ISSN 2007-9737

Unsupervised Keyphrase Extraction: Ranking Step and Single-Word Phrase Problem

Svetlana Popova[1,*], Vera Danilova[2], Mikhail Alexandrov[3,4], John Cardiff[1]

[1] Technical University of Dublin,
Ireland

[2] Uppsala University,
Sweden

[3] Russian Academy of National Economy and Public Administration, Moscow,
Russia

[4] FRUCT Association, Helsinki,
Finland

spbu.svp@gmail.com, vera.danilova@lingfil.uu.se,
malexandrov@mail.ru, john.cardiff@tudublin.ie

Abstract. Keyphrases provide a compact representation of a document‘s content and can be efficiently used to enhance Web search results and improve natural language processing tasks. This paper extends the state-of-the-art in unsupervised keyphrase extraction from scientific abstracts. We aim to demonstrate the difference between two types of datasets used in the keyphrase extraction domain: datasets where keyphrases for each text are manually assigned by readers, and datasets where keyphrases are assigned by the authors themselves. We aim to highlight the problem of single-word phrases and illustrate the role of this problem for each dataset type. Additionally, we noticed that well-known algorithms in the domain can be divided into two groups. Algorithms in the first group minimize the number of single-word phrases in the set of the extracted keyphrases. In contrast, algorithms in the second group allow the extraction of a larger number of single-word keyphrases. This property of algorithms ”to extract few or many single-word keyphrases” determines how they perform on each type of dataset. We explain the reasons for this.

Keywords. Unsupervised keyphrase extraction, single-word phrase problem, keyphrase length.

1 Introduction

Keyphrases provide a compact representation of a document’s content [32] and automatic keyphrase extraction (**KE**) concerns “the automatic selection of important and topical phrases from the body of a document” [36]. We distinguish KE from keyphrase assignment and keyphrase generation:

1. Assignment: Keyphrases are assigned to a text from a predefined controlled vocabulary.

2. Generation: The extracted phrases do not necessarily appear in a given document during keyphrase generation.

In KE, the extracted keyphrases should occur in the document and are not restricted by a predefined vocabulary. In Table 1 keyphrases are emphasized in bold for the following example abstract. Good keyphrases should satisfy the following properties [26]: Meaningfulness, relevance, and good coverage.

Being more descriptive than single keywords, keyphrases help to perform efficient text mining and are involved in improving the functionality

Table 1. Scientific abstracts and keyphrases (INSPEC dataset)

Accelerated simulation of the steady-state availability of non-Markovian systems. A **general accelerated simulation method** for evaluation of the **steady-state availability** of **non-Markovian systems** is proposed. It is applied to the investigation of a class of systems with repair. **Numerical examples** are given.

of information retrieval systems [17, 3, 39]. KE methods are divided into supervised and unsupervised. Unsupervised methods can be straightforwardly applied to documents and do not require training data.

Our research entirely focuses on unsupervised approaches. Also, we constrain the domain to titles and abstracts of scientific publications. KE from this type of text has been the focus of researchers for a long time due to the development of electronic libraries and e-learning systems. KE methods can be divided into word-based and candidate-based.

- **Word-based** methods consist of two stages: (1) document terms are weighed and then ranked to select words that belong to keyphrases and, (2) keyphrases are built from selected terms. Often it means merging terms that follow each other in a given text.
- **Candidate-based** methods include two stages: (1) keyphrase candidates extraction, and (2) keyphrase selection based on the ranking of the candidates.

Most unsupervised KE algorithms are candidate-based. We only explore candidate-based approaches in this research.

KE methods have traditionally been evaluated and compared to other methods based on the F1-score [21], which has become the standard and consistently used evaluation approach in the KE domain. Whenever we refer to the performance of the KE algorithm, its quality, or its evaluation score, we mean the evaluation of its performance using the macro-average F1-score. This score will be discussed below.

