Transactions of Karelian Research Centre RAS
No. 7. 2018. P. 149–163
DOI: 10.17076/mat829

Труды Карельского научного центра РАН
№ 7. 2018. С. 149–163

УДК 81.32

WSD ALGORITHM BASED ON A NEW METHOD OF VECTOR-WORD CONTEXTS PROXIMITY CALCULATION VIA ε-FILTRATION

A. N. Kirillov, N. B. Krizhanovskaya, A. A. Krizhanovsky

Institute of Applied Mathematical Research of the Karelian Research Centre of the Russian Academy of Sciences

The problem of word sense disambiguation (WSD) is considered in the article. Set of synonyms (synsets) and sentences with these synonyms are taken. It is necessary to automatically select the meaning of the word in the sentence. 1285 sentences were tagged by experts, namely, one of the dictionary meanings was selected by experts for target words. To solve the WSD problem, an algorithm based on a new method of vector-word contexts proximity calculation is proposed. A preliminary ε-filtering of words is performed, both in the sentence and in the set of synonyms, in order to achieve higher accuracy. An extensive program of experiments was carried out. Four algorithms are implemented, including the new algorithm. Experiments have shown that in some cases the new algorithm produces better results. The developed software and the tagged corpus have an open license and are available online. Wiktionary and Wikisource are used. A brief description of this work can be viewed as slides (https://goo.gl/9ak6Gt). A video lecture in Russian about this research is available online (https://youtu.be/-DLmRkepf58).

K e y w o r d s: synonym; synset; corpus linguistics; word2vec; Wikisource; WSD; RusVectores; Wiktionary.

А. Н. Кириллов, Н. Б. Крижановская, А. А. Крижановский.

АЛГОРИТМ РЕШЕНИЯ WSD-ЗАДАЧИ НА ОСНОВЕ НОВОГО СПОСОБА ВЫЧИСЛЕНИЯ БЛИЗОСТИ КОНТЕКСТОВ С УЧЕТОМ ε-ФИЛЬТРАЦИИ СЛОВ

Рассмотрена задача разрешения лексической многозначности (WSD), а именно: по известным наборам синонимов (синсеты) и предложений с этими синонимами требуется автоматически определить, в каком значении использовано слово в предложении. Экспертами были размечены 1285 предложений, выбрано одно из заранее известных значений (синсетов). Для решения WSD-задачи предложен алгоритм, основанный на новом способе вычисления близости контекстов. При этом для более высокой точности выполняется предварительная ε-фильтрация слов, как в предложении, так и в наборе синонимов. Проведена обширная программа экспериментов. Реализовано четыре алгоритма, включая предложенный. Эксперименты показали, что в ряде случаев новый алгоритм показывает лучшие результаты. Разработанное программное обеспечение и размеченный корпус с открытой лицензией доступны онлайн. Использованы синсеты Викисловаря и тексты Викитеки. Краткое описание работы в виде слайдов доступно по ссылке (https://goo.gl/9ak6Gt), видео с докладом также доступно онлайн (https://youtu.be/-DLmRkepf58).

К л ю ч е в ы е с л о в а: синоним; синсет; корпусная лингвистика; word2vec; Викитека; WSD; RusVectores; Викисловарь.

Introduction

The problem of word sense disambiguation (WSD) is a real challenge to computer scientists and linguists. Lexical ambiguity is widespread and is one of the obstructions in natural language processing.

In our previous work "Calculated attributes of synonym sets" [6], we have proposed the geometric approach to mathematical modeling of synonym set (synset) using the word vector representation. Several geometric characteristics of the synset words were suggested (synset interior, synset word rank and centrality). They are used to select the most significant synset words, i.e. the words whose senses are the nearest to the sense of the synset.

The topic related to polysemy, synonyms, filtering and WSD is continued in this article. Let us formulate the mathematical foundations for solving the problems of computational linguistics in this article.

Using the approach proposed in the paper [2], we present the WSD algorithm based on a new context distance (proximity) calculation via ε-filtration. The experiments show the advantages of the proposed distance over the traditional average vectors similarity measure of distance between contexts.

New ε-proximity between finite sets

It is quite evident that the context distance choice is one of the crucial factors influencing WSD algorithms. Here, in order to classify discrete structures, namely contexts, we propose a new approach to context proximity based on Hausdorff metric and symmetric difference of sets: $A \triangle B = (A \cup B) \setminus (A \cap B)$.

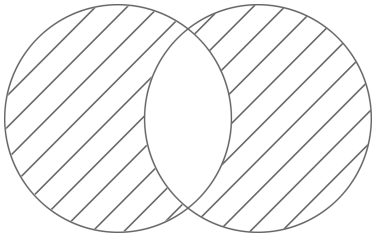

Fig. 1. The set $A \triangle B$ is the shaded part of circles

Recall the notion of Hausdorff metric. Consider a metric space (X, ϱ) where X is a set, ϱ is a metric in X. Define the ε-dilatation $A + \varepsilon$ of a set $A \subset X$

$$A + \varepsilon = \cup\{\overline{B}_\varepsilon(x) : x \in A\},$$

where $\overline{B}_\varepsilon(x)$ is a closed ball centered at x with the radius ε.

The Hausdorff distance $\varrho_H(A, B)$ between compact nonempty sets A and B is

$$\varrho_H(A, B) = \min\{\varepsilon > 0 : (A \subset B + \varepsilon) \wedge (B \subset A + \varepsilon)\},$$

where $A + \varepsilon$, $B + \varepsilon$ are the ε-dilatations of A and B. Consider the following sets (Fig. 2):

$$A(\varepsilon) = A \cap (B + \varepsilon), \quad B(\varepsilon) = B \cap (A + \varepsilon).$$

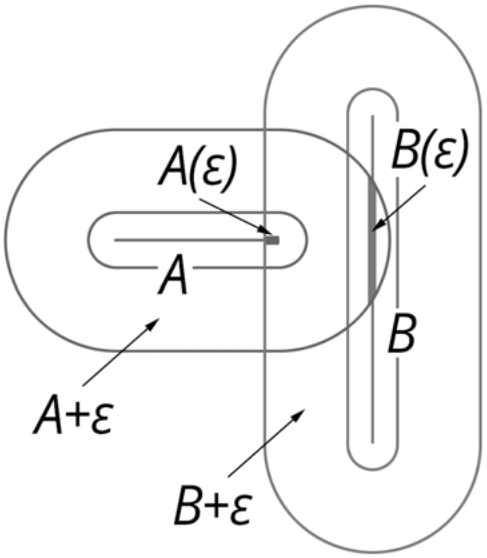

Fig. 2. Two sets $A + \varepsilon$ and $B + \varepsilon$ are the ε-dilatations of segments A and B, and two new proposed set-valued maps $A(\varepsilon)$ and $B(\varepsilon)$ were inspired by Hausdorff distance

Then

$$\varrho_H(A, B) = \min\{\varepsilon > 0 : A(\varepsilon) \cup B(\varepsilon) = A \cup B\}.$$

Consider two contexts $W_1 = \{w_{11}, ..., w_{1m}\}$, $W_2 = \{w_{21}, ..., w_{2n}\}$, where w_{1i}, w_{2j} are words in the contexts, $i = 1, .., m$, $j = 1, ..., n$. Denote by $V_1 = \{v_{11}, ..., v_{1m}\}$, $V_2 = \{v_{21}, ..., v_{2n}\}$ the sets of vectors v_{1i}, v_{2j} corresponding to the words w_{1i}, w_{2j}. Recall that generally in WSD procedures, the distance between words is measured by similarity function, which is a cosine of angle between vectors representing words: $sim(v_1, v_2) = \frac{(v_1, v_2)}{||v_1||||v_2||}$, where (v_1, v_2) is a scalar (inner) product of vectors v_1, v_2, and $||v_i||$ is a norm of vector, $i = 1, 2$. In what follows, $sim(v_1, v_2) \in [-1, 1]$. Thus, the less distance the more similarity. Keeping in mind the latter remark, we introduce the following ε-proximity

of vector contexts $V_1,\ V_2$. Given $\varepsilon \geqslant 0$, construct the sets

$$C(V_1, V_2, \varepsilon) = \{u, v : u \in V_1, v \in V_2, sim(u, v) \geqslant \varepsilon\}.$$

$$D(V_1, V_2, \varepsilon) = (V_1 \cup V_2) \setminus C(V_1, V_2).$$

Supposing that sim plays the role of a metric, then $C(V_1, V_2, \varepsilon)$ is analogous to the expression $A(\varepsilon) \cup B(\varepsilon)$ in the definition of the Hausdorff distance.

Denote by $|Y|$ the power of a set $Y \subset X$, $\mathbb{R}_+ = \{x : x \geqslant 0, x \in \mathbb{R}\}$.

