Keyword Extraction from Single Russian Document

Mikhail Vadimovich Sandul
ITMO University
St. Petersburg, Russia
sandulmv@yandex.ru

Elena Georgievna Mikhailova
Saint Petersburg State University, ITMO University
St. Petersburg, Russia
e.mikhaylova@spbu.ru

***Abstract*—The problem of automatic keyword and phrases extraction from a text occurs in different tasks of information retrieval and text mining. The task is the identification of terms that best describe the subject of a document. Currently there are a lot of research to solve this problem. Basically, algorithms are developed for texts in English. The possibility of applying these algorithms to the Russian texts are not sufficiently investigated. One of the most known algorithms for solving the problem of keyword extraction is RAKE. This article examines the effectiveness of RAKE algorithm for texts in Russian. The work also applies the hybrid method, which uses the Γ-index metric for phrases weighting, which were obtained using the algorithm RAKE. The article shows that this algorithm is more accurate than PAKE while reducing the number of selected phrases.**

I. Introduction

Organizing data into a database requires expenses on database designing, data organizing, etc. These expenses are not necessarily related to the analysis of the data itself. So, we spend additional time and effort on preprocessing to get new knowledge from a data. The worst part is that organizing some types of a data into a database can lead to a loss of the content. Usually it's impossible to convert text documents into a table (in case of relational DB) without loss of its meanings. Thus, texts are usually stored unchanged as BLOB-fields. As a result, we can say that using databases for analysing texts is not efficient.

As we can see, it is hard to bring structure into a text for analysis. However, we would like to extract useful information from it anyway. The branch of science which deals with such problems is called Text Mining[1].

Nowadays a lot of real-world problems are subjects of study of Text Mining starting from typical Data Mining problems such as classification and clustering to exclusively Text Mining problems like automatic keyword extraction and annotation[1].

Currently, amount of unstructured text information constantly grows. Keywords can help to get new knowledge from it because they let us understand the meaning of a text without reading it. Automatic keyword extraction is the subject of study of Text Mining.

Keywords can be applied to improve the functionality of information retrieval systems. For example, Phrasier[8] system uses keywords to find documents related to the primary one and words themselves serve as links between documents. That allows a user to navigate within documents much faster. Another example is Keyphind[9]. It is the search engine for digital libraries. Keywords are used there as the main source for building index. They are also used to enrich the presentation o search results.

Despite the wide range of applications, most documents do not have assigned keywords. Most approaches are focused on manual assignment of keywords. Usually, this procedure is done by specialists in the relevant field. In their work, they may use a fixed taxonomy or rely on author's judgment to provide a representative list[2]. The main goal of further investigations is automatization of this process.

Early approaches to automatic keyword extraction focus on corpus-oriented statistics of individual words. However, they have some flaws. For instance, while some words might be evaluated as keywords within the whole corpus, keywords within a single document or several documents might not[2]. Also, such methods can not help us in finding keywords consists of two or more words. To avoid these drawbacks, we focus on approaches that operate on a single document.

We will describe keyword as a sequence of one or more words which provide a representation of the document's content. An algorithm should return us a list of such sequences. Ideally, this list represents in condense for the essential content of a document. However, there is no need for these sequences to form coherent text. To avoid ambiguity, further in the article we will refer to such sequences as key phrases to highlight that they may contain more than one word. We will use the term "keyword" only to emphasize that we are dealing with a single-word sequence.

To evaluate the efficiency of an algorithm we use Precision and Recall metrics. Recall is the fraction of correctly extracted keywords among all keywords in a document:

$$\frac{correctly\ extracted}{all\ keywords\ in\ a\ text}$$

Precision is the fraction of correctly extracted keywords to a count among all extracted keywords:

$$\frac{correctly\ extracted}{all\ extracted}$$

Different approaches to automatic keyword extraction exist: supervised and unsupervised machine learning algorithms, statistical approaches, approaches based on linguistic features. They all can be used to solve our problem. These methods can be divided into groups if they domain-dependent or domain-independent, corpus-oriented or made for single documents, require training set or not. In this particular work will be considered domain-independent methods which do not require training set.

II. Related Work and Background

A. TextRank

TextRank[3] is the agile graph-based method that can be used not only for key phrase extraction but also for creating annotation for a text.

