KeyVector: Unsupervised Keyphrase Extraction Using Weighted Topic via Semantic Relatedness

Alymzhan Toleu, Gulmira Tolegen, and Rustam Mussabayev

Institute of Information and Computational Technologies
Almaty, Kazakhstan
{alymzhan.toleu, gulmira.tolegen.cs, rmusab}@gmail.com

Abstract. Keyphrase extraction is a task of automatically selecting topical phrases from a document. We present KeyVector, an unsupervised approach with weighted topics via semantic relatedness for keyphrase extraction. Our method relies on various semantic relatedness of documents, topics and keyphrases in the same vector space, which allow us to compute three keyphrase ranking scores: global semantic score, find more important keyphrases for a given document by measuring the semantic relation between documents and keyphrase embeddings; topic weight, pruning/selecting the candidate keyphrases on the topic level; topic inner score, ranking the keyphrases inside each topic. Keyphrases are then generated by ranking the values of combined three scores for each candidate. We conducted experiments on three evaluation data sets of different length documents and domains. Results show that KeyVector outperforms state of the art methods on short, medium and long documents.

Keywords: Keyphrase extraction · Clustering · Topic modeling · Semantic relatedness · Text mining.

1 Introduction

Keyphrase extraction aims to automatically extract keyphrases from a document and ensure the selected keyphrases covey the main topic of the document. Key phrases are an essential component for solving the tasks of information retrival [17, 21, 9, 19], summarization [6], text mining and topic modeling [2]. Word/phrase embeddings are distributed representations of text in an n-dimensional space. In such space, the semantic/syntactic features of words can be captured by the embeddings, and the machine learning algorithms could reach better results in natural language processing (NLP) tasks by grouping words/phrases. Owing to its importance, the embedding becomes necessary for solving many NLP tasks [16, 22, 23] better nowadays.

Many graph and topic-based approaches (TextRank [15], SingleRank [24], TopicRank [4]) for keyphrase extraction have been proposed to use internal and external discrete features such as positional features, word frequency, co-occurrences and some other Wikipedia-based statistical features. Instead of relying on either internal or external discrete features, in this paper, we present

KeyVector, an unsupervised keyphrase extraction method by computing the semantic relatedness of words/phrases through embeddings. Our approach has several advantages over existing state of the arts. 1) Global semantic score. Embedding the sentences, candidate keyphrases into the same vector space, which allows us to compute their semantic relatedness more efficiently than discrete features. Based on the embeddings, we propose to use the global semantic score that is the semantic relatedness between document and keyphrases. 2) Weighted topics. Ranking a large number of candidate keyphrases is often tricky. Intuitively, it is more efficient if we group the candidates into topics by their embeddings, then ranking them on the topic or global level. To do this, we compute representation for each topic after the clustering process and assign a weight for each topic by measuring the semantic relatedness between topic and documents. We show that the technique of weighted topic influences the process of keyphrase selection. 3) For each candidate inside each topic, we propose to compute a local ranking score, and we refer to it as topic inner semantic score.

We use three standard data sets of different document size and domain to evaluate KeyVector. We compare KeyVector to five different state of the art approaches. Experiments show that KeyVector outperforms other baselines on short/medium/long documents. It yields better results for both short and long documents. It indicates that KeyVector has better stability in its performances against the various length of documents compared to other topic-based approaches. The rest of the paper is organized as follows: Section 2 presents the existing methods for the keyphrase extraction task; Section 3 describes the details of KeyVector; Section 4 describes the evaluation process and report the experimental results; Section 5 concludes this work.

2 Related Work

In general, keyphrase extraction methods can be classified into two groups: supervised and unsupervised approaches. In supervised approaches [14, 10], the problem of keyphrase extraction is regarded as a binary classification task and learn models from training data. Many details about the supervised methods and statistical features for keyphrase extraction can be found in the survey [7]. Here, we focus on unsupervised approaches that often have two ways: corpus-dependent and corpus-independent. The former requires all documents to do the extraction of keyphrases, and the TFIDF [20] is the simple, widely used approach contains two features: term-frequency and inverse document frequency. Methods belonging to the latter like TextRank [15], KeyCluster [13] TopicRank [12] and EmbedRank [1], including our proposed method, requires no other documents than the one document from which to extract keyphrases.