Definition 1. The K-proximity of contexts V_1, V_2 is the function

$$K(V_1, V_2, \varepsilon) = \frac{|C(V_1, V_2, \varepsilon)|}{|V_1 \cup V_2|}.$$

It is clear that $K(V_1, V_2, \varepsilon) \in [0, 1]$. We also define the following function.

Definition 2. The $\tilde{K}$-proximity of contexts V_1, V_2 is the function

$$\tilde{K}(V_1, V_2, \varepsilon) = \frac{|C(V_1, V_2, \varepsilon)|}{1 + |D(V_1, V_2, \varepsilon)|},$$

describing the ratio of "near" and "distant" elements of sets.

The definition implies that $\min \tilde{K}(V_1, V_2, \varepsilon) = 0$, $\max \tilde{K}(V_1, V_2, \varepsilon) = |V_1 \cup V_2|$. The presence of 1 in the denominator permits to avoid zero denominator when $|D(V_1, V_2, \varepsilon)| = 0$.

The ubiquitous distance ϱ between contexts V_1, V_2 is based on the similarity of average vectors: $\varrho(V_1, V_2) = sim(\overline{V}_1, \overline{V}_2)$. But the example (Fig. 3) shows that for two geometrically distant and not too similar structures $\varrho(V_1, V_2) = 1$, that is the similarity ϱ takes the maximum value.

Example

Consider the sets $A = \{a_1, a_2, a_3\}$, $B = \{b_1\}$ pictured in Fig. 3, where $a_1 + a_3 = \overrightarrow{0}$, $a_2 = b_1$. Then, $sim(A, B) = sim(\frac{1}{3}(a_1 + a_2 + a_3), b_1) = sim(a_2, b_1) = 1$, $\tilde{K}(A, B, \varepsilon) = \frac{2}{3}$, $K(A, B, \varepsilon) = \frac{1}{2}$.

The equality of average vectors does not mean the coincidence of A and B, which are rather different (Fig. 3).

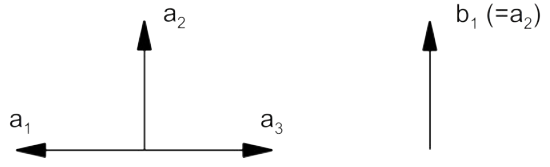

Fig. 3. An example of similar average vectors ($\overline{A} = a_2 = b_1 = \overline{B}$) and totally different sets of vectors: $\{a_1, a_2, a_3\}$ and $\{b_1\}$

Average algorithm with synonyms ε-filtration

Consider a sentence $S_w = (w_1 \dots w_i^* \dots w_n)$ containing a target word w_i^* (denote it as w^*). and a vector representation $S = (v_1 \dots v_i^* \dots v_n)$ of S_w, where w_j is a word, v_j is a vector representation of w_j. Denote v_i^* as v^*. Suppose the target word w^* has l senses. Denote by syn_k^w a synset corresponding to k-th sense, $k = 1, \dots, l$, $syn_k^w = \{w_{k1}, \dots, w_{ki_k}\}$, where w_{kp} are synonyms. Let $syn_k = \{v_{k1}, \dots, v_{ki_k}\}$ be a set of vector representations of synonyms w_{kp}, $p = 1, \dots, i_k$.

In what follows, we introduce a procedure of ε-filtration, the idea of which is borrowed from the paper [2].

The synset filtration is the formation of a so called candidate set which consists of those synonyms whose similarity with the words from a sentence is higher than a similarity threshold ε.

The first average algorithm 1, described below, uses average vectors of words of sentences and average vectors of the candidate set of synonyms in synsets.

This algorithm contains the following lines.

Line 1. Calculate the average vector of words of the sentence S

$$\overline{S} = \frac{1}{n} \sum_{j=1}^{n} v_j$$

Lines 3–6. Given $\varepsilon > 0$, let us construct the filtered set of synonyms for each synset

$$cand_k(\varepsilon) = \{u \in syn_k : u \neq v^*,\ sim(u, v^*) > \varepsilon\}.$$

Denote by $s_k(\varepsilon) = |(cand_k(\varepsilon))|$ the power of a set $cand_k(\varepsilon)$.

Line 7. Calculate for $s_k(\varepsilon) > 0$ the average vector of the synset candidates

$$\overline{syn}_k(\varepsilon) = \frac{1}{s_k(\varepsilon)} \sum_{u \in cand_k(\varepsilon)} u.$$

If $s_k(\varepsilon) = 0$, then let $\overline{syn}_k(\varepsilon)$ be equal to the zero vector.

Line 8. Calculate the similarity of the average vectors of the sentence and the k-th filtered synset

$$sim_k(\varepsilon) = sim(\overline{syn}_k(\varepsilon), \overline{S}).$$

Line 10–11. Suppose $max_{k=1,\dots,l}\{sim_k(\varepsilon)\} = sim_{k^*}(\varepsilon)$, i.e. $k^* \in \{1, \dots, l\}$ is the number of the largest $sim_k(\varepsilon)$. If k^* is not unique, then take

Algorithm 1: Average algorithm with synonyms ε-filtration

Data: v^* – vector of the target word w^* with l senses (synsets),
$v_i \in S$, S – sentence with the target word w^*, $v^* \in S$,
$\{syn_k\}$ – synsets of the target word, that is $syn_k \ni v^*$, $k = \overline{1,l}$.
Result: $k^* \in \{1,\dots,l\}$ is the number of the sense of the word w^* in the sentence S.

1 $\overline{S} = \frac{1}{n}\sum_{j=1}^{n} v_j$, the average vector of words of the sentence S
2 **do**
3 take $\varepsilon > 0$
foreach synset of the target word
4 **foreach** $syn_k \ni v^*$ **do**
construct the filtered set $cand_k(\varepsilon)$ of the synset syn_k:
5 $cand_k(\varepsilon) = \{u \in syn_k : u \neq v^*,\ sim(u, v^*) > \varepsilon\}$
6 $s_k(\varepsilon) = |cand_k(\varepsilon)|$, number of candidates of synonyms
the average vector of synset candidates:
7 $$\overline{syn}_k(\varepsilon) = \begin{cases} \frac{1}{s_k(\varepsilon)} \sum\limits_{u \in cand_k(\varepsilon)} u, & \text{if } s_k(\varepsilon) > 0 \\ \overrightarrow{0}, & \text{if } s_k(\varepsilon) = 0 \end{cases}$$
the similarity of average vectors of the sentence and the k-th filtered synset:
8 $sim_k(\varepsilon) = sim(\overline{syn}_k(\varepsilon), \overline{S})$
9 **end**
10 $sim_{k^*}(\varepsilon) = \max\limits_{k=1,\dots,l}\{sim_k(\varepsilon)\} \Rightarrow k^* \in \{1,\dots,l\}$, k^* is the number of the largest $sim_k(\varepsilon)$
11 **while** k^* *is not unique*

another $\varepsilon > 0$ and repeat the procedure from line 3.

Result: the target word w^* has the sense corresponding to the k^*-th synset $syn^w_{k^*}$.

Remark: in the case $\varepsilon = 0$, we denote this algorithm as $\overline{A}_0$-algorithm. In this case, the traditional averaging of similarity is used.

Note. $\overline{A}_0$-algorithm was used in our experiments, it was implemented in Python.[1]

$\overline{A}_0$-algorithm example

A simple example and figures 4–6 will help to understand how this $\overline{A}_0$-algorithm works.

Take some dictionary word w_2 with several senses and several synonym sets (for example, syn_1 and syn_2) and the sentence S with this word (Fig. 4). The task is to select a meaning (synset) of w_2 (that is the target word is w_2^*) used in the sentence S via the $\overline{A}_0$-algorithm.

Let us match the input data and the symbols used in the $\overline{A}_0$-algorithm. The word "служить" (sluzhit') corresponds to the vector v_2.