Representing data as a graph is one of the approaches to retrieve information. The basic idea implemented by a graph-based ranking model is that of ”voting” or ”recommendations”. When there an edge exists from one vertex to another, it is basically casting a vote for that other vertex. The more votes a vertex gets, the more important it becomes. Moreover, an importance of the vertex determines the importance of its ”votes”, and this information is also taken into account by ranking model.

Formally, let $G = (V, E)$ be a directed graph with the set of vertexes V and the set of edges E, where E is the subset of $V \times V$. Let $In(V_i)$ be a set of vertexes which has an edge that goes to V_i. Let $Out(V_i)$ be a set of vertexes to which leads an edge from V_i. We will define a score of a vertex V_i as follows:

$$S(V_i) = (1 - d) + d \cdot \sum_{V_j \in In(V_i)} S(V_j)$$

where $d \in [0; 1]$ is a dumping factor, which has a role of integrating into the model the probability to jump from the given vertex to another random one in the graph. The algorithm starts with arbitrary values of $S(V_i)$ for each node, the computation iterates until convergence below given threshold is achieved.

To apply such method to a text, first, we need to introduce some rules according to which we will build a graph. We need to define units which will be treated as nodes and relations between them. We can distinguish the following steps:

1) Identify text units that best define the task at hand, and add them as vertexes in the graph
2) Identify relations that connect such text units, and add them as edges between respective vertexes
3) Iterate graph-based algorithm until it converges
4) Sort vertexes by their score and use this score for selection decisions

Such definition allows us to apply this method to a wide range of problems. In particular, we will show how to apply this algorithm to key phrase extraction.

First, we have to decide what to use as units when building the graph. Authors suggested to this role nouns and adjectives. They tried different part-of-speech filters but this one showed the highest performance. Two lexical units were related if they co-occur within a window of maximum N words, where N can be set anywhere from 2 to 10. As a result, authors constructed an unweighted and undirected graph. The initial score for each vertex was set to 1. After this, authors ran the algorithm described earlier. Once a final score was obtained for each vertex in the graph, first T vertexes with the highest scores were selected. T may be set to any fixed value or may depend on various factors for instance, on a text size. Then, sequences of adjacent keywords are collapsed into a key phrases.

The dataset used in the experiments is a collection of 500 abstracts from the Inspec database, and the corresponding manually assigned key phrases. Results are shown on Tab. 1.

It should be noted, that due to algorithm definition, the number of vertexes in a graph depends on the number of distinct words in a text. So, for large texts algorithm can be very long to run.

Precision	32.1%
Recall	36.2%

Tab. 1. TextRank evaluations

B. PageRank on Synonym Networks

The problem of a graph growth can be partially solved by merging words into groups. This can also improve the efficiency of the algorithm. One way to form groups is to define equivalence classes. Authors suggest merging words by their meaning. PageRank on Synonym Networks[7] implements this idea. They need a thesaurus for that matter. Each class can be represented by a unique number. Every word in a text is replaced by a unique number that corresponds to the equivalence class of the word. Then the algorithm described in the previous paragraph is applied to the modified text.

C. RAKE

RAKE[2] (Rapid Automatic Keyword Extraction) is a relatively effective algorithm that operates on individual documents, easily applied to new domains and does not depend on the structure of a document or specific grammar rules. RAKE is based on the observation that key phrases frequently contain multiple words but rarely contain standard punctuation or stop-words. As a stop-words, we denote function words like ”and”, ”of”, ”the”, or other words with minimal lexical meaning. It is worth to mention that such words are also ignored in building an index in information retrieval systems. The algorithm requires the list of such words.

In addition to a stop-word list, sets of phrase- and word-delimiters needed. RAKE uses stop-words and phrase-delimiters to partition a document into a set of candidate key phrases. Word delimiters are used to split candidates into individual words.

RAKE begins key phrase extraction with partitioning a text into candidate key phrases. On the next step, the set of unique words is obtained by splitting candidates into individual words with word-delimiters. After this, scores are calculated for each word. Authors propose metrics based on word frequency and word degree. Word frequency ($freq(w)$) is a number of occurrences of the word in candidates. Word degree ($deg(w)$) is a total length of all candidates which contain the word. The score of a word is defined as

$$s(w) = \frac{deg(w)}{freq(w)}$$

Then a score is calculated for each candidate key phrase and defined as the sum of its member word scores.