TextRank [15] is the well-known graph-based approach, and it builds a graph from one document. Each node of the graph corresponds to candidate keyphrases and the edge connects two candidates. For each node, calculate the score from other nodes connected by the edges. The top-ranked nodes from the graph are then selected as keyphrases. KeyCluster [13] is the clustering-based approach

that clustering semantically similar candidates using statistical features such as word co-occurrences and positional features etc. The main idea of this method is that a candidate is to be selected as keyphrase if the candidate close to the centroid of a cluster. The clusterized candidates can be viewed as topics that a document covers. The drawback of this method is that those unimportant topics in the document could be selected as keyphrases, which is limiting the quality of the resulting sets of keyphrase. TopicRank[12] was proposed to overcome the weakness of the KeyCluster. In order to ensure that extracted keyphrases cover the main topics, this method uses Latent Dirichlet Allocation [2] to generate topics for a document and uses TextRank multiple times for a document and once for the generated topics.

EmbedRank [1] is an embedding-based approach that computes the document embedding and the embedding of each candidate phrase separately. The embeddings are obtained from the popular Doc2vec [11] and Sent2vec [18] models. The top keyphrases are selected by ranking the candidate phrases according to their cosine distance to the document embedding. EmbedRank is comparable to KeyVector, but they are different in several points: 1) the global semantic score is used to compute the semantic relation between sentences and keyphrases. 2) weighted topics are applied to do global pruning for the candidates that unlikely to be keyphrase. 3) topic inner score is used to rank the keyphrases in each topic.

3 KeyVector: Automatic Keyphrase Extraction

In this section, we introduce KeyVector, a novel weighted topic keyphrase extraction method via semantic relatedness, which is designed to handle the problematic situation of the task when each of the documents has a large number of candidate keyphrases to rank. Before describing the method, let us clarify the elements described in the following sections. Keyphrase, it is made up of one or multiple words. Candidate keyphrase, it is extracted for each document by using heuristic rules (Section 4.2) and each document has a large number of candidates. Gold Keyphrases, they are given by the annotators or authors of data sets. Topic, it consists of a set of candidate keyphrases and each document contains several topics. It also treated as cluster/group.

The method consists of three main steps: 1) Project each of sentences and candidate keyphrases into the same and high dimensional space to compute their semantic relatedness. 2) Compute the weights to topics by clustering the candidate keyphrases of each document. 3) Obtaining the ranking scores by measuring the semantic relatedness between the candidate with the sentences, and the inner semantic score of each candidate in a topic plus topic weights.

The global architecture of KeyVector is given in Fig 1. The process of keyphrase extraction is from words (w) to sentence (s), then to keyphrase (p). The edge arrows between words and sentences mean that the embeddings of each sentence are computed by averaging each word embeddings. The edge arrows between sentences and candidate mean the computation of the semantic relatedness between them.

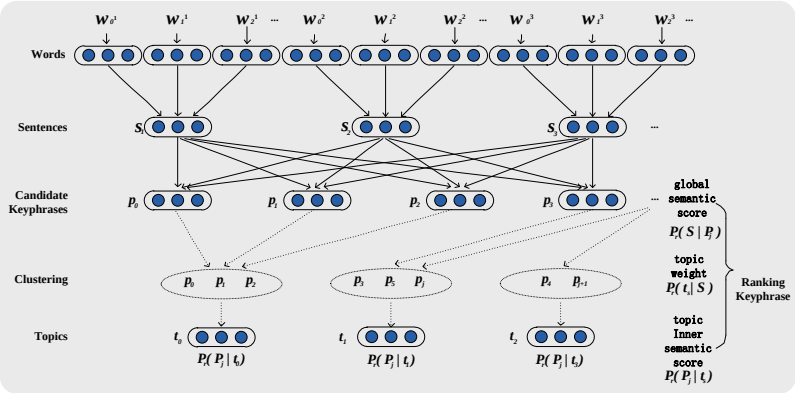


Fig. 1: The architecture of KeyVector.

3.1 Embedding the Sentences and Phrases

Representing text such as words, sentences and documents into vector representation, which allows the model to capture the semantic relatedness via word/phrase vectors within the shared high-dimensional vector space. We use this property to rank the candidate keyphrases, which allow us to partially capture the semantics between text and candidate keyphrase to meet the informativeness of keyphrase.