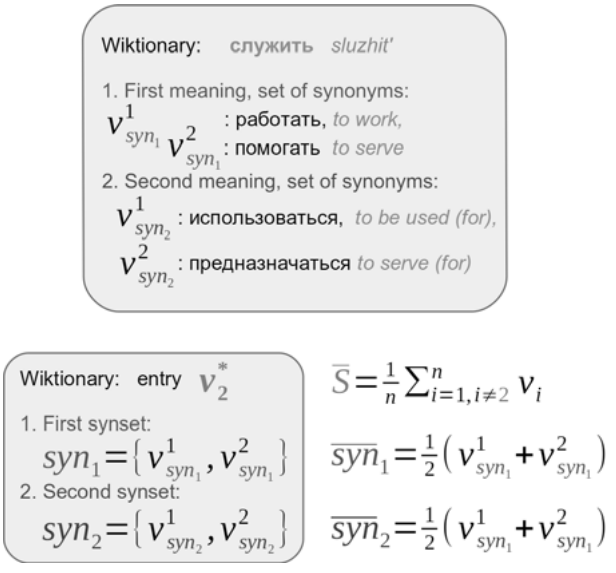

Fig. 4. Digest of the Wiktionary entry "служить" (sluzhit') and mean vectors $\overline{syn_1}$ and $\overline{syn_2}$ of the synonyms sets syn_1, syn_2 and the sentence S with this word w_2^*

[1] See the function *selectSynsetForSentenceByAverageSimilarity* in the file `https://github.com/componavt/wcorpus.py/blob/master/src/test_synset_for_sentence/lib_sfors/synset_selector.py`

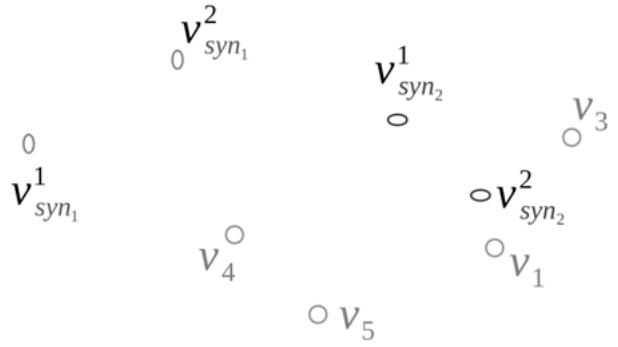

Fig. 5. Sample source data are (1) vertices $v_1 ... v_5$ corresponding to words of the sentence S, the vertex v_2 was excluded since it corresponds to the target word w_2^*, and (2) the target word w_2^* with two synsets syn_1 and syn_2 (Fig. 4), (3) vertices (vectors correspond to words) of the first synset are $\{v^1_{syn_1}, v^2_{syn_1}\}$ and the second synset – $\{v^1_{syn_2}, v^2_{syn_2}\}$

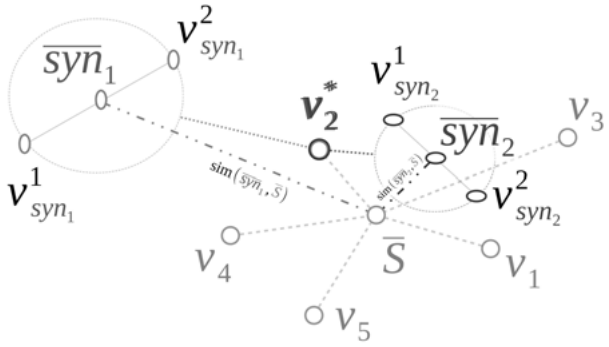

Fig. 6. Similarity between the mean value of vectors of the sentence and the first synonym set is lower than the similarity with the second synset, that is $sim(\overline{syn}_1, \overline{S}) < sim(\overline{syn}_2, \overline{S})$. Thus, the second sense of the target word w_2^* (the second synset syn_2) will be selected in the sentence S by $\overline{A}_0$-algorithm

There is a dictionary article about this word in the Wiktionary, see Fig. 4 (a parsed database of Wiktionary is used in our projects).[2]

Two synonym sets of this Wiktionary entry are denoted by syn_1 and syn_2.

Mean values of the vectors corresponding to synonyms in these synsets will be denoted as $\overline{syn}_1$ and $\overline{syn}_2$, and $\overline{S}$ is the mean vector of all vectors corresponding to words in the sentence S containing the word "служить" (sluzhit').

Average algorithm with sentence and synonyms ε-filtration ($\overline{A}_\varepsilon$)

This algorithm 2 is a modification of algorithm 1. The filtration of a sentence is added to synset filtration. Namely, we select a word from the sentence for which the similarity with at least one synonym from the synset is higher than the similarity threshold ε. Then, we average the set of selected words forming the set of candidates from the sentence. Let us explain algorithm 2 line by line.

Lines 2–5. Given $\varepsilon > 0$, let us construct the set of words of the sentence S filtered by synonyms of the k-th synset syn_k

$$cand_k S(\varepsilon) = \{v \in S : \exists u \in syn_k, sim(v,u) > \varepsilon, \quad v \neq v^*, u \neq v^*\}$$

Denote by $S_k(\varepsilon) = |cand_k S(\varepsilon)|$ the power of the set $cand_k S(\varepsilon)$.

Line 6. Calculate the average vector of words of the filtered sentence

$$\overline{cand_k S}(\varepsilon) = \frac{1}{S_k(\varepsilon)} \sum_{v \in cand_k S(\varepsilon)} v$$

If $S_k(\varepsilon) = 0$, then let $\overline{cand_k S}(\varepsilon)$ be equal to the zero vector.

Lines 7–8. Construct filtered sets of synonyms

$$cand\ syn_k(\varepsilon) = \{u \in syn_k : \exists v \in S, sim(u,v) > \varepsilon, \quad u \neq v^*, v \neq v^*\}.$$

Denote by $s_k(\varepsilon) = |cand\ syn_k(\varepsilon)|$ the power of the k-th filtered synonym set.

Line 9. Calculate for $s_k(\varepsilon) > 0$ the average vector of the k-th synset of candidates

$$\overline{cand\ syn_k}(\varepsilon) = \frac{1}{s_k(\varepsilon)} \sum_{u \in cand\ syn_k(\varepsilon)} u.$$

If $s_k(\varepsilon) = 0$, then $\overline{cand\ syn_k}(\varepsilon)$ equals to the zero vector.

Line 10. Calculate the similarity of the average vectors of the filtered sentence and the k-th filtered synset

$$sim_k(\varepsilon) = sim(\overline{cand_k S}(\varepsilon), \overline{cand\ syn_k}(\varepsilon)).$$

Lines 12–13. Suppose $max_{k=1,...,l}\{sim_k(\varepsilon)\} = sim_{k^*}(\varepsilon)$, i.e. $k^* \in \{1, ..., l\}$ is the number of the largest $sim_k(\varepsilon)$. If k^* is not unique then take another $\varepsilon > 0$ and repeat the procedure from line 2.

Result: the target word w^* in the sentence S has the sense corresponding to the k^*-th synset $syn^w_{k^*}$.

This algorithm was implemented in Python.[3]

[2] See section "Web of tools and resources" on page 156.

[3] See the function *selectSynsetForSentenceByAverageSimilarityModified* in the file `https://github.com/componavt/wcorpus.py/blob/master/src/test_synset_for_sentence/lib_sfors/synset_selector.py`

Algorithm 2: Average algorithm with sentence and synonyms ε-filtration ($\overline{A}_\varepsilon$)

Data: v^* – vector of the target word w^* with l senses (synsets),
$v_i \in S$, S – sentence with the target word w^*, $v^* \in S$,
$\{syn_k\}$ – synsets of the target word, that is $syn_k \ni v^*$, $k = \overline{1,l}$.
Result: $k^* \in \{1,\dots,l\}$ is the number of the sense of the word w^* in the sentence S.

1 **do**
2 take $\varepsilon > 0$
foreach synset of the target word
3 **foreach** $syn_k \ni v^*$ **do**
construct the set of words of the sentence S filtered by synonyms of the k-th synset syn_k:
4 $cand_k S(\varepsilon) = \{v \in S : \exists u \in syn_k, sim(v,u) > \varepsilon, v \neq v^*, u \neq v^*\}$
5 $S_k(\varepsilon) = |cand_k S(\varepsilon)|$, number of candidates of the sentence;
the average vector of sentence candidates:
6 $$\overline{cand_k S}(\varepsilon) = \begin{cases} \frac{1}{S_k(\varepsilon)} \sum\limits_{v \in cand_k S(\varepsilon)} v, & \text{if } S_k(\varepsilon) > 0 \\ \overrightarrow{0}, & \text{if } S_k(\varepsilon) = 0 \end{cases}$$
ε-filtration of the synset syn_k by the sentence S:
7 $cand\ syn_k(\varepsilon) = \{u \in syn_k : \exists v \in S, sim(u,v) > \varepsilon, u \neq v^*, v \neq v^*\}$
8 $s_k(\varepsilon) = |cand\ syn_k(\varepsilon)|$, number of candidates of synonyms
the average vector of synset candidates:
9 $$\overline{cand\ syn_k}(\varepsilon) = \begin{cases} \frac{1}{s_k(\varepsilon)} \sum\limits_{u \in cand\ syn_k(\varepsilon)} u, & \text{if } s_k(\varepsilon) > 0 \\ \overrightarrow{0}, & \text{if } s_k(\varepsilon) = 0 \end{cases}$$
the similarity of the average vectors of the sentence and the k-th filtered synset:
10 $sim_k(\varepsilon) = sim(\overline{cand_k S}(\varepsilon), \overline{cand\ syn_k}(\varepsilon))$
11 **end**
12 $sim_{k^*}(\varepsilon) = \max\limits_{k=1,\dots,l} \{sim_k(\varepsilon)\} \Rightarrow k^* \in \{1,\dots,l\}$, k^* is the number of the largest $sim_k(\varepsilon)$
13 **while** k^* *is not unique*

K-ALGORITHM BASED ON ε-DILATATION

The algorithm 3 (K-algorithm) is based on the function $\tilde{K}(A,B,\varepsilon)$ (see previous section "New ε-proximity between finite sets" on page 150), where $A = syn_k$, that is k-th synset, and $B = S$, where S is a sentence. The algorithm includes the following steps.

