Journal of Theoretical and Applied Information Technology
30th November 2020. Vol.98. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UNSUPERVISED KEYWORD EXTRACTION USING NON-SMOOTH NMF

[1]ALIYA NUGUMANOVA, [2,3]DARKHAN AHMED-ZAKI, [2]MADINA MANSUROVA, [1]YERZHAN BAIBURIN, [4]KURMASH APAYEV, [1]ALMASBEK MAULIT

[1] S. Amanzholov East Kazakhstan State University, Ust-Kamenogorsk, Republic of Kazakhstan

[2] Al-Farabi Kazakh National University, Almaty, Republic of Kazakhstan

[3] University of International Business, Almaty, Republic of Kazakhstan

[4] D. Serikbayev East Kazakhstan State Technical University, Ust-Kamenogorsk, Republic of Kazakhstan

E-mail: [1]yalisha@yandex.kz, [2]darhan_a@mail.ru, [3]mansurova.madina@gmail.com, [4]ebaiburin@gmail.com, [5]kapaev@ektu.kz, [6] maulit.almas@yandex.ru

ABSTRACT

In this paper, we introduce a novel unsupervised method for keyword extraction, based on non-smooth nonnegative matrix factorization. We generate a document-term matrix from a given corpus and factorize it into the product of two special matrices: documents-by-topics and topics-by-terms. In our method, we choose a low degree of factorization (k=3,4,5) and use only topics-by-terms matrix to extract top N keywords for each of k topics. Then we merge these obtained N*k keywords into a resulting keyword list excluding duplicates and assign keywords to documents. We validate our method with a large text corpora: "Introduction to information retrieval" textbook (by Manning, Raghavan and Schütze), available online. The result of our method is compared with three popular unsupervised keyword extraction algorithms: TextRank, Rake and Yake. The experiments confirm that the proposed method shows the promising performance in terms of precision, recall and F-measure with respect to various number of candidate keywords.

Keywords: *Keyword Extraction, NMF, nsNMF, NLP, Unsupervised Approach.*

1. INTRODUCTION

Keywords are words and phrases that represent the content of a given text in a compact and informative manner [1,2]. Keywords provide a high-level thematic representation of texts and can be served as rich sources of information for various applications of NLP [2]. There are a lot of approaches for automatic keyword extraction, and they all can be divided into two classes: unsupervised and supervised. The main disadvantage of supervised keyword extraction approach is that it requires manually annotated keywords for training [3]. Unsupervised keyword extraction approach which does not depend on any external data is very promising for deployment in online applications [2]. Such applications allow users to upload their texts and immediately get relevant keywords, sometimes in the form of tags or visual word clouds.

Within unsupervised methods, keyword extraction task is framed as a term ranking problem [2]. Among the most popular ranking algorithms are TextRank [4], Rake [5], SwiftRank [6], Yake [7,8], Take [9], and adapted Chi-square [10]. These algorithms show that the performance of F-score is near 10-30% [8], and as pointed in [11], "there is definitely still room for improvement". Ten years have passed since the writing of the cited paper [11], however the room for improvement is still wide open.

This work aims to increase the performance of unsupervised keyword extraction using non-smooth nonnegative matrix factorization [12]. Nonnegative matrix factorization (NMF) decomposes of an original document-term co-occurrence matrix where rows are documents and columns are terms, into the product of two lower rank matrices: documents-by-features and features-by-terms. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

number of features k (i.e. factorization degree) is usually selected much smaller than the number of terms, so each feature can be interpreted as a compressed topic. Therefore, the first matrix (documents-by-features) represents weights of topics in documents, also known as basis vectors, and the second matrix (features-by-terms) represents weights of terms in topics, also known as encoding vectors. Our hypothesis states that the most weighted terms in topics can be selected as keywords. There is a hard reason why only non-smooth NMF should be used; it is to avoid excess overlapping among the encoding vectors (i.e. topics) [12]. This reason will be described in more detail in the following sections.

This paper is organized as follows. Section 2 provides a brief discussion of the related work on unsupervised keyword extraction approach. In Section 3, the novel non-smooth NMF-based approach to keywords extraction is introduced, and difference between the standard and non-smooth NMF is described in detail. Section 4 presents experimental work aimed at evaluating and comparing several unsupervised keyword extraction methods including ours. Finally, Section 5 contains conclusions and plans for future work.

2. RELATED WORK

There are four major classes of unsupervised keyword extraction methods: statistics-based, graph-based, topic-based and embeddings-based. Statistics-based methods use term statistics such as term frequency or co-occurrence to extract the most statistically meaningful terms with the exception of general words and stop-words. These methods include TF-IDF [14], KP-Miner [15], Rake [5], Yake [7], Chi-square [10]. Graph-based methods represent documents as graphs and use graph metrics such as node authority or node degree to extract the most authoritative terms. These methods include TextRank [4], SingleRank [16], PositionRank [17], KECNW [18], MCI [19], sCake [20]. Topic-based methods use topic modeling to extract the most representative terms in topics. These methods include TopicRank [21], Topical PageRank [22, 23], LET [24], LSA [25]. Embeddings-based methods use word embeddings to extract the most semantically related terms. These methods include EmbedRank [26], RVA [27], OEWE [28], NamedKey [29].

The method proposed in this work refers to the class of topic-based keyword extraction methods so we will consider this issue in more detail below. The authors of [21] suggest that "intuitively, ranking topics instead of words is a more straightforward way to identify keyphrases that covers the main topics of a document". They adapt TextRank to topic ranking and propose a new method named TopicRank. TopicRank represents a document by a graph in which topics are vertices and edges are semantic relations between topics. The authors define topics as clusters of similar keyphrase candidates (two keyphrase candidates are similar if they have at least 25% of overlapping words). Two topics are related if their keyphrase candidates often appear close to each other in the document. Once the graph is created, the authors apply the above-mentioned method TextRank to rank the topics, and for each of the most important topics, they select only one the most representative keyphrase.