We represent each sentence and candidate phrase into vector representation by using word embeddings [16]. More formally, for a given collection of documents $d \in D$, we segment the sentences $S \in d$, $s_i \in S$ and tokenize them into words $W \in S$, $w_l \in W$. With the purpose of putting the method as simple as possible, we compute the sentence embedding by using the averaged vector sum of each word in the sentence.

$$\mathbf{s_i} = \frac{1}{|s_i|} \sum_{l=1}^{|s_i|} \mathbf{w}_l \tag{1}$$

where $|s_i|$ is the number of word in the sentence s_i. For simplicity, the notations within boldface denote vectors/matrix. The obtained sentence embedding is $\mathbf{s_i} \in \mathbb{R}^{|1 \times N|}$. N is the dimension. Note, the word embeddings are used only for the equation (1), in other cases we use keyphrase and sentence embeddings.

The process of computing the embedding for each candidate keyphrase $p_j \in P$ is the same with sentence embedding. Each p_j consists of words $w_l \in p_j$. The generation of embedding for word sequences in this model is simple enough and feasible. This embedding method also allows us to embed arbitrary-length sequences of words. In order to compute both sentences and phrases embedding, we employ publicly available pre-trained word embeddings[1], which allow both types of embeddings in a shared semantic vector space.

[1] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

3.2 Weighted Topics

Given the sentences, the candidate keyphrases and their embeddings, we compute the semantic relatedness of them in order to cluster similar candidates by their meanings. The idea behind the clustering the candidates is to handle the problematic situation of keyphrase extraction when a large number of candidates extracted from each document. To demonstrate this situation comprehensively, let us consider the numbers of extracted candidates per document from three standard data sets: **Inspec** [8], **DUC** [24] and **NUS** [17]. Figure 2 shows the distribution of the number of generated candidates and we could observe that the length of the input document as longer the document yield more candidate keyphrases. For NUS data set (the rightmost part of the curve), it can be seen that the extracted candidate keyphrases are more than 1000 and for DUC (the middle part of the curve) the number is about 200. The large number of candidates for each documents leads the selection process become very tricky.

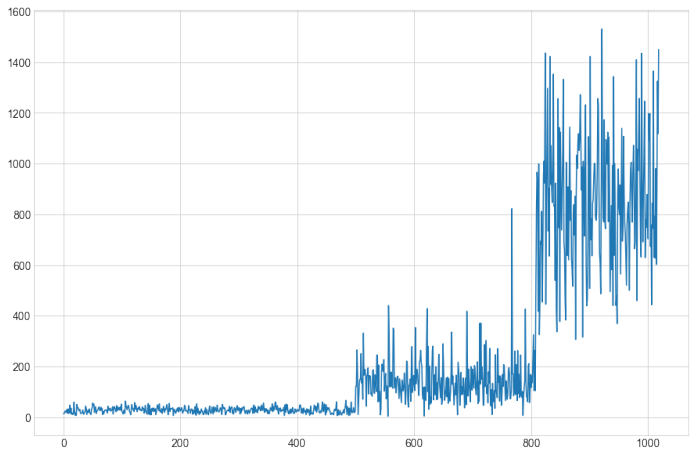


Fig. 2: The distribution of the number of generated candidates. The x-axis is the combined document ID of three data sets in the following order: Inspec (short), DUC (medium), NUS (long), and the y-axis is the number of extracted candidates.

The technique of weighted topics plays the role like pruning candidates that are unlikely to be keyphrases on a global level. Existing topic-based methods (KeyCluster [13], TopicRank [4]) apply statistical features (such as word co-occurrences, the number of overlapping words and positional features etc.) to compute the semantic relations, then group the candidates. The group of candidates can be treated as topics [4]. Here, we group the similar candidates P based on their embeddings' relatedness in the same space with the input document d_k. We apply affinity propagation method [5] to cluster the candidate in

terms of a given document, and it also automatically identifies how many clusters there are in each document. Once the candidates P are clusterized into topics, $T = \{t_1, ..., t_s, ..., t_{|T|}\}$ (t_s is the *s-th* topic that contains one or more candidates and $\mathbf{t_s}$ is its embedding), again, we compute vector representation for each topic by averaging the sum of the embeddings of each candidate inside the topic. We treat topic representation $\mathbf{t_s}$ as the centroid of each topic.