Lines 2–4. Given $\varepsilon > 0$, let us construct the $C_k(\varepsilon)$ set of "near" words of the k-th synset and the sentence S.

Line 5. Denote by $D_k(\varepsilon)$ the set of "distant" words

$$D_k(\varepsilon) = (S \cup syn_k) \setminus C_k(\varepsilon).$$

Line 6. Calculate $\tilde{K}_k(\varepsilon)$ as the ratio of "near" and "distant" elements of the sets

$$\tilde{K}_k(\varepsilon) = \frac{|C_k(\varepsilon)|}{1 + |D_k(\varepsilon)|}.$$

Lines 8–9. Suppose $max_{k=1,\dots,l} \tilde{K}_k(\varepsilon) = \tilde{K}_{k^*}(\varepsilon)$. If k^* is not unique, then take another $\varepsilon > 0$ and repeat the procedure from line 2.

Algorithm 3: K-algorithm based on ε-dilatation

Data: v^* – vector of target word w^* with l senses (synsets), $v_i \in S$, $v^* \in S$, $\{syn_k\}$ – synsets of v^*, $k = \overline{1,l}$.
Result: $k^* \in \{1,\dots,l\}$ is the number of the sense of the word w^* in the sentence S.

1 **do**
2 take $\varepsilon > 0$
foreach synset of the target word
3 **foreach** $syn_k \ni v^*$ **do**
set of near words:
4 $C_k(\varepsilon) = \{u, v : u \in syn_k, v \in S, sim(u,v) > \varepsilon\}$
set of distant words:
5 $D_k(\varepsilon) = (S \cup syn_k) \setminus C_k(\varepsilon)$
ratio of "near" and "distant" elements of the sets:
6 $\tilde{K}_k(\varepsilon) = \frac{|C_k(\varepsilon)|}{1+|D_k(\varepsilon)|}$
7 **end**
get the number of the largest ratio k^*
8 $\tilde{K}_{k^*}(\varepsilon) = \max\limits_{k=1,\dots,l} \tilde{K}_k(\varepsilon)$
9 **while** k^* *is not unique*

Result: the target word w^* has the sense corresponding to the k^*-th synset $syn^w_{k^*}$.

An example of constructing C and D sets is presented in Fig. 7 and Table. It uses the same source data as for the $\overline{A}_0$-algorithm, see Fig. 5.

Remark. This algorithm is applicable to the K-function described in the previous section[3] as well. This algorithm was implemented in Python.[4]

More details for this example (Fig. 7) are presented in Table, which shows C and D sets with different ε and values of the $\tilde{K}$-function.

Bold type of word-vertices in Table indicates new vertices. These new vertices are captured by a set of "near" vertices C and these vertices are excluded from the set of "distant" vertices D with each subsequent dilatation extension with each subsequent ε. For example, in the transition from ε_1 to ε_2 the set $D_2(\varepsilon_1)$ loses the vertex v_3. During this transition $\varepsilon_1 \to \varepsilon_2$ the set $C_2(\varepsilon_2)$ gets the same vertex v_3 in comparison with the set $C_2(\varepsilon_1)$.

In Fig. 8, the function $\tilde{K}_1(\varepsilon)$ shows the proximity of the sentence S and the synset syn_1, the function $\tilde{K}_2(\varepsilon)$ – the proximity of S and the synset syn_2. It can be seen in Figure 8 that with decreasing ε, the value of $\tilde{K}_2(\varepsilon)$ grows faster than $\tilde{K}_1(\varepsilon)$.

Therefore, the sentence S is closer to the second synset syn_2. The same result can be seen in the previous Fig. 7.

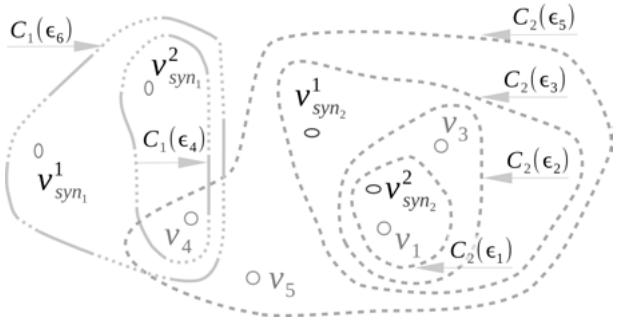

Fig. 7. An example of series of $C_k(\varepsilon)$ (sets of words of k-th synset which are close and near to the sentence S) in the K-algorithm based on ε-dilatation. The growth of the dilation of the vertices of the second synset $\{v^1_{syn_2}, v^2_{syn_2}\}$ captures the vertices of the sentence $S = \{v_1, v_3, v_4, v_5\}$ faster than the dilation of the vertices of the first synset. In other symbols: $(syn_2 + \varepsilon) \cap S \supset (syn_1 + \varepsilon) \cap S$. That is, according to the K-algorithm, the second value of the word-vector v_2, represented by the synset syn_2, will be selected for the sentence S

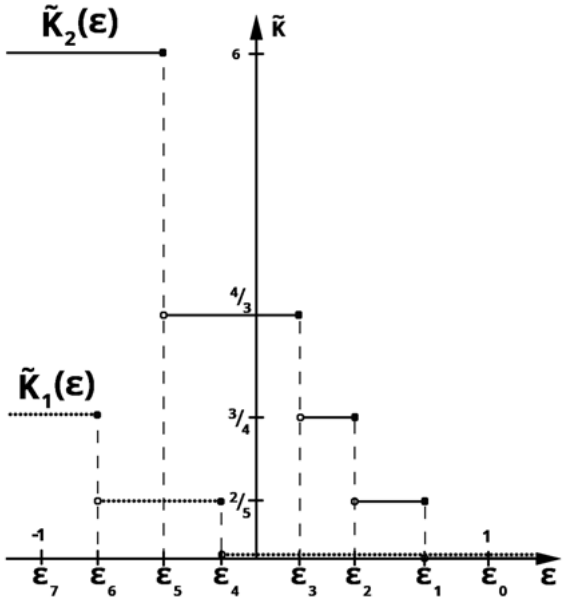

Fig. 8. Left-continuous step functions $\tilde{K}_1(\varepsilon)$, $\tilde{K}_2(\varepsilon)$ show that the sentence S is closer to the second synset syn_2

[4] See the function *selectSynsetForSentenceByAlienDegree* in the file `https://github.com/componavt/wcorpus.py/blob/master/src/test_synset_for_sentence/lib_sfors/synset_selector.py`

An example of the K-algorithm treating the word w_2, which has two synsets syn_1, syn_2 and the sentence S, where $w_2 \in S$, see Fig. 4. The number of the algorithm iteration corresponds to the index of ε. Let the series of ε be ordered so that $1 = \varepsilon_0 > \varepsilon_1 > \varepsilon_2 > ... > \varepsilon_7 = -1$. It is known that $|C_1 \cup D_1 \setminus v_2| = |S \setminus v_2| = 6$, that is the total number of words in the synsets and in the sentence are constants.