The authors of [22] use Latent Dirichlet Allocation (LDA) to acquire the topics and calculate topic scores for all words in the document. They construct a graph in which vertices are words and edges are weighted in accordance with the number of co-occurrences between corresponding words. The authors apply biased PageRank to this graph and name their method as Topical PageRank. In traditional PageRank there are equal probabilities of random jump to all vertices, whereas in biased PageRank the larger topic score of a vertex, the larger its probability. Therefore, the authors calculate the ranking scores of candidate keyphrases separately for each topic. By considering the topic distribution in the document, they further integrate topic-specific rankings of candidate keyphrases into a final ranking and extract top-ranked ones.

The authors of [23] propose an improvement of Topical PageRank. While the original algorithm requires a random walk for each topic in the document, the authors' Topical PageRank is computed only once regardless of the number of topics in the document. They use the cosine similarity between the vector of word-topic probabilities and the document-topic probabilities and assess the "topical importance" of a word for this document. The authors show that this modification does not reduce the performance too much but significantly saves the computation time.

The authors of [24] apply Latent Semantic Analysis (LSA) to extract latent topics from documents. LSA maps documents and terms into a k dimensional space, and each dimension is called a latent topic. The mapping is achieved through decomposition of the original document-term matrix $A_{m\times n}$ into the product of three matrices

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$U_{m\times k}$, $S_{k\times k}$ and $(V_{n\times k})^T$, where the rows of U are coordinates of points in this k dimensional space representing documents, the rows of V are coordinates of terms, respectively, and the elements of S are singular values which express the importance of latent topics. Therefore, matrices U and V reflect importance weights of latent topics on documents and terms, respectively. Using these matrices, the authors for each candidate keyphrase and for each topic compute a topic-based weight of the keyphrase and select keyphrases with the highest topic-based weights.

The author of [25] also uses LSA to extract keywords from a single document. At first, the document is represented as a matrix A, where rows correspond to the terms and columns to the sentences. Then the term-sentence matrix $A_{n\times s}$ is decomposed into the product of three matrices $U_{n\times k}$, $S_{k\times k}$ and $(V_{s\times k})^T$, and the first column of U is considered as the main topic of the document. At the end, the most important keywords representing the document are picked up by selecting the top N values of this column. This paper is very relevant to our work except for decomposition technique. In our work, nonnegative matrix factorization is used instead of LSA's singular value decomposition.

3. PROPOSED APPROACH

The proposed approach consists of four steps: 1) construction of a document-term matrix; 2) non-smooth NMF decomposition of the document-term matrix into document-topic and topic-term matrices; 3) keywords extraction based on topic-term matrix; 4) assigning keywords to documents, based on the document-topic matrix.

3.1 Construction of Document-Term Matrix

In the first step, we construct a document-term matrix that reflects a given textual collection. If a collection consists of only one document, we divide this document into parts (chapters or paragraphs), so each part can be considered as an individual document. Then we extract terms (i.e. single words, bigrams and trigrams) from these documents, and create a matrix where the rows are documents, columns are extracted terms and entries are the frequencies of terms in documents. We denote this matrix as A.

We impose the following simple restrictions on extracted terms: all terms before extraction are lemmatized; all terms with frequencies lower than a certain threshold are excluded; all terms that contain stop-words are excluded (except trigrams in which stop-words can occur in the middle, for example, "bag of words"); single words that are not nouns are excluded; all terms that include special words such as verbs, adverbs, auxiliary verbs, numbers, symbols, pronouns, determiners, etc. are excluded.

3.2 Non-Smooth NMF Decomposition of Document-Term Matrix

In the second step, we apply non-smooth nonnegative matrix factorization (nsNMF) to the constructed document-term matrix A. In general, NMF is used to represent an input nonnegative matrix A by two smaller nonnegative matrices W and H, which, when multiplied, approximately reconstruct A:

$$A \approx WH \quad (1)$$

This standard NMF model is called a two-factor model. In case when the input matrix A represents a document-term matrix, the matrices W and H represent a document-topic and a topic-term matrices, respectively (see Fig. 1).

The limitation of standard two-factor NMF model is that it often returns overlapped features in the output matrices. In our case, two-factor NMF returns overlapped topics in a topic-term matrix. To overcome this limitation, we use a tri-factor non-smooth NMF model that represents the input matrix A by three nonnegative matrices W, S and H [31]:

$$A \approx WSH \quad (2)$$

In this model, matrix S is fixed and known, and can be used to control the sparsity or smoothness of matrix W and/or H [32]:

$$S = (1-\theta)I_k + \frac{\theta}{k}\cdot 1_{k\times k} \quad (3)$$

where k is the factorization degree, I_k is $k \times k$ identity matrix and $1_{k\times k}$ is the matrix of all ones. The parameter θ controls the smoothness of matrix S. For $\theta = 0$, S = I_k , and the model reduces to the standard two-factor NMF, and for $\theta \to 1$ strong smoothing is imposed on S. Therefore, we apply nsNMF to increase sparseness of both document-topic and topic-term matrices in order to avoid the overlapping of topics.

3.3 Keywords Extraction Based on the Topic-Term Matrix

In the third step, we extract keywords from topic-term matrix H. Each row of matrix H is a vector that encodes one topic and its elements are weights of terms in this topic. We sort each of k

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

rows of matrix H in a descending order, so the most weighty terms will be at the top. Then we extract top N terms from each row and combine these $N \cdot k$ extracted terms into a common list of keywords excluding duplicates.

Table 1 shows the top 10 keywords of "Introduction to Information Retrieval" textbook, extracted using nsNMF (the number of topics k is equal to 3 and the lower threshold of term frequency T is equal to 3). As shown in Table 1, the top 10 keywords in all topics are single words, whereas many researchers point to the dominance of compound terms among keywords [33]. We can adjust this bias by increasing weights of bigrams and trigrams in encoding vectors. We double the weights of bigrams and triple the weights of trigrams. Table 2 shows how the top 10 keywords list is reordered after adjusting weights.