To weighting the topics, we calculate the semantic relatedness between the centroid and the input document.

$$Pr(t_s|s_i) = \sum(\sum_{i=0}^{|S_k|} cos(\mathbf{s}_i, \mathbf{t_s})) \tag{2}$$

where $Pr(t_s|s_i)$ is the weight for topic t_s. $\mathbf{S}_k \in \mathbb{R}^{|S_k| \times N}$ is the matrix representation of the document d_k and each sentence's embedding of d_k is $\mathbf{s_i} \in \mathbb{R}^{1 \times N}$. Note that the weights for topics are computed using embeddings of sentences and topic centroids, and they are in the same vector space.

3.3 Keyphrase Selection

Based on the above, we select the top-M keyphrases by three strategies: 1) **global semantic score**, the semantic relatedness between sentences and candidates, $Pr(S|p_j)$. The underlying hypothesis of this measurement is that a sentence is more important if it contains more important keyphrases, and a candidate keyphrase is important if it is related to a large number of sentences. 2) **topic weights**, $Pr(t_s|S)$. It is used for topic importance determination, and it is a global pruning technique for the candidates that are not likely to be keyphrases. 3) **topic inner semantic score**, the semantic distances between each candidate to the centroid, $Pr(p_j|t_s)$. It is an inner selection in each topic to the candidates. Given a document d or its sentences S, we compute a score for j-th candidate keyphrase by computing its likelihood.

$$Pr(p_j|S) = Pr(t_s|S)Pr(p_j|t_s)Pr(S|p_j) \tag{3}$$

where $Pr(t_s|S) = \sum(\sum_{i=0}^{|S_k|} cos(\mathbf{s}_i, \mathbf{p_j}))$ is the probability of the sentences given the keyphrase. It means that which keyphrases produce larger probabilities for sentences, it could be the keyphrases. $Pr(p_j|t_s) = cos(\mathbf{p_j}, \mathbf{t_s})$, $p_j \in t_s$ is topic inner score. It means that the most important candidate should be close to the centroid of the topic. $Pr(t_s|S)$ is weighted score for topic t_s. Doing the above calculation, the ranking scores for each candidate can be computed. Then, according to the ranking scores, we can suggest top-M ranked candidates as the keyphrase.

4 Experiments

4.1 Data set

KeyVector is evaluated on three publicly available data sets[2]. Table 1 shows the statistics about data set[3]. The data set of **Inspec** [8] consists of 2 000 short documents from scientific journal abstracts. We evaluate our model on 500 documents of test set. The **DUC-2001** data set [24] contains 308 medium length newspaper articles from TREC-9. The **NUS** [17] consists of 211 long length of scientific article. Each document contains several sets of keyphrases. One is created by author and the others are assigned by annotators. Following [17], we evaluate on the union of all sets of author's and annotators' keyphrases.

Table 1: Statistics of the data sets.

dataset	types	#docs.	#avg. tok.	#avg. c.	#kps	%miss. kp.	%miss. w.	%miss. c.
Inspec	short	500	134.12	26.60	4913	41.66	17.5	6.78
DUC	medium	211	847.23	142.89	2488	13.46	24.76	2.74
NUS	long	308	7379.19	854.32	2317	37.16	20.13	2.21

4.2 Preprocessing

The preprocessing has impacts on the performance of keyphrase extraction models [3]. In the experiments, we used preprocessed version of Inspec [8][4] and DUC-2001 [24][5] data set that are publicly available. We applied following preprocessing to NUS [17] data set, namely, sentence segmentation, word tokenization and POS tagging (*nltk pos-tagger*). Then, we extracted candidate phrases that consist of zero or more adjective followed by one or multiple nouns. The stopwords were filtered out from the data sets and the stemming is not performed at preprocessing stage.

4.3 Results

To evaluate our approach, we designed several set of experiments: one of them is to compare KeyVector with other baselines; another one is to evaluate the performance of all models concerning the number of top-M.