ε	$C_2(\varepsilon)$	$D_2(\varepsilon)$	$\lvert C_2\rvert$	$\lvert D_2\rvert$	$\tilde{K}_2(\varepsilon)$
					$\tilde{K}_k(\varepsilon) = \frac{\lvert C_k(\varepsilon)\rvert}{1+\lvert D_k(\varepsilon)\rvert}$
ε_0	$\varnothing$	$\boldsymbol{v_1}$, v_3, v_4, v_5, $v^1_{syn_2}$, $\boldsymbol{v^2_{syn_2}}$	0	6	0.0
ε_1	v_1, $v^2_{syn_2}$	$\boldsymbol{v_3}$, v_4, v_5, $v^1_{syn_2}$	2	4	$\frac{2}{5}$
ε_2	v_1, $v^2_{syn_2}$, $\boldsymbol{v_3}$	v_4, v_5, $\boldsymbol{v^1_{syn_2}}$	3	3	$\frac{3}{4}$
ε_3	v_1, $v^2_{syn_2}$, v_3, $\boldsymbol{v^1_{syn_2}}$	$\boldsymbol{v_4}$, $\boldsymbol{v_5}$	4	2	$\frac{4}{3}$
	$C_1(\varepsilon)$	$D_1(\varepsilon)$	$\lvert C_1\rvert$	$\lvert D_1\rvert$	$\tilde{K}_1(\varepsilon)$
ε_4	$v^2_{syn_1}$, v_4	$\boldsymbol{v^1_{syn_1}}$, v_1, v_3, v_5	2	4	$\frac{2}{5}$
	$C_2(\varepsilon)$	$D_2(\varepsilon)$	$\lvert C_2\rvert$	$\lvert D_2\rvert$	$\tilde{K}_2(\varepsilon)$
ε_5	v_1, $v^2_{syn_2}$, v_3, $v^1_{syn_2}$, $\boldsymbol{v_4}$, $\boldsymbol{v_5}$,	$\varnothing$	6	0	6
	$C_1(\varepsilon)$	$D_1(\varepsilon)$	$\lvert C_1\rvert$	$\lvert D_1\rvert$	$\tilde{K}_1(\varepsilon)$
ε_6	$v^2_{syn_1}$, v_4, $\boldsymbol{v^1_{syn_1}}$	v_1, v_3, v_5	3	3	$\frac{3}{4}$

Experiments

Web of tools and resources

This section describes the resources used in our research, namely: Wikisource, Wiktionary, WCorpus and RusVectores.

The developed WCorpus[5] system includes texts extracted from Wikisource and provides the user with a text corpus analysis tool. This system is based on the Laravel framework (PHP programming language). MySQL database is used.[6]

Wikisource. The texts of Wikipedia have been used as a basis for several contemporary corpora [5]. But there is no mention of using texts from Wikisource in text processing. Wikisource is an open online digital library with texts in many languages. Wikisource sites contains 10 millions of texts[7] in more than 38 languages.[8] Russian Wikisource (the database dump as of February 2017) was used in our research.

Texts parsing. The texts of Wikisource were parsed, analysed and stored to the WCorpus database. Let us describe this process in detail. The database dump containing all texts of Russian Wikisource was taken from "Wikimedia Downloads" site.[9] These Wikisource database files were imported into the local MySQL database titled "Wikisource Database" in Fig. 9, where "WCorpus Parser" is the set of WCorpus PHP-scripts which analyse and parse the texts in the following three steps.

1. First, the title and the text of an article from the Wikisource database are extracted (560 thousands of texts). One text corresponds to one page on Wikisource site. It may be small (for example, several lines of a poem), medium (chapter or short story), or huge size (e.g. the size of the page with the novella "The Eternal Husband" written by Fyodor Dostoyevsky is 500 KB). Text preprocessing includes the following steps:
   - Texts written in English and texts in Russian orthography before 1918 were excluded; about 12 thousands texts were excluded.
   - Service information (wiki markup, references, categories and so on) was removed from the text.
   - Very short texts were excluded. As a result, 377 thousand texts were extracted.
   - Texts splitting into sentences produced 6 millions of sentences.
   - Sentences were split into words (1.5 millions of unique words).

[5] https://github.com/componavt/wcorpus

[6] See WCorpus database scheme: https://github.com/componavt/wcorpus/blob/master/doc/workbench/db_scheme.png

[7] https://stats.wikimedia.org/wikisource/EN/TablesWikipediaZZ.htm

[8] https://stats.wikimedia.org/wikisource/EN/Sitemap.htm

[9] https://dumps.wikimedia.org/backup-index.html

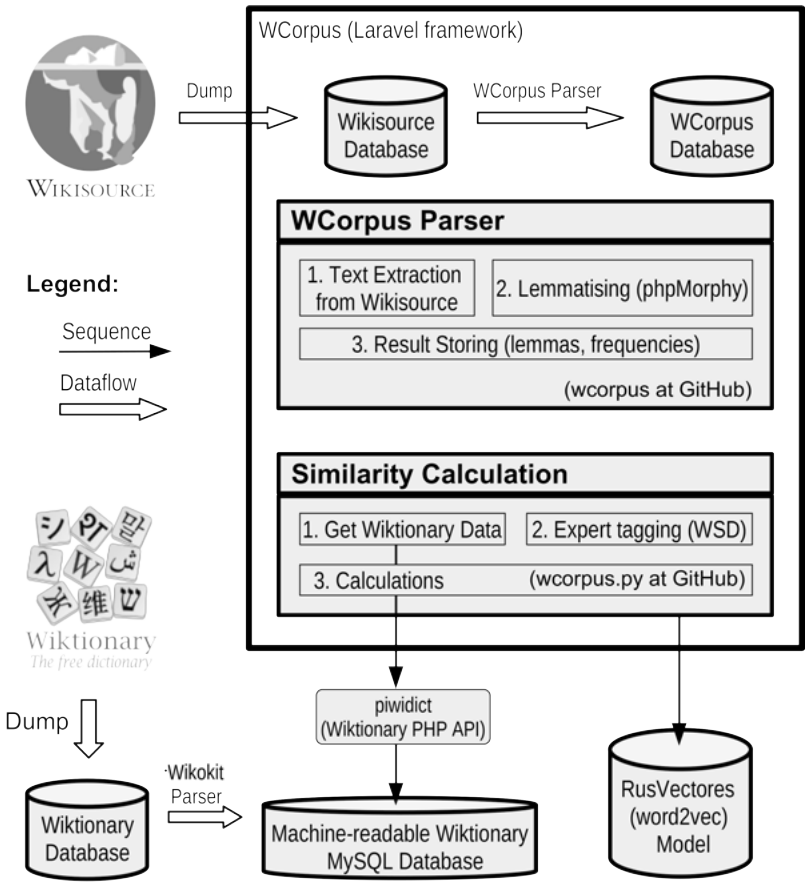

Fig. 9. The architecture of WCorpus system and the use of other resources

3. Secondly, word forms were lemmatized using phpMorphy[10] program (0.9 million lemmas).

4. Lastly, lemmas, wordforms, sentences and relations between words and sentences were stored to WCorpus database (Fig. 9).

In our previous work "Calculated attributes of synonym sets" [6] we also used neural network models of the great project RusVectores[11], which is a kind of a word2vec tool based on Russian texts [9].

Context similarity algorithms evaluation

In order to evaluate the proposed WSD algorithms, several words were selected from a dictionary, then sentences with these words were extracted from the corpus and tagged by experts.

Nine words

Only polysemous words which have at least two meanings with different sets of synonyms are suitable for our evaluation of WSD algorithms.

The following criteria for the selection of synonyms and sets of synonyms from Russian Wiktionary were used:

1. Only single-word synonyms are extracted from Wiktionary. This is due to the fact that the RusVectores neural network model "ruscorpora_2017_1_600_2" used in our research does not support multiword expressions.

2. If a word has meanings with equal sets of synonyms, then these sets were skipped because it is not possible to discern different meanings of the word using only these synonyms without additional information.

[10] `https://packagist.org/packages/componavt/phpmorphy`
[11] `http://rusvectores.org/en/`
[12] `http://whinger.krc.karelia.ru/soft/wikokit/index.html`
[13] `https://github.com/componavt/piwidict`
[14] See information about the subcorpus in the section "Sentences of three Russian writers" on page 158.

A list of polysemous words was extracted from the parsed Russian Wiktionary[12] using PHP API piwidict[13] (Fig. 9).

Thus, 9 polysemous Russian words (presented in the subcorpus[14]) were selected by experts from this Wiktionary list, namely: “бездна” (bezdna), “бросать” (brosat’), “видный” (vidnyy), “донести” (donesti), “доносить” (donosit’), “занятие” (zanyatiye), “лихой” (likhoy), “отсюда” (otsyuda), “удачно” (udachno). The tenth word “служить” (sluzhit’) was left out of consideration, because there are 1259 of 1308 sentences with this frequent word to be tagged by experts in the future (Fig. 10).