Authors evaluated the efficiency of the algorithm on the Inspec set, the set of 500 abstracts with manually assigned key phrases. The same dataset was used in testing TextRank effectiveness. They also compared performance of the algorithm with different stop-word lists. The results are shown on the Tab. 2.

As we can see, the difference in the Precision with generated stop-word list based on keyword adjacency (KA stoplist) and with Fox's stop-word list is quite noticeable. This means that RAKE is strongly depends on the provided set of stop-words.

stop-word list	Precision	Recall
KA stop-word list	33.7%	41.5%
Fox stop-word list	26.0%	42.2%

Tab. 2. RAKE evaluations

D. Position Weighing

Position Weighing[6] is based on the fact that a word's position plays an important role in linguistics. Words in different positions carry different entropy. For example, if a word appears in an introduction or in a conclusion it usually carries more information. This observation also applies to words within the same paragraph: words appeared in a leading and summarization sentences are often more important than those in other positions of the same paragraph.

Authors propose their approach called Position Weight. For each word position they suggest computing a score which strongly depends on a paragraph and a sentence where the word occurrence is found and on a form of the word. Here is the formula:

$$pw(t_i) = pw(t_i, p_j) * pw(t_i, s_k) * pw(t_i, w_r)$$

Where $pw(t_i, p_j)$ represents the score of an occurrence t_i within the paragraph j; $pw(t_i, s_k)$ represents the score of an occurrence t_i within the sentence k; $pw(t_i, w_r)$ represents the score of an occurrence t_i as a word for r. Then the total weight of a term t in a document d computed as the sum of all position scores:

$$PW(t, d) = \sum_{i=1}^{m} pw(t_i)$$

We can conclude from the definition of the algorithm that it relies on the structure of a document. Also, we should notice when assumptions about a document structure are violated (for example, there are no clues to determine a type of sentences and paragraphs), the algorithm is reduced to simply calculating the frequency of a word.

E. Statistical Approach

1) Weighing of Individual Words: In addition to the frequency of a word, one can also obtain information about the distribution of a word. The authors of the approach suggest using this information to determine the significance of individual words. The metrics proposed in [5] are based on the phenomenon of the attraction of keywords in the text. So, for example, the authors give the distribution of distances to the nearest neighbor for the word "NATURE" in the book "The Origin of Species" by Charles Darwin (Fig. 1). On the graph, you can see that the distribution of distances resembles an exponential distribution. In the case of noise words, this property is much weaker. Thus, keywords tend to form clusters in a text unlike noise words.

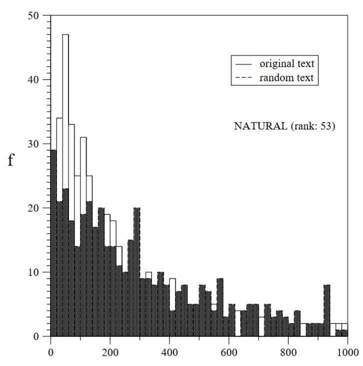

Fig. 1. Distribution of distances to the closest neighbor

2) σ-index: To study the spatial distribution of a given word in a text, first, we denote by t_i the absolute position in the corpus of the $i-th$ occurrence of a word. Thus we obtain a sequence of such positions: $\{t_0, t_1, \dots, t_{n+1}\}$, assuming there are $n+2$ occurrences. Next, we need to compute the average distance between two successive words:

$$\mu = \frac{1}{n+1} \sum_{i=0}^{n} (t_{i+1} - t_i)$$

The following step is to obtain the standard deviation:

$$s = \sqrt{\frac{1}{n+1} \sum_{i=0}^{n} ((t_{i+1} - t_i) - \mu)^2}$$

To eliminate the dependence on the frequency of occurrence for different words, authors suggest normalizing the token spacing, thus they define σ-index as follows:

$$s/\mu$$

Given that standard deviation grows rapidly when inhomogeneity of a distribution spacing $t_{i+1} - t_i$ increases. However, a value of sigma can be strongly affected by the change of a single occurrence position. Also, high values of sigma do not always imply a cluster concentration.