While researchers have dedicated efforts to enhance the final KE evaluation scores produced by previous algorithms, there appears to be a gap in addressing the underlying causes for variations in ranking quality observed among these algorithms. To the best of our knowledge, the factors contributing to these differences in ranking performance have not been deeply explored yet. This study partially deals with this problem. In addition, we generalize the properties of different datasets and categorize them into two groups with distinct characteristics.

This work brings the following new contributions:

1. We show that the advantage or weakness of an algorithm over the others can be determined by the number of single-word phrases that this algorithm extracts as keyphrases.
2. We highlight the difference between two types of datasets used in the KE domain: datasets where keyphrases for each text are manually assigned by readers, and datasets where keyphrases are assigned by the authors themselves. We demonstrate that datasets with reader-assigned keyphrases tend to exhibit a problem related to single-word phrases, the extent of which varies by dataset. Specifically, algorithms that extract fewer single-word phrases often perform better in evaluations, as accurately extracting single-word keyphrases from a set of candidates is more difficult than it is for multi-word phrases.
3. In this work, we demonstrate that it is possible to improve the quality of the existing algorithms by considering the properties of single-word phrases, especially for datasets where the single-word phrase problem is strongly pronounced.

2 Related Work

This section will focus on

1. Unsupervised KE algorithms.
2. On the length feature in KE algorithms.

We divide unsupervised KE methods into graph-based, statistics-based, and embedding—transformer-based methods. Then we will highlight the length feature that is actively used. In the review, we primarily focus on the algorithms used in the study. These algorithms are described in more detail than the others.

2.1 Graph-based Unsupervised Methods

The original TextRank [28] is word-based and this algorithm can be defined as a baseline for graph-based methods. Firstly, it creates a graph to rank words, where words are vertices and edges represent the fact of word co-occurrence in a given text within a window of size n (edges are not weighted). The words (vertices of the graph) are weighted and ranked.

To calculate the word weight, a modification of the PageRank formula [8] is used. Then one-third of the top-ranked words are selected, which are denoted as one-word keyphrases or merged into multi-word keyphrases if they follow each other in the text. The authors report that the best results are achieved with $n = 2$ when only nouns and adjectives are allowed in the keyword set [28].

The approach proposed for graph construction and the vertices weighting is actively used in the KE domain. While the vertices can be text words, candidate phrases, noun phrases, or document topics, the edges may or may not be weighted and represent various types of connections between nodes, such as the co-occurrence of words or semantic similarity of units in the nodes.

SingleRank [37] is a candidate-based approach. Keyphrase candidates are extracted as the longest continuous sequences of nouns and adjectives in a given text. Phrases ending with an adjective are not allowed. The score of candidates is calculated by summing the scores of the words it contains. Word scores are counted recursively, similar to PageRank [8] modification in TextRank [28], using a local graph for a given document that includes only nouns and adjectives as vertices and weighs edges based on word co-occurrence. ExpandRank presented in the same article [37] is similar but it uses k most similar documents to weigh the edges based on word co-occurrence.

In PositionRank [16], phrases are built as contiguous noun phrases that match the pattern (adjective)*(noun)+ and have a length of up to three words. The weight of a phrase is calculated as the sum of the weights of its constituent words. The weight of each word in a phrase is calculated by building a word graph and using a modified version of PageRank, similar to TextRank [28], but taking into account the position of each word in the text. The idea behind the algorithm is to assign higher weights to words that are frequent and appear early in documents.

Another part of the graph-based algorithms uses a modification of the PageRank formula and integrates topic information in it in various ways. In [25], topics are first defined based on LDA [4]. The words are then weighed using a modified formula for each of the topics. That is, the word will have different weights regarding different topics. These scores are used to weigh the phrases. Candidate phrases in this method are extracted as noun phrases. A candidate's weight is calculated as the sum of the scores of the words included in the phrase. Since the score of a word varies across different topics, the weight of the candidate will be different relative to different topics. Candidate weights across different topics are combined into a final phrase weight by considering the document's topic distribution. All candidate phrases are ranked based on their final weights, and top-ranked phrases are selected as keyphrases.