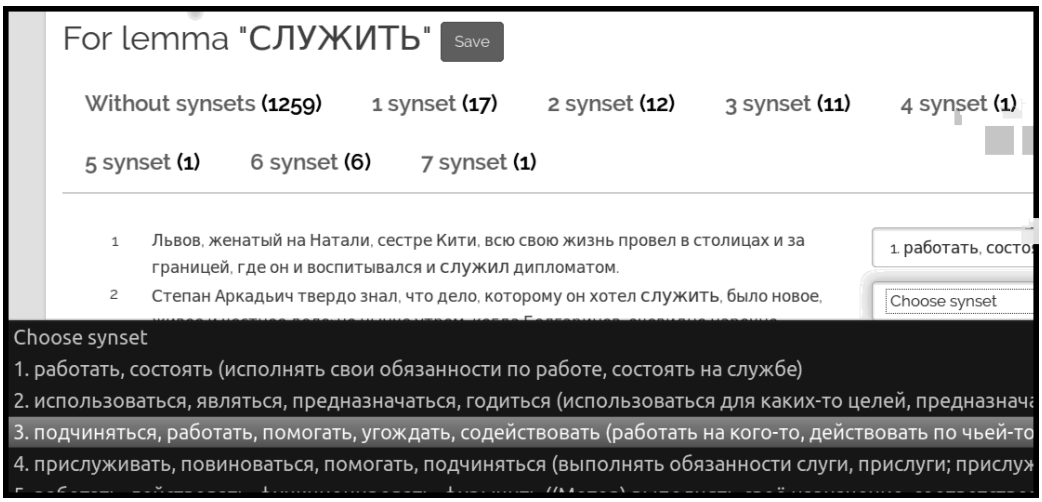

Fig. 10. Russian verb “служить” (sluzhit’) has seven meanings and seven synsets in the developed system WCorpus. 49 sentences are already linked to relevant senses of this verb. 1259 sentences remain to be tagged by experts

Sentences of three Russian writers

The sentences which contain previously defined 9 words were to be selected from the corpus and tagged by experts. But the Wikisource corpus was too huge for this purpose. So, in our research a subcorpus of Wikisource texts was used. These are the texts written by Fyodor Dostoevsky, Leo Tolstoy and Anton Chekhov.

Analysis of the created WCorpus database with texts of three writers shows that the subcorpus contains:[15]

- 2635 texts;
- 333 thousand sentences;
- 215 thousand wordforms;
- 76 thousand lemmas;
- 4.3 million wordform-sentence links;

Texts of this subcorpus contain 1285 sentences with these 9 words, wherein 9 words have in total 42 synsets (senses). It was developed A graphical user interface (webform) of the WCorpus system (Fig. 10) was developed, where experts selected one of the senses of the target word for each of the 1285 sentences.

This subcorpus database with tagged sentences and linked synsets is available online [7].

Text processing and calculations

These 1285 sentences were extracted from the corpus. Sentences were split into tokens. Then wordforms were extracted. All the wordforms were lowercase and lemmatized. Therefore, a sentence is a bag of words. Sentences with only one word were skipped.

The phpMorpy lemmatizer takes a wordform and yields possible lemmas with the corresponding part of speech (POS). Information on POS of a word is needed to work with the RusVectores’ prediction neural network model “ruscorpora_2017_1_600_2”, because to get a vector it is necessary to ask for a word and POS, for example “serve_VERB”. Only nouns, verbs, adjectives and adverbs remain in a sentence bag of words, other words were skipped.

The computer program (Python scripts) which works with the WCorpus database and RusVectores was written and presented in the

[15]See SQL-queries applied to the subcorpus `https://github.com/componavt/wcorpus/wiki/SQL`

[16]`https://github.com/componavt/wcorpus.py`

form of the project *wcorpus.py* at GitHub.[16] The source code in the file *synset_ selector.py*[17] implements three algorithms described in the article, namely:

- $\overline{A}_0$-algorithm implemented in the function *selectSynsetForSentenceByAverageSimilarity()*;
- K-algorithm – function *selectSynsetForSentenceByAlienDegree()*;
- $\overline{A}_\varepsilon$-algorithm – function *selectSynsetForSentenceByAverageSimilarityModified()*.

These three algorithms calculated and selected one of the possible synsets for each of 1285 sentences.

Two algorithms (K and $\overline{A}_\varepsilon$) have an input parameter of ε, therefore, a cycle with a step of 0.01 from 0 to 1 was added, which resulted in 100 iterations for each sentence.

Then, answers generated by the algorithms were compared with the synsets selected by experts.

The number of sentences with the sense correctly tagged by the K-algorithm for nine Russian words presented in Fig. 11.

The legend of this figure lists target words with numbers in brackets (X, Y), where X is the number of sentences with these words, Y is the number of senses.

The curves for the words "ЗАНЯТИЕ" ("ZANYATIYE", solid line with star points) and "ОТСЮДА" ("OTSYUDA", solid line with triangle points) are quite high for some ε, because (1) there are many sentences with these words (352 and 308) in our subcorpus, (2) these words have few meanings (3 and 2).

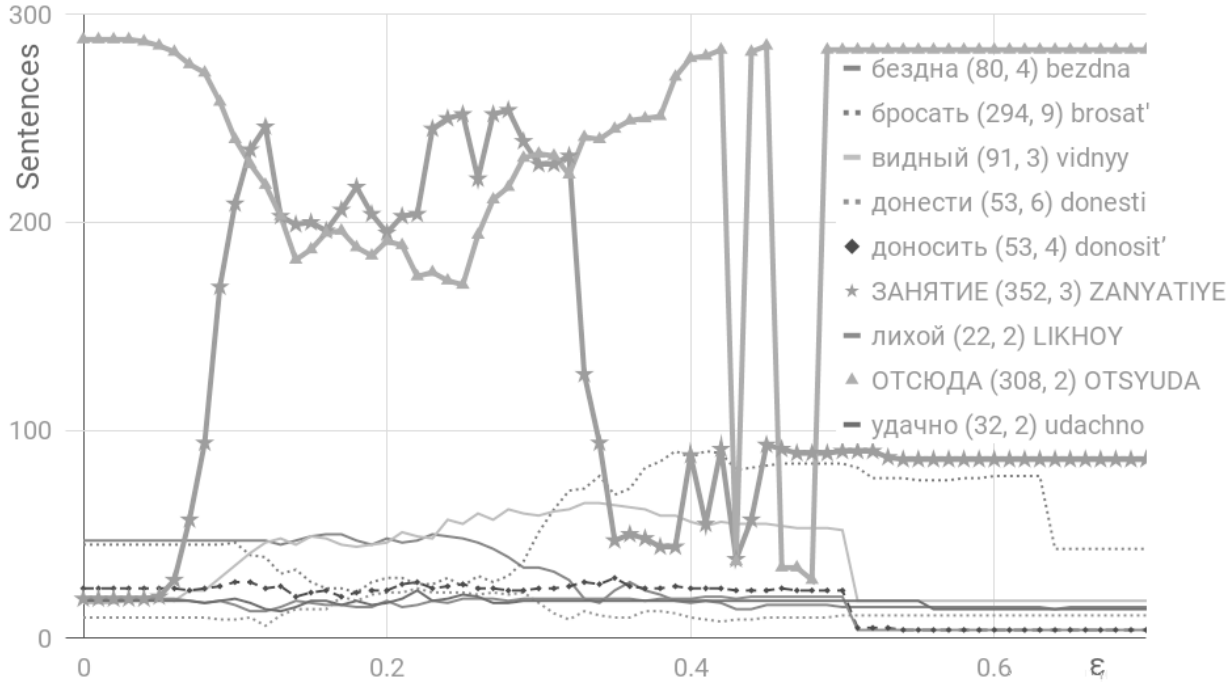

Fig. 11. Number of sentences with the correct tagged sense for nine Russian words by the K-algorithm

[17] https://github.com/componavt/wcorpus.py/blob/master/src/test_synset_for_sentence/lib_sfors/synset_selector.py

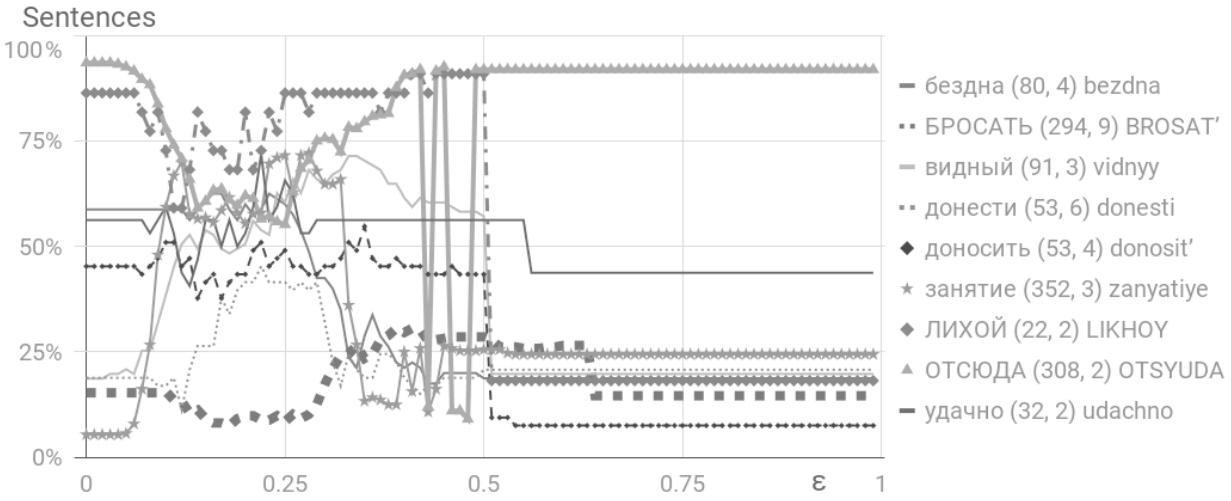

Fig. 12. Normalised data with the fraction of sentences with correctly tagged sense for nine Russian words

More meanings, poorer results.