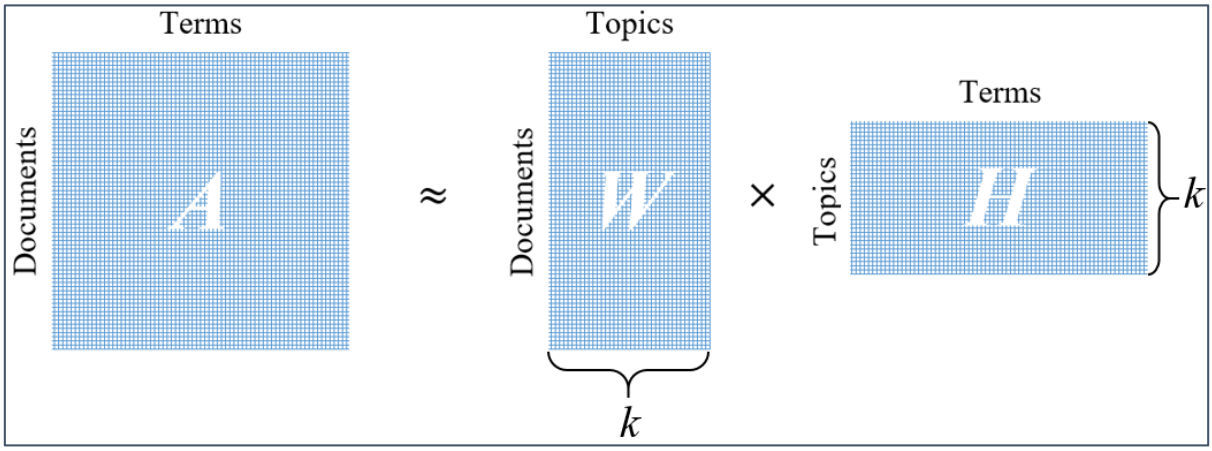

Figure 1: Conceptual Illustration of NMF Applied to a Term-Document Matrix A (Source: [30])

Table 1 : The Top 10 Keywords Extracted from "Introduction to Information Retrieval" Textbook

Rank	Topic 1		Topic 2		Topic 3	
	Term	Weight	Term	Weight	Term	Weight
1	document	700.468	page	626.268	document	1073.532
2	class	405.000	web	424.000	term	993.398
3	set	308.756	list	299.321	query	832.000
4	classification	296.000	search	294.599	model	325.186
5	information	291.083	cluster	273.398	word	310.000
6	data	275.613	posting	260.600	vector	247.610
7	classifier	252.000	index	251.887	language	198.000
8	method	240.180	clustering	225.281	section	192.061
9	text	225.326	time	215.475	space	179.056
10	problem	204.008	number	197.417	collection	162.599

Table 2: The Biased Top 10 Keywords Extracted from "Introduction to Information Retrieval" Textbook (After Adjusting Keyword Weights Based on the Number of Words in Keywords)

Rank	Topic 1		Topic 2		Topic 3	
	Term	Weight	Term	Weight	Term	Weight
1	document	700.468	page	626.268	document	1073.532
2	class	405.000	posting list	624.000	term	993.398
3	text classification	372.000	web page	604.000	query	832.000
4	information retrieval	315.708	search engine	434.436	vector space	425.316
5	set	308.756	web	424.000	model	325.186

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6	relevance feedback	304.000	web search engine	414.000	query term	320.000
7	classification	296.000	web search	348.000	word	310.000
8	information	291.083	list	299.321	language model	256.000
9	data	275.613	search	294.599	Vector	247.610
10	set of documents	263.655	cluster	273.398	language	198.000

It is worth noting that in this step, keyword extraction process might be finished if there is no need in further assigning keywords to documents.

3.4 Assigning Keywords to Documents

The keywords extracted from matrix H describe a collection of documents as a whole and this does not quite correspond to the conventional understanding of the task of keyword extraction. As noted in [33], the difference between extraction of keywords and extraction of terms which is sometimes called automatic term recognition is that keywords are intended for annotation of a separate document, while terms are descriptors of all the subject domain. Accordingly, if the task is not to annotate separately each document of the collection, this step – distribution of the extracted keywords to separate documents – can be omitted.

If this step can't be omitted, after obtaining the total list of keywords it is necessary to compute the weight of every keyword in each document of the collection. There are a great number of methods for doing this, and the simplest one among them is to sort out the list of keywords according to their occurrence in the document and select those which come most often in this document.

Also, it is possible to use to keywords a measure $Tf\text{-}Idf$, which smoothes the frequency of all keywords found in all or almost all documents of the collection. In this work, we use this measure but amplifying it by a multiple of the number of terms included in the keyword:

$$Tf\text{-}Idf(t,d)^{**} = len(t) \times \frac{f(t,d)}{\sum_{t_j \in d} f(t_j,d)} log\left(\frac{|D|}{|\{d_i \in D | t \in d\}|}\right), \quad (4)$$

where $len(t)$ is the length of the keyword, i.e. the number of single words in it; $f(t,d)$ is frequency of the keyword in this document; $\sum_{t_j \in d} f(t_j,d)$ is the sum of frequencies of all keywords in this document; $|D|$ is cardinality of the collection of documents, i.e. the number of all documents in the collection; $|\{d_i \in D | t \in d\}|$ is the number of those documents in the collection which contain this keyword.

Also, in this work, to evaluate the weight of a keyword in the document, we use our own empirical formula:

$$Score(t,d) = = len(t) \times \left(1 + \frac{w(t)}{\sum_{t_j \in Kw} w(t_j)}\right) \times \left(1 + \frac{f(t,d)}{\sum_{t_j \in d} f(t_j,d)}\right) \quad (5)$$

where $w(t)$ is the weight of a keyword, $\sum_{t_j \in Kw} w(t_j)$ – the sum of weights of extracted keywords, $f(t,d)$ – frequency of the keyword in this documents, $\sum_{t_j \in d} f(t_j,d)$ – the sum of frequencies of all extracted keywords occurring in this document.

Table 3 shows the top 10 keywords extracted with the help of formula (4) for the documents "13.3 The Bernoulli model", "9.2.3 Automatic thesaurus generation" and "8 Evaluation in information retrieval" of "Introduction to Information Retrieval" textbook.