3) Γ-index: To solve such problems, authors suggest considering the average separation around the occurrence at t_i. They define it as follows:

$$d(t_i) = \frac{(t_{i+1} - t_i) - (t_i - t_{i-1})}{2} = \frac{t_{i+1} - t_{i-1}}{2}$$

. For each position t_i the *cluster index* γ is computed:

$$\gamma(t_i) = \begin{cases} \frac{\mu - d(t_i)}{\mu}, & if\ d(t_i) < \mu \\ 0, & else \end{cases}$$

Finally, the score of a word is obtained from the average of all its cluster indexes:

$$\Gamma = \frac{1}{n} \sum_{i=1}^{n} \gamma(t_i)$$

Γ-index is more stable than σ-index. However, it is more time-consuming to evaluate than σ.

Precision	31.8%
Recall	74.2%

Tab. 3. Evaluations for RAKE without any modification

4) Extracting of Key Phrases: The metrics described above makes it possible to rank only individual words. In [4], proposed several ways to generalize metrics to two-word phrases:

First Way: Rank whole word sequences. The weight of the sequence is calculated similarly to a single word. The problem is that the frequency of the desired phrases may be insufficient for analysis.

Second way: Make up all possible unique phrases of a given length. To compute the score of a phrase we need to sum up all score of individual words from which the phrase consists. The problem with this approach is that the number of phrases grows exponentially with the given length.

III. Experiments

A. Data Description

We used the book "Abel Theorem in Solutions Problems" written by Alekseev V.B. as a test data. The text consists of 39519 words 1576 of which are unique. The alphabetical index at the end of the book was used to evaluate the retrieval capabilities of different key phrase extractors. The choice of the test data is explained, firstly, by a large volume of the text, which allows us to give more accurate estimates of the efficiency of algorithms, and secondly, by the requirements of the statistical approach to a number of occurrences of analyzed words (for small texts this condition may not be fulfilled even for one word).

B. Getting Evaluations

We measured the efficiency of algorithms in terms of Recall and Precision. The result of each algorithm is a set of phrases. To obtain the Precision estimate, we must determine how many key phrases are in the result set. We will assume that the phrase from the result set is a key phrase if it fully contains any key phrase from the reference set and the order of words does not count. To obtain the Recall estimate, we must determine how many key phrases from the reference set are in the result set. Also, we will assume that the phrase from the reference set is in the result set if it is fully contained in any phrase in the result set without regard to the order of words. In addition, all the words in all phrases in both sets were stemmed, which solves the problem with comparing different forms of a word.

C. RAKE

The algorithm requires a set of stop-words. We used publicly available one from ranks.nl site. To avoid the problem with different forms of words We used SnowballStemmer from the nltk library for Python. Every word in every phrase was stemmed. For each stemmed phrase was kept the count of unstemmed ones from which it could be obtained. These numbers were used in computing scores for individual words. Results are shown on Tab. 3:

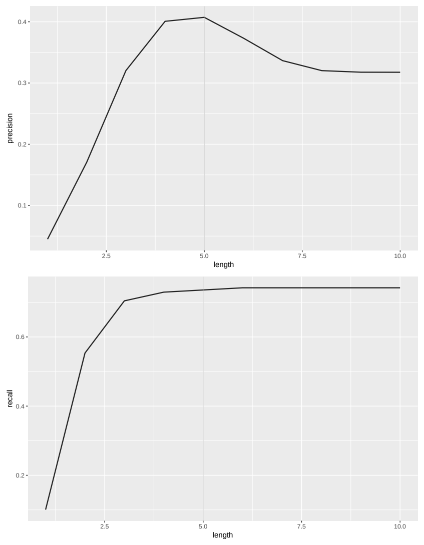

Fig. 2. The dependence of the efficiency on the restriction on the length of a phrase

Max. length	Precision	Recall	NWords
4	40.8%	73.0%	9327
5	41.5%	73.6%	7293

Tab. 4. RAKE evaluations with phrase length limitations

The result set contained 4320 phrases and all these phrases were unique in terms of word forms. However, we faced the problem: the output contained long phrases, for example, the longest one consisted of 10 words. We would like to have shorter phrases. It was decided to set the limit for the maximum phrase length. All phrases with length above the limit were split into shorter ones. From words that make up the long phrase, all possible phrases of a given length were made without taking into account the word order. Also, the limit was set for the maximum degree of each word to support this changes.