In TopicRank [7] candidate phrases for a given document are defined as the longest sequence of nouns and adjectives. The algorithm extracts candidates and groups them by topic using hierarchical agglomerative clustering. A graph-based ranking model based on the PageRank formula [8] is applied to assign a weighted score to each topic. Here, topics are vertices, and edges are weighted according to the strength of semantic relations between topics, which depends on how often these topics' keyphrase candidates appear close to each other in the document. Keyphrases are then extracted by selecting a candidate from each of the top-ranked topics. Three strategies were considered for selecting a candidate for each topic. The first is to choose the candidate that appears earliest in the document. The second is to choose the candidate with the highest frequency of occurrence. The third is to select the centroid of the topic cluster. The authors showed that selecting the first occurrence of a keyphrase candidate is the best strategy of the three.

In MultipartiteRank [6], candidate phrases are built as continuous noun phrases that match the pattern (adjective)*(noun)+. Candidates are grouped into topics with hierarchical agglomerative average linkage clustering. The graph is constructed as follows: candidate phrases are vertices that are connected if they belong to different topics. The authors adjust the incoming edge weights of the nodes that correspond to the first occurrence for each topic. Thus, candidates that occur at the beginning of the text are promoted in comparison with other candidates from the topic. When the graph is built, keyphrase candidates that are nodes of the graph are ranked based on a modification of the PageRank algorithm. Top-ranked candidates are selected as keyphrases.

2.2 Statistical and Embedding/Transformer Based Unsupervised Methods

RAKE [32] extracts candidate keyphrases from a text by identifying the longest sequences of continuous words that are split at phrase delimiters, stop-word positions, and word delimiters. RAKE proposed a method to extract additional words that act as delimiters between phrases. The weight of a candidate is determined by summing the weights of its words. To estimate the weights, it uses a graph of word co-occurrences within extracted candidates. Word's score is calculated based on the ratio of its degree in the obtained graph to its frequency in the text. Since the degree of a graph vertex is the number of graph edges incident to this vertex, then the degree of a vertex word shows the number of words with which this word co-occurs within the extracted candidates. So, the total score of a word is computed by dividing the number of co-occurring words with that particular word by its frequency. That is why we can exclude graph-based notation here and look at this approach as statistical. In contrast, in previously described graph-based methods we could not exclude graph-based notation.

KP-miner [15] extracts candidate phrases in two main steps. First, it identifies n-grams by extracting sequences of words separated by punctuation marks and stop words. It then applies two additional filters to these candidates. The first filter requires each phrase to occur at least n times in the text, where n depends on the length of the document. The second filter considers the position of the phrase within the text. It is controlled by the CutOff parameter. To filter the phrases, it uses the number of words encountered before the first occurrence of a phrase. In the second step, the algorithm scans the text again and selects, from the unfiltered units, the sequences of maximum length as candidates.

KP-miner uses Tf-Idf for ranking. Since the problem with this method is that it prefers single-word phrases, which are the most frequently generated, the authors introduce a special boosting factor for multi-word phrases. It allows the algorithm to balance this bias towards single terms. In addition, the document frequency of multi-word phrases is assumed to be equal to 1 when calculating their Idf. This gives an advantage to multi-word phrases over single-word ones.

YAKE [9] uses a sliding window of 3-g generating a contiguous sequence of 1, 2, and 3-g candidate keywords. Candidates that begin or end with a stop word are not allowed. YAKE exploits the following features: casing, word position, word frequency, word relatedness to context, and word difference. These features are combined within a single complex formula [10, 11].

The following algorithms involve word embeddings and transformer models into KE. Candidates are commonly extracted as noun phrases, sequences of nouns and adjectives. EmbedRank [2] is an embedding-based method. It allows for the calculation of the distance between the embedding of a candidate and that of the entire document, as well as between candidates themselves. To achieve this, the authors represent the candidates and the document in the same high-dimensional vector space and utilize publicly available pre-trained models of Sent2Vec [29] and Doc2Vec [22]. Keyphrases were selected by ranking the candidate phrases according to their cosine distance to the document embedding.