Table 3: The Top 10 Keywords Assigned to Documents "The Bernoulli Model", "Automatic Thesaurus Generation" and "Evaluation in Information Retrieval"

Rank	The Bernoulli model		Automatic thesaurus generation		Evaluation in information retrieval	
	Term	Weight	Term	Weight	Term	Weight
1	Bernoulli model	0.363	thesaurus	0.189	test collection	0.123
2	multinomial model	0.291	interest rate	0.180	retrieval result	0.123
3	document of class	0.200	positive and false	0.126	inverse document frequency	0.103

4	model	0.157	expansion	0.109	relevant and non relevant	0.087
5	test document	0.127	collection of document	0.105	notion	0.085
6	classification decision	0.094	term and document	0.101	user	0.081
7	occurrence	0.087	relevance feedback	0.098	utility	0.081
8	number of occurrence	0.065	false negative	0.079	quality	0.069
9	generative model	0.062	document matrix	0.065	inverse document	0.068
10	fraction	0.059	false positive	0.065	user interface	0.068

4. EXPERIMENTAL WORK

4.1 Data

To evaluate the effectiveness of the proposed approach, we tested it on "Introduction to information retrieval" textbook (by Manning, Raghavan and Schütze) [33], available online. "Introduction to information retrieval" textbook consists of 245 texts, where each text corresponds to a section or subsection in the book.

This collection of texts is transformed into a corpus suitable for further analysis with the help of the linguistic processing package Udpipe realized in the language R. In particular, the package allows dividing the corpus documents into sentences and words providing each word with its lemma and POS tag (morphological information on the word). Lemmas are necessary to reduce the words to normal forms and POS tags – to exclude, where it is necessary, auxiliary words and tokens (for example, interjections, articles, punctuation marks). Then, using the same package, single words, bigrams and trigrams were extracted from the corpus (see Fig. 2).

The so-called gold index was used as a standard set of keywords with which the extracted keywords – candidates were compared. The copyright index numbers 604 keywords including 174 single word terms, 336 bigrams, 79 trigrams, 14 4-grams and 1 6-gram. We used 589 keywords as a reference, i.e. only unigrams, bigrams and trigrams. Figure 3 shows a fragment of this gold index.

4.2 Baseline Algorithms

As mentioned above, we compare the approach being proposed with three algorithms for extracting keywords: TextRank, Rake and Yake! The choice of these algorithms is determined by the fact that, for them, there are

	doc_id	lemma	bigram	trigram
1	1 Boolean retrieval.txt	the	the meaning	the meaning of
2	1 Boolean retrieval.txt	meaning	meaning of	meaning of the
3	1 Boolean retrieval.txt	of	of the	of the term
4	1 Boolean retrieval.txt	the	the term	the term information
5	1 Boolean retrieval.txt	term	term information	term information retrieval
6	1 Boolean retrieval.txt	information	information retrieval	information retrieval can
7	1 Boolean retrieval.txt	retrieval	retrieval can	retrieval can be

Figure 2: Results of Applying Sliding Window Size of 3 to the First Sentence of the Chapter "Boolean Retrieval

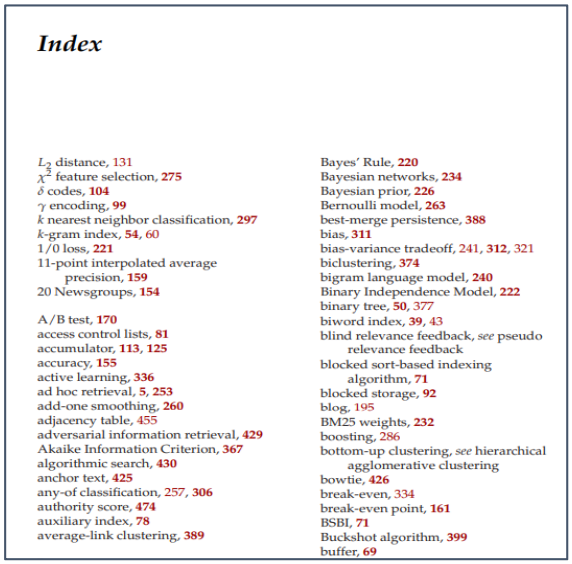

Figure 3: A Fragment of the Author's Keyword Index for "Introduction to Information Retrieval" Textbook

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ready realizations which allow to use them easily on their own corpus of texts. Realizations of algorithms TextRank and Rake are available via library Udpipe for language R [35], while realization of algorithm Yake! is available via special API [36]. These algorithms were already compared with each other in [8] the authors of which describe realization of algorithm Yake! and demonstrate its efficiency in comparison with 10 methods. With this aim, they use 20 datasets which are benchmarks in extraction of keywords.

Table 4:The Top 10 Keywords Extracted by Yake!

Rank	Keyword
1	figure
2	documents
3	section
4	query
5	query terms
6	term
7	terms
8	web page
9	web search engines
10	web search

We present below the tables demonstrating top 10 keywords extracted from the considered by us corpus by the mentioned baseline algorithms. There are definite features how each of the algorithms returns the result. For instance, algorithm Yake! (see Table 4) returns just a ranked list of keywords without indicating the scores, therefore, in order to use these keywords for evaluation of the method, every time we cut off the necessary for us number of words. As is seen in the table, the algorithm can return one and the same word both in singular and plural form (compare keywords "term" and "terms").

Algorithm TextRank returns the ranked list of keywords, too, but truncated according to the set up proportion. By default, such proportion is equal to 1/3, i.e. the total list of keywords includes only the upper third of all extracted keywords. Table 5 contains top 10 keywords from this total list.

Table 5: The Top 10 Keywords Extracted by TextRank

Rank	Keyword	Frequency
1	document	1443
3	term	825
4	query	762
6	page	607
7	set	481
8	section	457
9	model	441
10	number	434

Algorithm Rake, apart from the list of extracted keywords, for each keyword returns also its Rake score. Therefore, the list of keywords can be ranked either according to Rake score (see Table 6), or the frequency (see Table 7), or by the product of these values (see Table 8). In our experiments which are described below, the greatest efficiency of algorithm Rake was achieved when we used ranking of keywords according to the product of Rake score and frequency. One can observe this tendency visually when comparing Tables 6-8.