Next, We studied the dependency of efficiency on the restriction on the length of a phrase. As we can see on the Fig.2 the highest precision was reached when the maximal phrase length was limited to 4 or 5 words. On Tab. 5 are shown exact evaluations for RAKE with phrase length limitations.

It makes sense to consider only such restrictions on length, since at these limitations Precision reaches its highest values, with Recall close to maximum. It can be seen that the number of words extracted by the algorithm has increased dramatically. At the same time, the number of correctly extracted words also increased, because there is an increase in Precision.

This way of generating candidates, in theory, can cause an exponential growth of their number. By the length of a phrase we will understand the number of words it consists of. Let k be a restriction on the maximum length of a phrase, m - the maximum length of a phrase that hit the derivation of the algorithm. By the definition of the algorithm, phrases obtained after partitioning do not have intersections in the text, i.e. for any two different phrases, their positions in the text will not

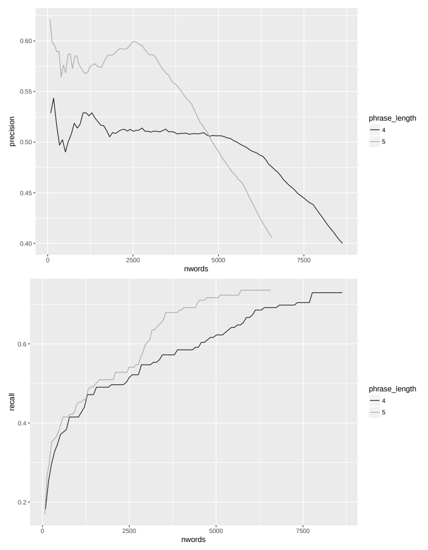

Fig. 3. Comparison of estimates for RAKE with 4 and 5 words phrase length limitations. Graph shows dependence on the number of words with the highest ranks extracted

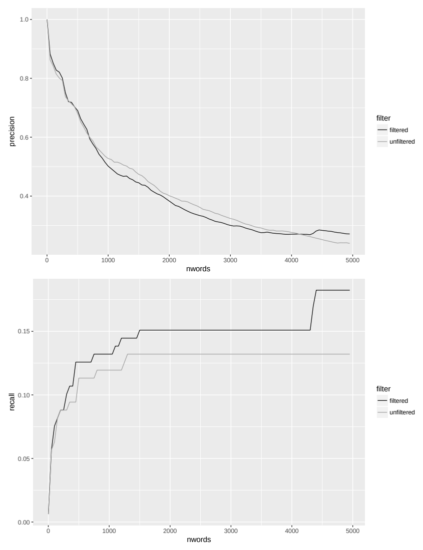

Fig. 4. Dependence of estimates for statistical approach on the number of words with the highest ranks extracted when phrase length is limited to two words

overlap. Let N be the number of words in a text. Thus, the number of such phrases is not greater than $\frac{N}{m}$. We assume that $m > k$. When generating new phrases, the word order is not taken into account, so the number of new phrases can be calculated as the number of combinations without repetitions:

$$C_m^k = \frac{m!}{(m-k)! \cdot k!}$$

Thus, we can give the following estimate of the new number of phrases:

$$\frac{N}{m} \cdot C_m^k$$

As we can see, the growth is proportional to the number of combinations. By definition of the algorithm, the length of the key phrase in average can not be more than the average sentence length. The average sentence consists of 10-11 words[10], which means that the average value of m will not exceed this number. In this way, we can conclude that the increase in the number of candidates will be within reasonable limits for random texts.

Limiting the size of the result set, we can improve Precision of an algorithm. But discarding part of the results may lead to the decreasing of Recall. Fig. 3 shows the dependence of Precision and Recall on the number of selected candidates.

D. Statistical Approach

The statistical approach provides the way of measuring the significance of individual words. Nevertheless, key phrases can consist of more than one word. In this connection, the question arises how to form a set of candidates. The solutions proposed in [4] showed good results, but they require a large amount of memory to store all candidates. For example, the number of all possible key phrases formed of two words (without taking into account the order) will be equal to the number of combinations. So, for the text used in the work, the number of unique words is around 1500, thus there will be around 10^6 phrases of two words and around 10^9 phrases of three words. Actually, there is no need to store all candidates as we only need first T of them with the maximal score.