If a word has more meanings, then the algorithm yields even poorer results. It is visible in the normalised data (Fig. 12), where examples with good results are "ОТСЮДА" (OTSYUDA) and "ЛИХОЙ" (LIKHOY, dash dot line with diamond points) with 2 meanings; the example "БРОСАТЬ" (BROSAT', bold dotted line) with 9 meanings has the worst result (the lowest dotted curve).

Comparison of three algorithms

Let us compare three algorithms by summing the results for all nine words. Fig. 13 contains the following curves: $\overline{A}_0$-algorithm – long dash line; K-algorithm – solid line; $\overline{A}_\varepsilon$-algorithm – dash line.

The $\overline{A}_0$-algorithm does not depend on ε. It showed mediocre results.

The K-algorithm yields better results than $\overline{A}_\varepsilon$-algorithm when $\varepsilon > 0.15$.

The K-algorithm showed the best results on the interval [0.15; 0.35]. Namely, more than 700 sentences (out of 1285 human-tagged sentences) were properly tagged with the K-algorithm on this interval (Fig. 13).

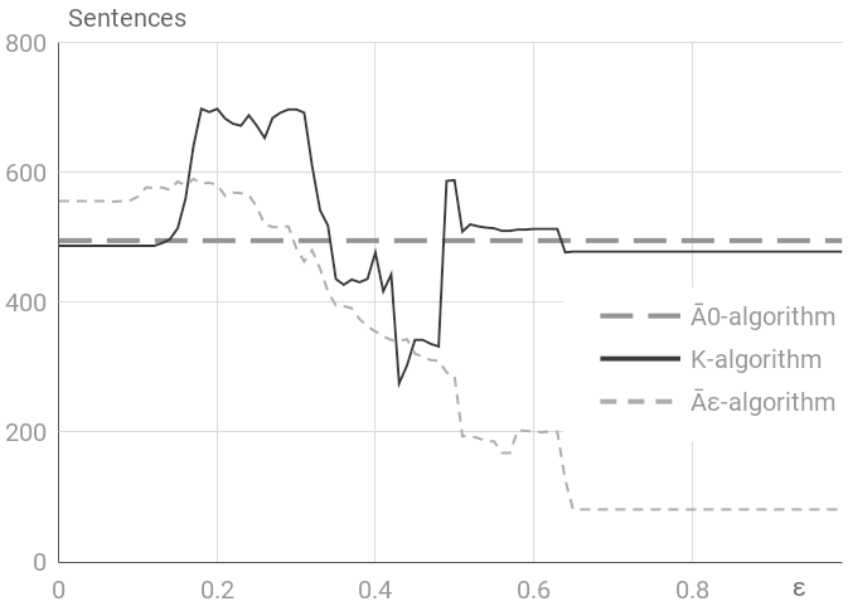

Fig. 13. Comparison of $\overline{A}_0$-algorithm, K-algorithm, $\overline{A}_\varepsilon$-algorithm

Comparison of four algorithms as applied to nine words

Let us compare the results of running four algorithms for each word separately (Fig. 14): $\overline{A}_0$-algorithm – long dash line with triangle points; K-algorithm – solid line with square points; $\overline{A}_\varepsilon$-algorithm – dash line with circle points; "Most frequent meaning" – dashed line with X marks.

The simple "most frequent meaning" algorithm was added to compare the results. This algorithm does not depend on the variable ε, it selects the meaning (synset) that is the most frequent in our corpus of texts. In Fig. 14 this algorithm corresponds to a dashed line with X marks.

The results of the "most frequent meaning" algorithm and $\overline{A}_0$-algorithm are similar (Fig. 14).

The K-algorithm is the absolute champion in this competition, that is for each word there exists an ε such that the K-algorithm outperforms other algorithms (Fig. 14).

Let us explain the calculation of the curves in Fig. 14.

For the $\overline{A}_0$-algorithm and the "most frequent meaning" algorithm, the meaning (synset) is calculated for each of the nine words on the set of 1285 sentences. Thus, $1285 \cdot 2$ calculations were performed.

And again, the $\overline{A}_\varepsilon$-algorithm and the K-algorithm depend on the variable ε. But how can the results be shown without the ε axis? If at least one value of ε gives a positive result, then we suppose that the WSD problem was correctly solved for this sentence by the algorithm.

The value on the Y axis for the selected word (for $\overline{A}_\varepsilon$-algorithm and K-algorithm) is equal to the sum of such correctly determined sentences (with different values of ε) in Fig. 14.

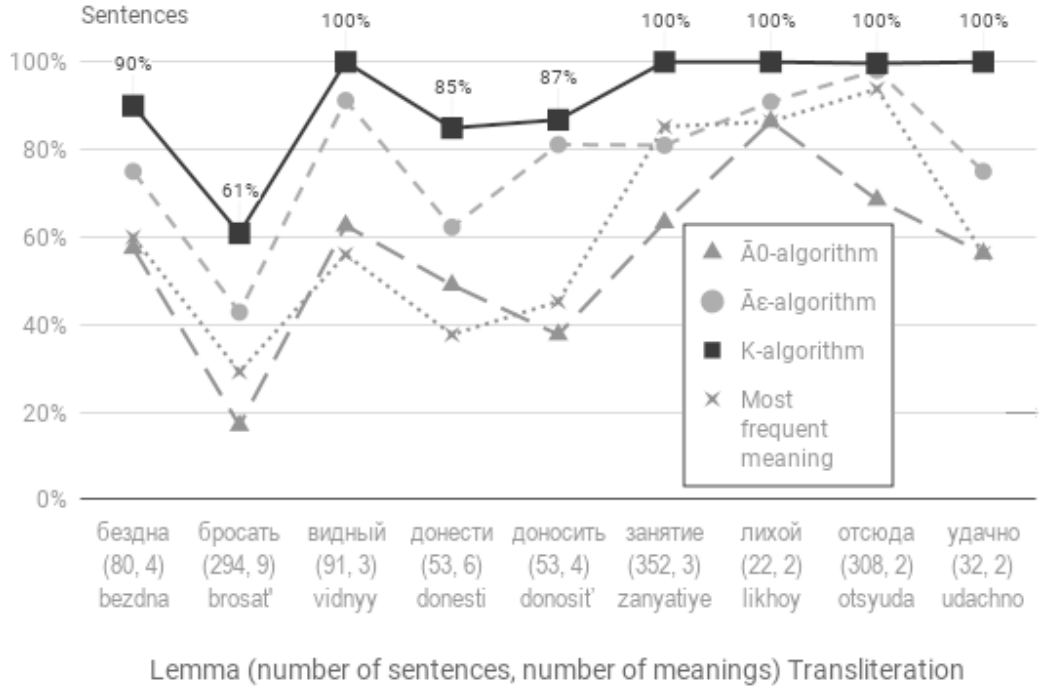

Fig. 14. Comparison of $\overline{A}_0$-algorithm, K-algorithm, $\overline{A}_\varepsilon$-algorithm and the most frequent meaning

Perhaps it would be more correct to fix ε corresponding to the maximum number of correctly determined sentences. Then, the result will not be so optimistic.

To show the complexity of comparing and evaluating ε-algorithms (that is, algorithms that depend on ε), let us try to analyze the results of the K-algorithm, shown in Fig 15.

The percentage (proportion) of correctly determined 1285 sentences for 9 words by the K-algorithm, where the ε variable changes from 0 to 1 in increments of 0.01, is presented in Fig. 15. Thus, $1285 \cdot 100$ calculations were performed.

These proportions are distributed over a set of possible calculated results from 0% (no sentence is guessed) to 100% (all sentences are guessed) for each of nine words.