Table 6:The Top 10 Keywords Extracted by Rake (Keywords are Ranked by Rake Score)

Rank	Keyword	Frequency	Rake	Frequency * Rake
1	cheap CD	1	12.000	12.000
2	red wine	1	11.333	11.333
3	query Doc	1	11.049	11.049
4	Doc FP	1	10.071	10.071
5	ad hoc	2	8.798	17.596
6	computational linguistic	1	8.455	8.455
7	US patent	1	8.167	8.167
8	false negative	1	8.084	8.084
9	greater concurrency	1	8.077	8.077
10	false positive	2	7.919	15.838

Table 7: The Top 10 Keywords Extracted by Rake (Keywords are Ranked by Their Frequencies)

Rank	Keyword	Frequency	Rake	Frequency*Rake
1	document	1336	0.302	403.662
2	term	719	0.370	266.016
3	Query	546	0.412	225.094
4	section	444	0.037	16.373
5	number	420	0.039	16.376
6	page	408	0.414	169.047
7	example	368	0.132	48.654
8	cluster	321	0.207	66.302
9	class	306	0.306	93.689
10	index	262	0.376	98.501

*Table 7: The Top 10 Keywords Extracted by Rake (Keywords are Ranked by the Product Frequency*Rake)*

Rank	Keyword	Frequency	Rake	Frequency*Rake
1	document	1336	0.302	403.662
2	posting list	59	4.760	280.864
3	term	719	0.370	266.016
4	query	546	0.412	225.094
5	web page	47	4.050	190.365
6	query term	45	4.056	182.534
7	training set	35	4.869	170.404
8	page	408	0.414	169.047
9	model	239	0.645	154.211
10	search engine	26	5.262	136.809

4.3 Evaluation Metrics

Owing to the presence of the standard copyright list of keywords we can evaluate the accuracy and completeness of the considered methods of extraction of keywords. For this, it is necessary to calculate the values as shown in Table 9.

Table 8: Reference Values for Calculating the Accuracy and Completeness of Keyword Extraction

Notation	Name	Defined as
TP	True Positive	the number of candidate keywords that are present in the reference list
FP	False Positive	the number of candidate keywords that are not present in the reference list
FN	False Negative	the number of keywords of the reference list that are missed among the extracted candidate keywords

Then the accuracy and completeness can be evaluated by the following formulae:

$$Precision = \frac{TP}{TP+FP} \quad (6)$$

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The obtained values of the accuracy and completeness of keyword extraction can be combined into an index called F-score with the help of the harmonic mean:

$$F1 = 2\frac{Precision*Recall}{Precision+Recall} \quad (8)$$

4.4 Results

The results of extraction of keywords from the book "Introduction to Information Retrieval" using the above described baseline methods are presented in Tables 10-12.

Table 9: Results of TextRank Algorithm

Number of candidates	Number of extracted keywords	Precision	Recall	F1
600	100	0.1667	0.1698	0.1682
700	107	0.1529	0.1817	0.1661
800	116	0.1450	0.1969	0.167
900	119	0.1322	0.2020	0.1598
1000	123	0.1230	0.2088	0.1548

As can be seen in the tables, among the baseline method, the method TextRank was most effective on the proposed corpus. This contradicts the results of work [8] the authors of which prove on 20 standard corpora the preference of their algorithm Yake! Over TextRank, Rake and other state-of-the-art methods. The thing is that our experiments were carried out on a large corpus of texts (a book) and the keywords were extracted not for separate documents (sections of the book) but for the whole book. That is stop 3.4 was omitted as we did not have marked keywords for each document in the corpus.

Table 10:Results of Rake Algorithm

Number of candidates	Number of extracted keywords	Precision	Recall	F1
600	85	0.1417	0.1443	0.1430
700	95	0.1357	0.1613	0.1474
800	107	0.1338	0.1817	0.1541
900	114	0.1267	0.1935	0.1531
1000	123	0.1230	0.2088	0.1548

Table 11:Results of Yake! Algorithm

Number of candidates	Number of extracted keywords	Precision	Recall	F1
600	81	0.1350	0.1375	0.1362
700	88	0.1257	0.1494	0.1365
800	94	0.1175	0.1596	0.1354
900	100	0.1111	0.1698	0.1343
1000	103	0.1030	0.1749	0.1296

As was mentioned in section 3.4, such a statement of the task for keyword extraction is not quite traditional but occurs in literature. For example, in [37] the authors proposed an approach for extracting keywords from a large literature corpus and as a standard collection they used Charles Darwin's scientific work "The Origin of Species". A glossary compiled by a reputable expert and consisting of 283 keywords was used as a standard list of keywords to evaluate the efficiency of the proposed method.

Tables 13-15 present the results of our approach at different values of factorization degree k (i.e. at different numbers of topics being modelled). Apart from parameter k, we used one more parameter T with the help of which we controlled the dimensions of the documents-term matrix. We limited the number of columns in this matrix by only those terms which occurred in the corpus not less than three times.

Thus, the factorized matrix documents-by-terms contained 245 rows-documents and 2552 terms (unigrams, bigrams and trigrams). As is seen in Tables 13-15, our approach is most effective at parameter k=3. This can be explained by the fact that the less the number of modelled topics, the lower is their overlapping. It is low overlapping that we try to achieve in our work using non-smooth NMF instead of classic NMF.