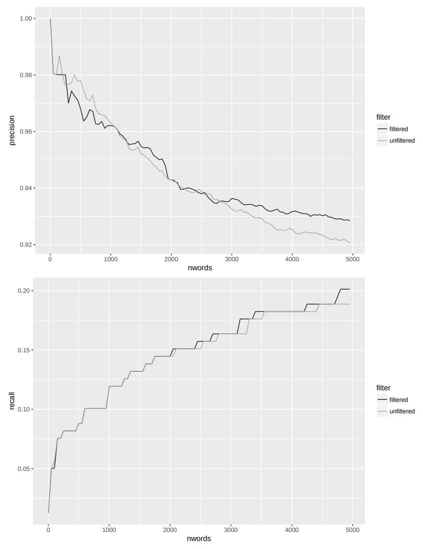

Fig. 5. Dependence of estimates for statistical approach on the number of words with the highest ranks extracted when phrase length is limited to three words

Selection of candidates: We will consider the case of selecting phrases from two words. This problem can be addressed to the next. Given two sorted arrays $A[1 \dots n]$ and $B[1 \dots n]$. We want to print all n^2 sums $A[i] + B[j]$ of two elements from different arrays in descending order. There is the solution to the problem that performs in $O(n^2 log(n))$ time and requires $O(n)$ space.

Description of the solution of the problem of arrays: We will use max-heap to store tuples consist of a sum $A[i] + B[j]$ and a couple of indexes (i, j) which defines such sum. Elements in the heap are ordered by the first element of a tuple.

At the beginning, heap stores all pairs such as: $\forall j = 0 \dots n : (A[0] + B[j], (0, j))$.

The step of the algorithm is that the first element is extracted from the heap - this is the element with the largest sum. Suppose that indexes (i, j) were associated with that sum. We print extracted sum and then put a new sum in the heap $A[i+1] + B[j]$ with the pair of respective indexes. When the elements of the arrays are finished, the remaining content of the heap is displayed. From the definition of the solution, the validity of estimates of time and space consumption is obvious.

Correctness of the solution: To prove the correctness of the solution it is handy to consider square matrix formed from all possible sums. We denote such matrix as C and it forms as follows: $C[i, j] = A[i] + B[j]$. Since the elements of arrays A and B are sorted, we can notice that the elements of the matrix are decreasing from left to right and from top to bottom. In other words: $\forall i, j : c[i, j] \geq c[i + 1, j]$ and $c[i, j] \geq c[i, j + 1]$.

At each step, the following invariant is supported: the heap stores the maximal element of each column that have not been printed yet. Since the maximal element that should get into the output is in one of the columns, this proves the correctness.

Efficiency of the algorithm: Before calculating scores, each word was stemmed with SnowballStemmer taken from nltk version 3.2.2. Thus, words that differ only by their forms were merged into equivalence classes. Then, scores were calculated for such classes. We used $Gamma$-index in the experiments. Also, at the stage of selecting words for weighing, it was decided to use a filter along the length of the word: those words which length was less than the specified threshold did not participate in the weighing and in the construction of candidates. In the experiments with filtering the threshold value for the word length was set to 3. Experiments were carried out for phrases of two and three words.

On Fig. 4 and Fig. 5 is shown that the use of the filter have almost no impact on Recall and Precision of the algorithm.

The considered algorithm shows high Precision. This is due to the construction of candidates. Γ-index accurately ranks individual words, hence top-scored ones are most likely to be actual key phrases or to be contained in multi-word key phrases. Because of the way of constructing candidates and due to the definition of efficiency metrics top-scored candidates are most likely to be treated as properly extracted key phrases.

Based on the results of the experiments, it can be concluded that the metrics Γ-index correctly ranks individual words. Nevertheless, artificially constructed phrases can not ensure the high Recall of the algorithm.

E. Hybrid approach

The advantage of the statistical approach is the high Precision of the ranking of single words. And the main drawback is the complexity of constructing candidates and incapability to ensure high Recall with those candidates. On the other hand, the RAKE algorithm allows us to create phrases that provide a relatively high Recall, but proposed ranking formula does not allow to achieve high Precision while limiting the number of candidates. In this connection, the idea of using Γ-index for ranking of a single word, but use phrases that are obtained as a result of the RAKE algorithm. Fig. 6 and Fig. 7 show a comparison of corresponding methods.