[2] https://github.com/snkim/AutomaticKeyphraseExtraction

[3] The columns of Table 1 are: #docs - the number of the documents; #avg. tok. - average number of tokens per document; #avg. cand - average number of candidates; #kps - total number of keyphrases; # miss. kp. - percentage of keyphrases not present in candidates; #miss. w. - percentage of words out of vocabulary of embeddings; #miss. c. - percentage of candidates that have embedding with value 0.

[4] https://github.com/boudinfl/hulth-2003-pre

[5] https://github.com/boudinfl/duc-2001-pre

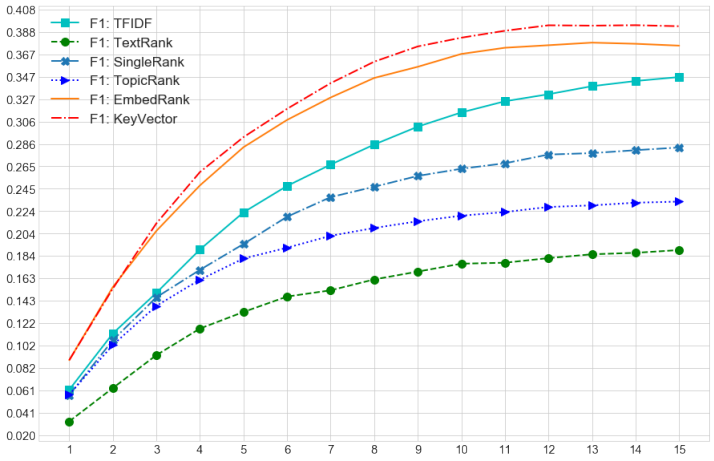
Table 2 shows the results of KeyVector and other five baselines. Overall, KeyVector outperforms TFIDF, TextRank, SingleRank and EmbedRank in terms of precision, recall, and F1 score. On Inspec, which contains short documents, KeyVector outperforms all other competing approaches. In this case, TopicRank fails to do better than other baselines of non-topic ranked methods like TFIDF, SingleRank and EmbedRank. However, from the results of KeyVector on short documents, it seems that KeyVector performs well and stable. On Duc, the medium documents, KeyVector also shows the case with Inspec. We could observe that KeyVector has approximately 11.95% (on Inspec), 6.62% (on Duc), 2.53% (on NUS) and 16.22% (on Inspec), 7.32% (on Duc) , 0.99% (on NUS) improvements in F1-score compared with SingleRank and TopicRank, respectively.

Table 2: Comparison of KeyVector with state of the art on the three data sets. The W is the window size and M is the number of selected keyphrases, and the N is the dimension of word embedding.

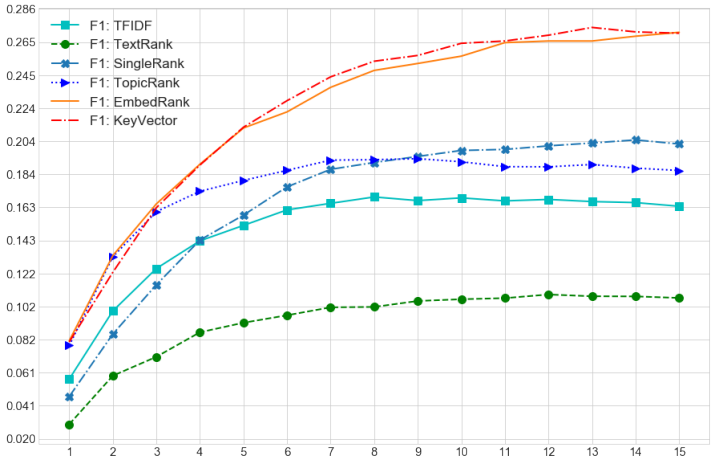
	Methods	Parameter	Precison	Recall	F1-score
Inspec	TFIDF	M=10	31.31	31.56	31.44
	TextRank	W=2, T= 0.3	17.80	17.52	17.66
	SingleRank	W=10, M=10	26.14	26.48	26.30
	TopicRank	W=10, M=10	22.00	22.06	**22.03**
	EmbedRank	N=300, M=10	36.62	36.92	**36.77**
	KeyVector	N=300, M=10	**38.09**	**38.40**	**38.25**
DUC	TFIDF	M=10	15.32	18.86	16.91
	TextRank	W=2, T= 0.3	9.66	11.86	10.65
	SingleRank	W=10, M=10	17.96	22.14	19.83
	TopicRank	W=10, M=10	**17.34**	**21.34**	**19.13**
	EmbedRank	N=300, M=10	23.23	28.64	25.65
	KeyVector	N=300, M=10	**23.95**	**29.52**	**26.45**
NUS	TFIDF	M=10	9.19	8.37	**8.76**
	TextRank	W=2, T= 0.3	5.02	4.57	4.78
	SingleRank	W=10, M=10	8.0	7.29	7.63
	TopicRank	W=10, M=10	9.62	8.76	**9.17**
	EmbedRank	N=300, M=10	8.24	7.50	7.86
	KeyVector	N=300, M=10	10.66	9.71	**10.16**