Table 12: Results of Proposed Non-Smooth NMF-Based Algorithm (k=3)

Number of candidates	Number of extracted keywords	Precision	Recall	F1
600	110	0.1833	0.1868	0.185
700	117	0.1671	0.1986	0.1815

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

800	130	0.1625	0.2207	0.1872
900	139	0.1544	0.236	0.1867
1000	146	0.146	0.2479	0.1838

Table 13: Results of Proposed Non-Smooth NMF-Based Algorithm (k=4)

Number of candidates	Number of extracted keywords	Precision	Recall	F1
600	104	0.1733	0.1766	0.1749
700	111	0.1588	0.1885	0.1724
800	122	0.1546	0.2071	0.1770
900	131	0.1467	0.2224	0.1768
1000	140	0.1410	0.2377	0.1770

Table 14: Results of Proposed Non-Smooth NMF-Based Algorithm (k=5)

Number of candidates -	Number of extracted keywords	Precision	Recall	F1
600	99	0.165	0.1681	0.1665
700	111	0.1586	0.1885	0.1723
800	118	0.1475	0.2003	0.1699
900	128	0.1422	0.2173	0.1719
1000	143	0.1430	0.2428	0.1800

5. CONCLUSION

In this work, we showed that nonnegative matrix factorization has its own place in the class of unsupervised keyword extraction. This is the niche of the methods directed to extraction of keywords from large literature corpora, such as textbooks, scientific works, technical literature, books. Well-known keyword extraction methods such as TextRank, Rake or a new very promising method Yake! being applied to this kind of corpora significantly yield to the nonnegative matrix factorization though they are faster. Despite the fact that we do not present in this work the speed performance assessment of each of the compared algorithms, it is a common knowledge that nonnegative matrix factorization is quite a weighty algorithm which yields in performance to graphical or pure statistical methods of keyword extraction.

Also, the question remains how the proposed by us method will behave on those collections of documents where keywords must be matched separately for each document. This question requires additional consideration in our future researches as in this work, when carrying out experiments, we omitted this step described in detail at the theoretical level in section 3.4. To see how good is the proposed by us method when performing this step in practice, we should use principally other standard collections such as, for example, SemEval 2010 [38] or SemEval 2017 [39] in which a separate list of keywords is given for each text document. We suppose that our method can surpass the considered baselines as, unlike them, it has a principal possibility to annotate separate documents of a collection with not only the keywords that are explicitly present in them but also with the keywords which are absent in them but are deeply related to its content and topic.

One more interesting direction for our future investigations is to indirectly measure, in the absence of universal standard collections, the quality of extracted keywords, for example, by assessing the quality of text classification or clustering [40, 41, 42].

Journal of Theoretical and Applied Information Technology
30th November 2020. Vol.98. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

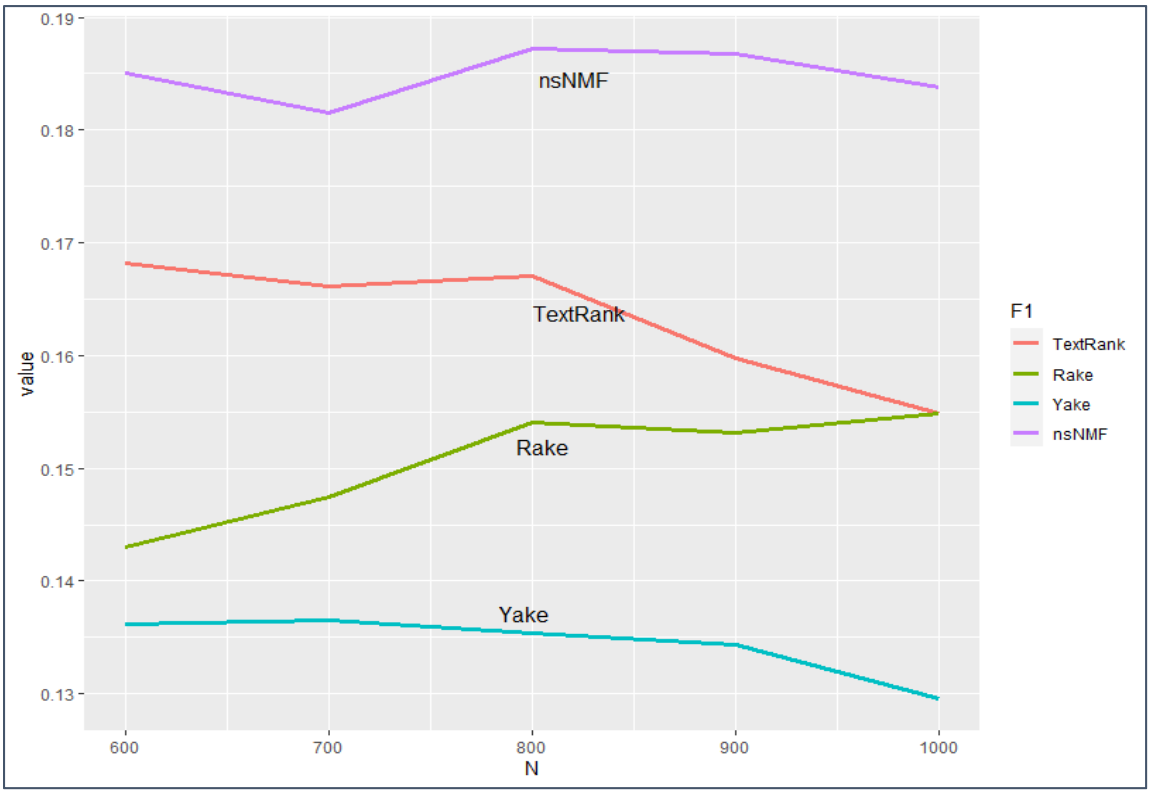

Figure 4: Comparing of the Proposed Approach nsNMF With Baselines (When the Parameter of the Proposed Approach k=3)