This Figure 15 does not show which ε-values produce better or poorer results, although it could be seen in Figures 11–13. But the Figure does show the set and the quality of the results obtained with the help of the K-algorithm. For

example, the word “лихой” (likhoy) with 22 sentences and 100 different ε has only 8 different outcomes of the K-algorithm, seven of which lie in the region above 50%, that is, more than eleven sentences are guessed at any ε.

For example, the word “бросать” (brosat’) has the largest number of meanings in our data set, it has 9 synonym sets in our dictionary and 11 meanings in Russian Wiktionary.[18] All possible results of the K-algorithm for this word are distributed in the range of 10–30%. The maximum share of guessed sentences is 30.61%. Note that this value is achieved when $\varepsilon = 0.39$, and this is clearly shown in Figure 12, see the thick dotted line.

All calculations, charts drawn from experimental data and results of the experiments are available online in Google Sheets [8].

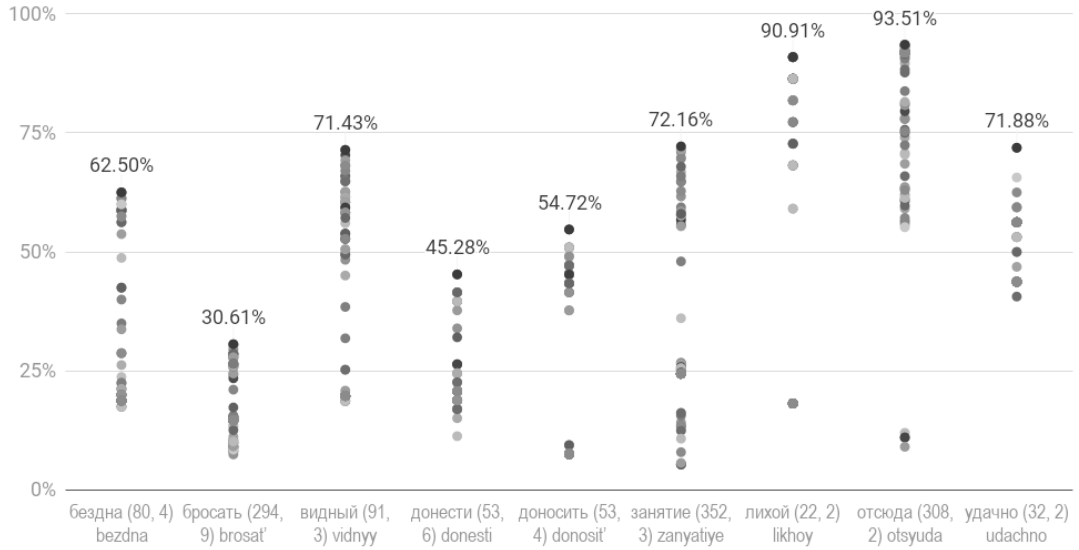

Fig. 15. Proportions of correctly guessed sentences distributed over a set of possible calculated results

Conclusions

The development of the corpus analysis system WCorpus[19] was started. 377 thousand texts were extracted from Russian Wikisource, processed and uploaded to this corpus.

Context-predictive models of the RusVectores project are used to calculate the distance between lemmas. Scripts in Python were developed to process RusVectores data, see the *wcorpus.py* project on the GitHub website.

The WSD algorithm based on a new method of vector-word contexts proximity calculation is proposed and implemented. Experiments have shown that in a number of cases the new algorithm shows better results.

The future work is matching Russian lexical resources (Wiktionary, WCorpus) to Wikidata objects [11].

[18] https://ru.wiktionary.org/wiki/бросать

[19] `https://github.com/componavt/wcorpus`

The study was supported by the Russian Foundation for Basic Research, grant No. 18-012-00117.

References

1. *Arora S., Liang Y., Ma T.* A simple but tough-to-beat baseline for sentence embeddings. *In Proceedings of the ICLR*, 2017. P. 1–16. URL: `https://pdfs.semanticscholar.org/3fc9/7768dc0b36449ec377d6a4cad8827908d5b4.pdf` (access date: 3.04.2018).

2. *Chen X., Liu Z., Sun M.* A unified model for word sense representation and disambiguation. *In Proceedings of the EMNLP*, 2014. P. 1025–1035. doi: 10.3115/v1/d14-1110. URL: `http://www.aclweb.org/anthology/D14-1110` (access date: 3.04.2018).

3. *Choi S. S., Cha S. H., Tappert C. C.* A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*. 2010. Vol. 8. no. 1. P. 43–48. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.`

352.6123&rep=rep1&type=pdf (access date: 3.04.2018).

4. *Haussler D.* Convolution kernels on discrete structures. *Technical report, Department of Computer Science, University of California at Santa Cruz*. 1999. URL: https://www.soe.ucsc.edu/sites/default/files/technical-reports/UCSC-CRL-99-10.pdf (access date: 3.04.2018).

5. *Jurczyk T., Deshmane A., Choi J.* Analysis of Wikipedia-based corpora for question answering. *arXiv preprint arXiv:1801.02073*. 2018. URL: http://arxiv.org/abs/1801.02073 (access date: 3.04.2018).

6. *Krizhanovsky A., Kirillov A.* Calculated attributes of synonym sets. *arXiv preprint arXiv:1803.01580*. 2018. URL: http://arxiv.org/abs/1803.01580 (access date: 3.04.2018).

7. *Krizhanovsky A., Kirillov A., Krizhanovskaya N.* WCorpus mysql database with texts of 3 writers. *figshare*. 2018. URL: https://doi.org/10.6084/m9.figshare.5938150.v1 (access date: 3.04.2018).

8. *Krizhanovsky A., Kirillov A., Krizhanovskaya N.* Assign senses to sentences of 3 writers. *Google Sheets*. 2018. URL: http://bit.ly/2I14QIT (access date: 27.04.2018).

9. *Kutuzov A., Kuzmenko E.* Texts in, meaning out: neural language models in semantic similarity task for Russian. *arXiv preprint arXiv:1504.08183*. 2015. URL: https://arxiv.org/abs/1504.08183 (access date: 3.04.2018).

10. *Lesot M-J., Rifqi M., Benhadda H.* Similarity measures for binary and numerical data: a survey. *International Journal of Knowledge Engineering and Soft Data Paradigms*. 2009. Vol. 1. no. 1. P. 63–84. doi: 10.1504/ijkesdp.2009.021985. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.212.6533&rep=rep1&type=pdf (access date: 3.04.2018).

11. *Nielsen F.* Linking ImageNet WordNet Synsets with Wikidata. *In WWW '18 Companion: The 2018 Web Conference Companion*. 2018. URL: https://arxiv.org/pdf/1803.04349.pdf (access date: 18.04.2018).

Received March 31, 2018

СВЕДЕНИЯ ОБ АВТОРАХ:

Кириллов Александр Николаевич
ведущий научный сотрудник, д. ф.-м. н.
Институт прикладных математических исследований КарНЦ РАН, Федеральный исследовательский центр «Карельский научный центр РАН»
ул. Пушкинская, 11, Петрозаводск,
Республика Карелия, Россия, 185910
эл. почта: kirillov@krc.karelia.ru
тел.: (8142) 766312

Крижановская Наталья Борисовна
ведущий инженер-исследователь
Институт прикладных математических исследований КарНЦ РАН, Федеральный исследовательский центр «Карельский научный центр РАН»
ул. Пушкинская, 11, Петрозаводск,
Республика Карелия, Россия, 185910
эл. почта: nataly@krc.karelia.ru
тел.: (8142) 766312

Крижановский Андрей Анатольевич
рук. лаб. информационных компьютерных технологий, к. т. н.
Институт прикладных математических исследований КарНЦ РАН, Федеральный исследовательский центр «Карельский научный центр РАН»
ул. Пушкинская, 11, Петрозаводск,
Республика Карелия, Россия, 185910
эл. почта: andew.krizhanovsky@gmail.com
тел.: (8142) 766312

CONTRIBUTORS:

Kirillov, Alexander
Institute of Applied Mathematical Research,
Karelian Research Centre,
Russian Academy of Sciences
11 Pushkinskaya St., 185910 Petrozavodsk,
Karelia, Russia
e-mail: kirillov@krc.karelia.ru
tel.: (8142) 766312

Krizhanovskaya, Natalia
Institute of Applied Mathematical Research,
Karelian Research Centre,
Russian Academy of Sciences
11 Pushkinskaya St., 185910 Petrozavodsk,
Karelia, Russia
e-mail: nataly@krc.karelia.ru
tel.: (8142) 766312

Krizhanovsky, Andrew
Institute of Applied Mathematical Research,
Karelian Research Centre,
Russian Academy of Sciences
11 Pushkinskaya St., 185910 Petrozavodsk,
Karelia, Russia
e-mail: andew.krizhanovsky@gmail.com
tel.: (8142) 766312