The results of KeyVector are competitive with TopicRank, and It has improvements about 0.99%, 2.53% and 5.38% compared to TopicRank, SingleRank and TextRank.

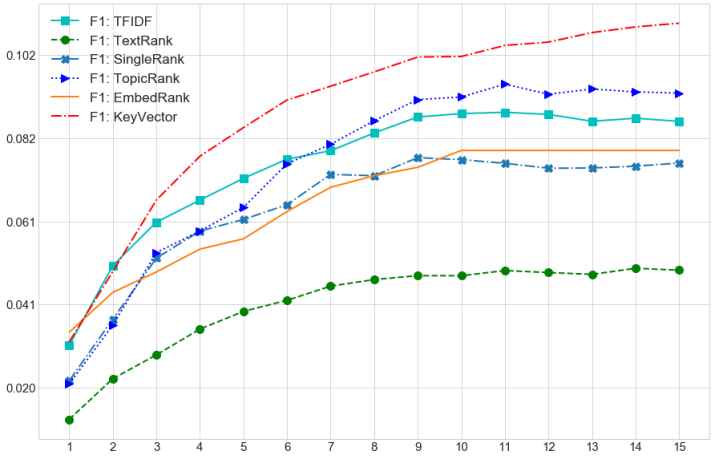
We investigate the effects of the top-M selected keyphrases with respect to F1-score in Figure 3. Figure 3a shows that KeyVector outperforms all baselines start from top-$M = 3$ and it grows continuously compared to EmbedRank that start to drop slightly when top-M=14. On DUC, the results of F1-score KeyVector and EmbedRank are comparable when increasing the number of top-M. It



(a) Inspec



(b) DUC



(c) NUS

Fig. 3: Comparison of F1-score for three data sets concerning the top-M keyphrases.

can be seen that KeyVector has significant improvements compared to others. On NUS, TopicRank shows its advantages for long documents compared to others (TextRank, SingleRank) and the TFIDF also gives excellent results. KeyVector has a 0.99% improvement approximately in F1-score compare to TopicRank. KeyVector computes semantic relatedness from sentences, topics and keyphrases' embeddings to do the extraction and it is sensitive for the way of generating the embeddings for sentences, topics and keyphrases including the different use of normalization/average for embeddings. Take into account the fact that some words are missing from the embeddings (Inspec: 17.5%, DUC: 24.76%, NUS: 20.13%); consequently, some of keyphrases have all zero value in representations (Inspec: 6.78%, DUC: 2.74%, NUS: 2.21%). We believe the improved representation for our method or decreasing the missing percentages of those words/phrases have effects on improving the results. Over all, from the results on short/medium/long documents, we could observe that KeyVector does not fail to do better (like topicRank does) on short/medium documents, and its performances on long documents are also stable. So we deduce that KeyVector has taken the balances on its performance when extracting keyphrases from various lengths of documents.

5 Conclusion

In this paper, we present KeyVector, an unsupervised method for keyphrase extraction. Our approach offers several advantages over existing keyphrase extraction methods. First, the semantic relatedness between sentences and candidates are computed through embeddings which are projected into the same high-dimensional space. The use of weighted topics captures those unimportant topics to reach the goal of pruning the candidates not likely to be keyphrases on the topic level. The topic inner semantic score is another strategy to rank the candidate inside the topic by the semantic distances between each candidate to the topic centroid. We conducted experiments on three standard evaluation data sets of different document sizes and domains. Results show that KeyVector outperforms other baselines on short/medium/long documents.