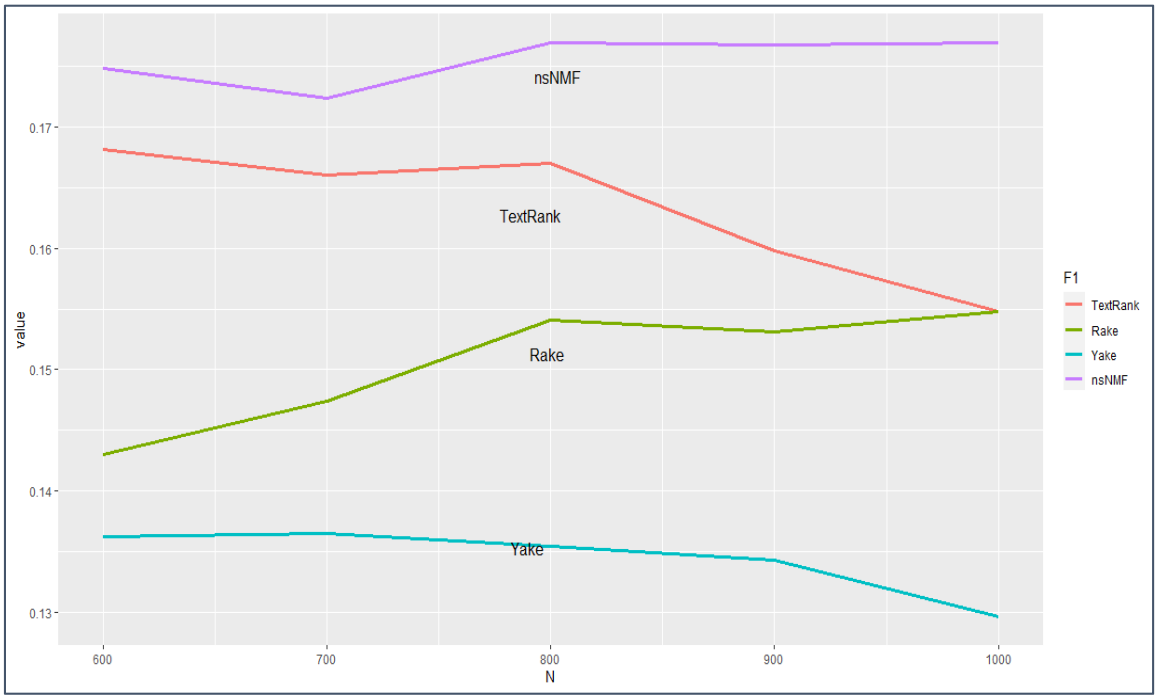

Figure 5: Comparing of the Proposed Approach nsNMF With Baselines (When Parameter of the Proposed Approach k=4)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

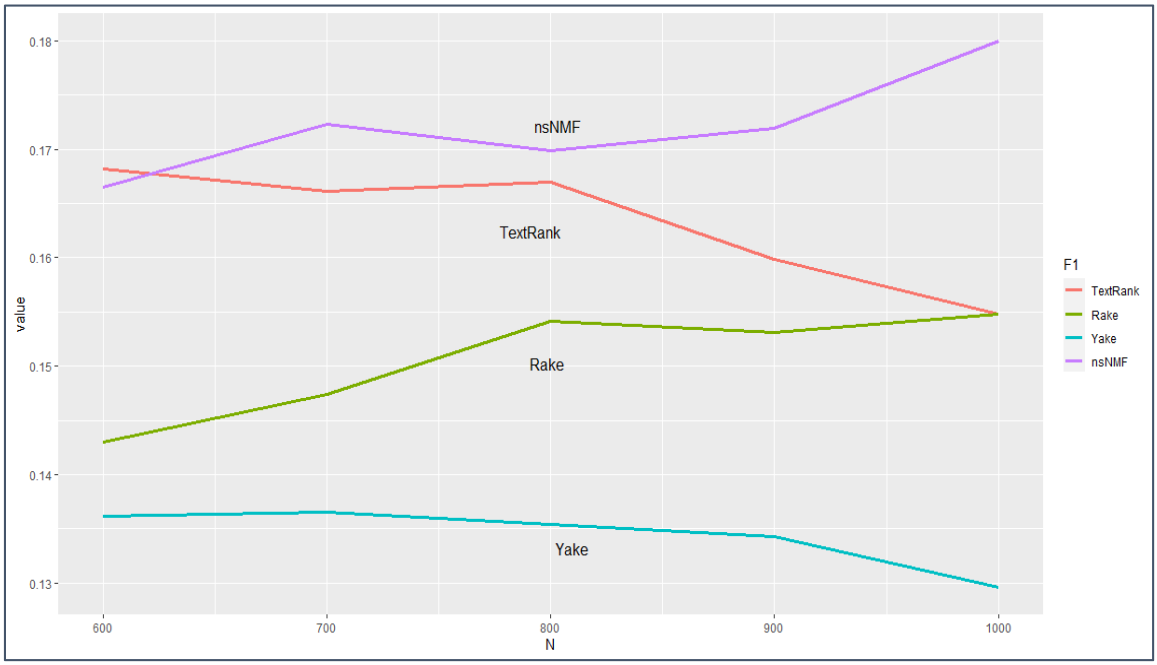

Figure 5:Comparing of the Proposed Approach nsNMF With Baselines (When the Parameter of the Proposed Approach k=5)

ACKNOWLEDGEMENTS

This work was supported in part under grant of Foundation of Ministry of Education and Science of the Republic of Kazakhstan "Creation of high-performance intelligent analysis and decision making technologies for the «logistics-agglomeration» system within formation of the Republic of Kazakhstan digital economics" under project ID no. BR05236340.

REFERENCES:

[1] Biswas S. K., Bordoloi M., Shreya J. A graph based keyword extraction model using collective node weight //Expert Systems with Applications. – 2018. – T. 97. – C. 51-59.

[2] Lahiri S., Choudhury S. R., Caragea C. Keyword and keyphrase extraction using centrality measures on collocation networks //arXiv preprint arXiv:1401.6571. – 2014.

[3] Beliga S. Keyword extraction: a review of methods and approaches //University of Rijeka, Department of Informatics, Rijeka. – 2014. – C. 1-9.

[4] Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: EMNLP 2004, pp. 404–411 (2004).

[5] Rose S. et al. Automatic keyword extraction from individual documents //Text mining: applications and theory. – 2010. – Vol. 1. – Pp. 1-20.

[6] Lynn H. M. et al. Swiftrank: an unsupervised statistical approach of keyword and salient sentence extraction for individual documents //Procedia computer science. – 2017. – Vol. 113. – C. 472-477.

[7] Campos R. et al. A text feature based automatic keyword extraction method for single documents //European Conference on Information Retrieval. – Springer, Cham, 2018. – C. 684-691.