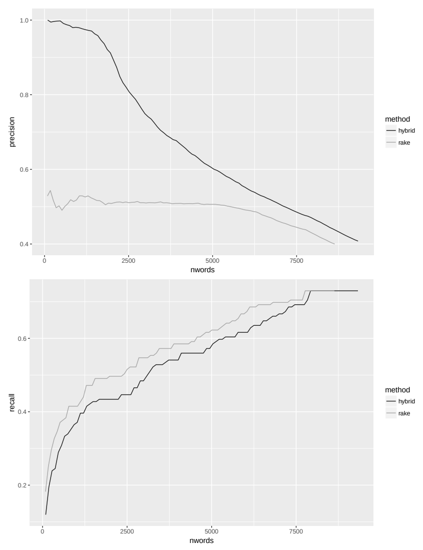

Fig. 6. Comparison of the efficiency of algorithms RAKE and Hybrid when the length of phrases is limited to 4 words. The graph shows dependence of estimates on the number of words with most ranks extracted for considered methods

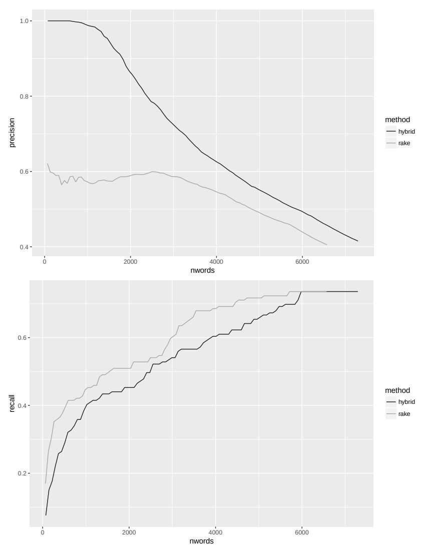

Fig. 7. Comparison of the efficiency of algorithms RAKE and Hybrid when the length of phrases is limited to 5 words. The graph shows dependence of estimates on the number of words with most ranks extracted for considered methods

IV. Conclusion

In the work it was shown that, with certain additions, the RAKE algorithm and the statistical approach can be used to extract key phrases from Russian texts. In addition, the paper proposed new Hybrid method that uses Γ-index for weighing phrases that are obtained as a result of RAKE algorithm. And it was shown that the algorithm allows to achieve higher Precision and Recall comparing to other algorithms considered in the paper.

V. References

1. Analysis of data and processes[Text] / A.A. Barsegyan, M.S. Kupriyanov, I.I. Kholod, and others. – BVH-Petesburg, 2009.-510 p.
2. Michael Berry. Text Mining: Applications and Theory[] / Michael Berry, Jacob Kogan – 2010, John Wiley and Sons, Ltd.-205 p.
3. Rada Mihalcea, Paul Tarau. TextRank: Bringing Order into Texts // In Proceedings of EMNLP 2004 (ed. Lin D and Wu D), pp. 404-411
4. Siddiqi, S., Sharan, A. Keyword and keyphrase extraction from single Hindi document using statistical approach // 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN) 2015, pp. 713-718
5. Juan P. Herrera, Pedro A. Pury. Statistical Keyword Detection in Literary Corpora // The European Physical Journal B, 2008, pp. 135-146
6. Xinghua Hu, Bin Wu. Automatic Keyword Exctraction Using Linguistic Features //Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06), 2006, pp. 19-23
7. Zhengyang Liu, Jianyi Liu, Wenbin Yao, Cong Wang. Keyword Extraction Using PageRank on Synonym Networks // 2010 International Conference on E-Product E-Service and E-Entertainment, 2010, pp. 1-4
8. Jones S., Paynter G. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications // Journal of the American Society for Information Science and Technology, 2002
9. Gutwin C, Paynter G, Witten I, Nevill-Manning C., Frank E. Improving browsing in digital libraries with keyphrase indexes // Decision Support Systems 27(12), 1999, pp. 81-104
10. S.A. Sharov. Frequency dictionary [Online]. Available: http://www.artint.ru/projects/frqlist.php [Accessed: 20.05.2017]