AttentionRank [13] is based on the self-attention mechanism of the BERT model [12], as well as the hierarchical attention retrieval (HAR) mechanism [40]. Self-attention determines the importance of a candidate within the context of a sentence, while cross-attention measures the semantic relevance between a candidate and sentences within a document. These two values are used to calculate the final score of a candidate.

In the TripleRank [23] method two features: keyphrase semantic diversity and keyphrase coverage are introduced to address the issue of synonyms, along with positional information. Keyphrase coverage calculates the similarity between candidates and other words in the document using Word2Vec, while semantic diversity uses LDA [4] to diversify the topics represented in phrases and avoid extracting phrases from the same words.

The sentence embedding model SIF [35] from SIFRank is intended to explain the relationship between sentence embeddings and the topic of the document. The authors combine ELMo [30] with SIF to compute phrase embeddings and document embeddings. To weigh the phrases, cosine similarity is used to calculate the distance between the candidates and the topic. The paper also proposes a method called document segmentation to speed up the computation of word embeddings in long documents and uses position-biased weight for long documents. Further development of embedding/transformer-based KE algorithms are presented in the [24, 14, 34].

2.3 Phrase Length Feature

One of the employed techniques in candidate weighting is the method, which assigns weights to candidates by summing the scores of the constituent words. This is exploited in SingleRank [37], PositionRank, Topical PageRank [16, 25], RAKE [32] and in the re-implementation of TextRank that is exploited in this paper (will be discussed below). Since all words have positive scores, we expect these algorithms to prioritize multi-word phrases over single-word phrases in the weighting process. As we will see later, the implementations of these algorithms we used either do not extract single-word keyphrases at all or extract only a minimal number of them. This will differentiate them and their performance from other algorithms that extract a greater number of single-word phrases.

KP-miner [15] also employs features that limit the extraction of single-word phrases, but as we will see later, despite these features, when processing abstracts of scientific publications, KP-miner extracts more single-word phrases than the methods discussed in the previous paragraph. This algorithm introduces a special boosting factor for multi-word phrases. Besides, KP-miner exploits Tf-Idf for candidate ranking where the document frequency of multi-word phrases is assumed to be equal to 1 when calculating their Idf. Both these factors give an advantage to multi-word phrases over single-word ones.

In the YAKE algorithm, there is also a remark [9] regarding the formula used to calculate the weight of a candidate phrase, which combines the features employed in the algorithm. In particular, one part of the formula is considered and explained: one of the potential problems with this part of the formula is that the length of a candidate keyphrase can significantly influence its score, favoring longer relevant candidates over shorter relevant ones. To mitigate this effect and extract keyphrases regardless of their length, YAKE introduces additional components into the general formula. However, as we will see later with short texts, YAKE significantly limits the number of extracted single-word phrases. It extracts more of them than the approaches from the first paragraph but fewer than, for example, KP-Miner.

This section aims to demonstrate that algorithms incorporate features and methods influencing the number of single-word keyphrases they extract. Additionally, we hypothesize that algorithms can be grouped based on their tendency to extract more or fewer single-word keyphrases. The central focus of this article is to examine how the tendency of algorithms to select more or fewer single-word phrases impacts the evaluation of their performance.

3 Research Tools

As mentioned in the Introduction, this article focuses solely on candidate-based KE approaches. This means that each exploited below algorithm operates as follows. In the first step, a set of candidate phrases is constructed. In the second step, this set of candidate phrases is ranked, and the top k-ranked phrases are extracted as keyphrases.

For all the algorithms in the experiments, we use the same method to extract the set of candidate phrases. The candidate phrases are extracted as noun phrases using grammar, the description of which is provided below. This method was chosen because it allows us to obtain a good quality set of candidate phrases compared to using other methods, such as extracting n-grams or extracting sequences of words between delimiters. By ”good quality” we mean that the set of candidate phrases should not be too large on the one hand, and on the other hand, it should contain a sufficient number of true positive keyphrases. In the KE domain, it has been noted that the size and quality of the set of candidate phrases affect the quality of the keyphrases obtained after ranking [38]. By quality here, we mean the evaluation with F1-score. Additionally, the vast majority of approaches reviewed in the Related Work extract candidate phrases as noun phrases or as sequences of nouns and adjectives that do not end with an adjective.

