdf/1810.04805.pdf, last accessed 2020/07/12.
8. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. ArXiv:1802.05365, https://arxiv.org/pdf/1802.05365.pdf, last accessed 2020/07/12.
9. Ethayarajh, K.: How Contextual are Contextualized Word Representations?Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. ArXiv:1909.00512v1, https://arxiv.org/pdf/1909.00512.pdf, last accessed 2020/07/12.
10. Levy, O., Goldberg, Y.: Linguistic Regularities in Sparse and Explicit Word Representations. In: Proc. of 18th Conf. on Computational Natural Language Learning, pp. 171-180. (2014).
11. Köper, M., Scheible, C., Schulte im Walde, S.: Multilingual reliability and "semantic" structure of continuous word spaces. In: Proc. of 11th International Conference on Computational Semantics, pp. 40–45. (2015)
12. Vylomova, E., Rimmel, L., Cohn, T., Baldwin, T.: Take and took, gaggle and goose, book and read: evaluating the utility of vector differences for lexical relation learning. In: Proc. of 54th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1671-1682. (2016).
13. Rogers, A., Drozd, A., Li, B.: The (Too Many) Problems of Analogical Reasoning with Word Vectors. In: Proc. of 6th Joint Conference on Lexical and Computational Semantics, pp. 135–148. (2017).
14. Newman-Griffis, D., Lai, A.M., Fosler-Lussier, E.: Insights into analogy completion from the biomedical domain. BioNLP, 19-28. (2017).
15. Drozd, A., Gladkova, A., Matsuoka, S.: Analogy-based Detection of Morphological and Semantic Relations With Word Embeddings: What Works and What Doesn't. In: Proc. of NAACL Student Research Workshop, pp. 8-15. (2016).
16. Linzen, T.: Issues in evaluating semantic spaces using word analogies. In: Proc. of 1st Workshop on Evaluating Vector-Space Representations for NLP, pp. 13–18. (2016).
17. Drozd, A., Gladkova, A., Matsuoka, S.: Word Embeddings, Analogies, and Machine Learning: Beyond King-Man+Woman=Queen. In: Proc. of COLING 2016, pp. 3519–3530. (2016).
18. Kutuzov, A., Kuzmenko, E.: WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models. In: Analysis of Images, Social Networks and Texts (AIST) 2016, vol. 661, pp. 155-161. (2016).
19. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning Word Vectors for 157 Languages. In: Proc of LREC'2018, pp. 3483-3487. (2018).