[8] Campos R. et al. YAKE! Keyword extraction from single documents using multiple local features //Information Sciences. – 2020. – T. 509. – C. 257-289.

[9] Pay T. Totally automated keyword extraction //2016 IEEE international conference on big data (Big Data). – IEEE, 2016. – C. 3859-3863.

[10] Matsuo Y., Ishizuka M. Keyword extraction from a single document using word co-occurrence statistical information //International Journal on Artificial Intelligence Tools. – 2004. – T. 13. – №. 01. – C. 157-169.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[11] Kim S. N. et al. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles //Proceedings of the 5th International Workshop on Semantic Evaluation. – 2010. – С. 21-26.
[12] Lee D. D., Seung H. S. Learning the parts of objects by non-negative matrix factorization //Nature. – 1999. – Т. 401. – №. 6755. – С. 788-791.
[13] Papagiannopoulou E., Tsoumakas G. A review of keyphrase extraction //Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. – 2020. – Т. 10. – №. 2. – С. e1339.
[14] Salton G., Yang C. S., Yu C. T. Contribution to the theory of indexing. – Cornell University, 1973.
[15] El-Beltagy S. R., Rafea A. KP-Miner: A keyphrase extraction system for English and Arabic documents //Information Systems. – 2009. – Т. 34. – №. 1. – С. 132-144.
[16] Wan X., Xiao J. Single Document Keyphrase Extraction Using Neighborhood Knowledge //AAAI. – 2008. – Т. 8. – С. 855-860.
[17] Florescu C., Caragea C. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents //Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). – 2017. – С. 1105-1115.
[18] Biswas S. K., Bordoloi M., Shreya J. A graph based keyword extraction model using collective node weight //Expert Systems with Applications. – 2018. – Т. 97. – С. 51-59.
[19] Vega-Oliveros D. A. et al. A multi-centrality index for graph-based keyword extraction //Information Processing & Management. – 2019. – Т. 56. – №. 6. – С. 102063.
[20] Duari S., Bhatnagar V. sCAKE: semantic connectivity aware keyword extraction //Information Sciences. – 2019. – Т. 477. – С. 100-117.
[21] Bougouin A., Boudin F., Daille B. TopicRank: Graph-based topic ranking for keyphrase extraction. – 2013.
[22] Liu Z. et al. Automatic keyphrase extraction via topic decomposition //Proceedings of the 2010 conference on empirical methods in natural language processing. – Association for Computational Linguistics, 2010. – С. 366-376.
[23] Sterckx L. et al. Topical word importance for fast keyphrase extraction //Proceedings of the 24th International Conference on World Wide Web. – 2015. – С. 121-122.
[24] Chen J. et al. Diverse topic phrase extraction through latent semantic analysis //Sixth International Conference on Data Mining (ICDM'06). – IEEE, 2006. – С. 834-838.
[25] Süzek T. Ö. Using latent semantic analysis for automated keyword extraction from large document corpora //Turkish Journal of Electrical Engineering & Computer Sciences. – 2017. – Т. 25. – №. 3. – С. 1784-1794.
[26] Bennani-Smires K. et al. Simple unsupervised keyphrase extraction using sentence embeddings //arXiv preprint arXiv:1801.04470. – 2018.
[27] Papagiannopoulou E., Tsoumakas G. Local word vectors guiding keyphrase extraction //Information Processing & Management. – 2018. – Т. 54. – №. 6. – С. 888-902.
[28] Qiu Q. et al. Geoscience keyphrase extraction algorithm using enhanced word embedding //Expert Systems with Applications. – 2019. – Т. 125. – С. 157-169.
[29] Gero Z., Ho J. C. NamedKeys: Unsupervised Keyphrase Extraction for Biomedical Documents //Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. – 2019. – С. 328-337.
[30] Kuang D., Brantingham P. J., Bertozzi A. L. Crime topic modeling //Crime Science. – 2017. – Т. 6. – №. 1. – С. 12.
[31] A. Pascual-Montano, J.M. Carazo, K. Kochi, D. Lehmean, and R. Pacual-Marqui. Nonsmooth nonnegative matrix factorization (nsNMF). IEEE Transactions Pattern Analysis and Machine Intelligence, 28(3):403–415, 2006.
[32] Cichocki A. et al. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. – John Wiley & Sons, 2009.
[33] Firoozeh N. et al. Keyword extraction: Issues and methods //Natural Language Engineering. – 2020. – Vol. 26. – №. 3. – С. 259-291.
[34] Manning C. D., Raghavan P., Schuetze H. Introduction to Information Retrieval. 2008 //Available also from: https://nlp. stanford. edu/IR-book/pdf/irbookonlinereading.pdf.
[35] Wijffels J. Udpipe: Tokenization, parts of speech tagging, lemmatization and dependency parsing with the "UDPipe""NLP" toolkit (R Package Version 0.6)[Computer software]. – 2018.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[36] Campos, R., Mangaravite, V., Pasquali, A., Jatowt, A., Jorge, A., Nunes, C. and Jatowt, A. YAKE! API // Available from: http://yake.inesctec.pt/apidocs/#/

[37] Herrera J. P., Pury P. A. Statistical keyword detection in literary corpora //The European Physical Journal B. – 2008. – T. 63. – №. 1. – C. 135-146.

[38] Kim S. N. et al. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles //Proceedings of the 5th International Workshop on Semantic Evaluation. – 2010. – C. 21-26.

[39] Augenstein I. et al. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications //arXiv preprint arXiv:1704.02853. – 2017.

[40] Onan A., Korukoğlu S., Bulut H. Ensemble of keyword extraction methods and classifiers in text classification //Expert Systems with Applications. – 2016. – T. 57. – C. 232-247.

[41] Tran T. C. T. et al. Text Classification Based on Keywords with Different Thresholds //Proceedings of the 2019 4th International Conference on Intelligent Information Technology. – 2019. – C. 101-106.

[42] Barakhnin V. B., Kozhemyakina O. Y., Pastushkov I. S. Comparative analysis of methods of automated classification of poetic texts based on lexical signs //CEUR Workshop Proceedings. – 2017. – C. 252-257.