3.1 PKE, Algorithms and Modifications, Text Pre-Processing

PKE [5] is an end-to-end Python KE pipeline that includes the re-implementation of KE algorithms. PKE allows using original versions of algorithms and modifications of its components. Our goal was to evaluate how well the ranking methods performed in KE approaches. To check this, every ranking method should take the same set of candidate phrases as input.

We use a modified version of TextRank. TextRank is a word-based approach but PKE allows us to work with it as with a candidate-based method [5]. TextRank re-implementation in the PKE framework is as follows: words are ranked and then keyphrase candidates are either composed from the T-percent highest-ranked words as in the original paper or extracted using the candidate selection method. In the latter, candidates are ranked using the sum of their words' weights. We exploit the second opportunity.

There are candidate-based methods that extract keyphrase candidates either as n-grams or as combinations of n-grams, filters, and sequences, including YAKE, TF-IDF, and KP-miner. To evaluate the ranking strategies, the candidate extraction method for these approaches was kept consistent with that used for the other methods: candidates were extracted as noun phrases.

We exploit the following methods implemented in PKE: graph-based (TextRank, SingleRank, TopicRank, PositionRank, MultipartiteRank, TopicalPageRank) and statistics-based (FirstPhrase, TfIdf, KP-miner, YAKE). The FirstPhrases method extracts a fixed number of phrases that are met at the beginning of a given document. The TfIdf method assigns a weight to each candidate phrase using the TfIdf formula and then ranks the phrases based on these weights. The ranking strategies of other exploited KE algorithms are described in the Related Work.

In all experiments, candidate phrases are extracted from texts as continuous sequences of nouns and adjectives satisfying the default PKE grammar:

```
[NBAR: {<NOUN|PROPN|ADJ>*<NOUN|PROPN>},
NP: {<NBAR>}{<NBAR><ADP><NBAR>}]
```

where nouns and adjectives are not stop words. We exploited the extended stop word list from [31], which was applied to all algorithms and across all datasets. Consequently, all exploited methods differ only in the ranking step.

3.2 Datasets

The initial hypothesis of this study was that datasets with keyphrases annotated by authors differ from those annotated by readers. Therefore, we initially categorized the datasets into two groups: datasets with reader-assigned keyphrases and datasets with author-assigned keyphrases. The first group includes INSPEC, SemEval 2010 (only reader-assigned keyphrases), and SemEval 2017. The second dataset group comprises kp20k, PubMed, and KPBiomed.

Datasets with reader-assigned keyphrases. From *INSPEC* [1] [19] we use a test subset comprising 500 scientific publication titles with abstracts, each with reference keyphrases manually assigned by readers. Dataset domains: Computers and Control and Information Technology. The average number of reference keyphrases per text in the INSPEC dataset is 9.5.

SemEval2010 [2] [20] test subset containing 100 full-text documents. Only reader-assigned keyphrases were used as reference keyphrases. We utilized the SemEval 2010 dataset in the following format: SemEval 2010 (TA) includes only the titles and abstracts of articles. Dataset domains: Distributed Systems, Information Search and Retrieval, Distributed Artificial Intelligence - Multiagent Systems, Social and Behavioral Sciences - Economics. The average number of reference reader-assigned keyphrases per text in the SemEval2010 dataset is 12.4.

SemEval2017 [1]. A corpus for this task was constructed from open-access publications available on ScienceDirect. Each document consists of one paragraph of text drawn from a scientific paper. A total of 500 paragraphs from journal articles, distributed among the domains of Computer Science, Material Sciences, and Physics, were selected. We utilized the dataset available in the YAKE-associated repository [3]. The first 200 documents were designated as the test dataset. The average number of reference keyphrases per text in the SemEval 2017 dataset is 11.5.