We will explore the following two points as future work: 1) analyze the effects of different clustering results for keyphrase extraction, and investigate other new clustering algorithms. 2) explore an new strategy for computing the topic inner scores.

Acknowledgments

This research has been conducted within the framework of the grant num. BR05236839 " Development of information technologies and systems for stimulation of personalitys sustainable development as one of the bases of development of digital Kazakhstan ".

References

1. Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., Jaggi, M.: Simple unsupervised keyphrase extraction using sentence embeddings. In: CoNLL (2018)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (Mar 2003)
3. Boudin, F., Mougard, H., Cram, D.: How document pre-processing affects keyphrase extraction performance. In: Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT). pp. 121–128. The COLING 2016 Organizing Committee, Osaka, Japan (December 2016)
4. Bougouin, A., Boudin, F., Daille, B.: Topicrank: Graph-based topic ranking for keyphrase extraction. In: Proceedings of the Sixth International Joint Conference on Natural Language Processing. pp. 543–551. Asian Federation of Natural Language Processing (2013)
5. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**, 2007 (2007)
6. Han, J., Kim, T., Choi, J.: Web document clustering by using automatic keyphrase extraction. In: 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops. pp. 56–59 (2007)
7. Hasan, K.S., Ng, V.: In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1262–1273. Association for Computational Linguistics, Baltimore, Maryland (June 2014)
8. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing. pp. 216–223. EMNLP '03, Association for Computational Linguistics, Stroudsburg, PA, USA (2003)
9. Khairova, N., Petrasova, S., Lewoniewski, W., Mamyrbayev, O., Kuralai, M.: Automatic extraction of synonymous collocation pairs from a text corpus. In: Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018. pp. 485–488 (2018). https://doi.org/10.15439/2018F186
10. Kim, S.N., Kan, M.Y.: Re-examining automatic keyphrase extraction approaches in scientific articles. In: Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications. pp. 9–16. MWE '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
11. Lau, J.H., Baldwin, T.: An empirical evaluation of doc2vec with practical insights into document embedding generation. In: Proceedings of the 1st Workshop on Representation Learning for NLP. pp. 78–86. Association for Computational Linguistics (2016)
12. Liu, Z., Huang, W., Zheng, Y., Sun, M.: Automatic keyphrase extraction via topic decomposition. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. pp. 366–376. EMNLP '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
13. Liu, Z., Li, P., Zheng, Y., Sun, M.: Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. pp. 257–266. EMNLP '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
14. Lopez, P., Romary, L.: HUMB: Automatic Key Term Extraction from Scientific Articles in GROBID. In: SemEval 2010 Workshop. p. 4 p. ACL SigLex event, Uppsala, Sweden (Jul 2010)

15. Mihalcea, R., Tarau, P.: TextRank: Bringing order into texts. In: Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing (July 2004)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
17. Nguyen, T.D., Kan, M.Y.: Keyphrase extraction in scientific publications. In: Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers. pp. 317–326. ICADL'07, Springer-Verlag, Berlin, Heidelberg (2007)
18. Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised learning of sentence embeddings using compositional n-gram features. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 528–540. Association for Computational Linguistics (2018)
19. Petrasova, S., Khairova, N., Lewoniewski, W., Mamyrbayev, O., Mukhsina, K.: Similar text fragments extraction for identifying common wikipedia communities. Data **3**(4) (2018). https://doi.org/10.3390/data3040066
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (Aug 1988)
21. Tolegen, G., Toleu, A., Zheng, X.: Named entity recognition for kazakh using conditional random fields. In: Proceedings of the 4-th International Conference on Computer Processing of Turkic Languages TurkLang 2016. pp. 118–127. Izvestija KGTU im.I.Razzakova (2016)
22. Toleu, A., Tolegen, G., Makazhanov, A.: Character-aware neural morphological disambiguation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 666–671. Association for Computational Linguistics, Vancouver, Canada (July 2017)
23. Toleu, A., Tolegen, G., Makazhanov, A.: Character-based deep learning models for token and sentence segmentation. In: Conference: 5th International Conference on Turkic Languages Processing (TurkLang 2017). Kazan, Tatarstan, Russian Federation (October 2017)
24. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI (2008)