Datasets with author-assigned keyphrases. We exploited the first 2,000 documents from the test subsets of *kp20k* [4] [27]. Each document comprises a title and abstract from scientific articles and includes author-assigned keyphrases. Dataset domain: Computer Science. The average number of reference keyphrases per text in the kp20k dataset is 5.3.

Dataset *PubMed* [5] [33] contains 1,320 articles with full text and author-assigned keyphrases. Titles are separated from full texts but abstracts are not. For each text, we took the title and the first 1,200 characters of the full text, assuming that in this way we would be able to use most of the abstracts. We exploited the first 500 documents from the database as a test collection. Dataset domain: Biomedical. The average number of reference keyphrases per text in the PubMed dataset is 5.4.

KPBiomed[6] [18]: the first 2,000 documents from the test subset of this dataset were used. Each document includes a title and abstract from a scientific article and author-assigned keyphrases. Dataset domain: Biomedical. The average number of reference keyphrases per text in the PubMed is 5.3.

3.3 Evaluation

Since this study is based on PKE, we utilize its built-in evaluation method to calculate the macro-average F1-score at n keyphrases (F@n). The F1-score is the most standard and widely used evaluation approach in the KE domain. The value of n indicates that the KE algorithm should select up to n phrases based on the ranking results, or fewer if selecting n phrases is not feasible. This evaluation scores compares the set of automatically extracted keyphrases with the set of reference keyphrases annotated by experts, readers, or authors.

We use $n = 3$ and $n = 5$ for datasets with author-assigned keyphrases, as the average

[1] https://huggingface.co/datasets/taln-ls2n/inspec
[2] https://huggingface.co/datasets/taln-ls2n/semeval-2010-pre
[3] https://github.com/LIAAD/KeywordExtractor-Datasets/blob/master/datasets/SemEval2017.zip
[4] https://huggingface.co/datasets/taln-ls2n/kp20k
[5] https://huggingface.co/datasets/taln-ls2n/pubmed
[6] https://huggingface.co/datasets/taln-ls2n/kpbiomed

number of keyphrases in the references for these collections is approximately five. For datasets with reader-assigned keyphrases, we use $n = 5$ and $n = 10$, as the average number of keyphrases in their references is approximately ten to twelve

Extracted phrases and reference phrases are stemmed before evaluation. We do not remove phrases that do not occur in the corresponding text from reference keyphrases. This practice exists in the domain, e.g., in [19, 37] and it makes Recall and F1-score higher.

4 Experiment Description, Results, and Discussion

We aim to accomplish the following in this section:

– *Examining dataset characteristics*: we analyze the characteristics of the datasets that may influence the quality of algorithm performance.

– *Formulating the single-word phrase problem*: we briefly outline and formulate the problem of single-word phrases.

– *Unsupervised KE algorithms evaluation*: we evaluate how 10 selected unsupervised KE algorithms perform on each dataset group (datasets with reader-assigned and author-assigned keyphrases) and examine the role single-word phrases play in their performance within each dataset group.

The length of a phrase refers to the number of words it contains. Reference keyphrases are manually assigned to a text and represent the keyphrases that the KE algorithm should ideally extract. All KE algorithms exploited in this study operate in two steps. In the first step, a set of candidate phrases is extracted, a process identical across all considered algorithms. In the second step, each algorithm ranks the set of candidate phrases.

4.1 Dataset Characteristics

In this part of the study, we collected the following characteristics for each dataset:

– *Reference Keyphrases*: the average number of reference keyphrases of lengths 1, 2, and 3 per document in the dataset.

– *Candidate Phrases*: the average number of candidate phrases per document of lengths 1, 2, and 3, extracted as continuous sequences of nouns and adjectives satisfying the chosen grammar.

– *Evaluation*: the Precision and F1-scores obtained when evaluating the quality of candidate phrases of lengths 1, 2, and 3. The quality of single-word phrases in the candidate set was evaluated against the single-word phrases in the reference set. Similarly, the quality of two-word candidate phrases was evaluated by comparison with two-word phrases in the reference set. The same procedure was applied to phrases with length 3.

The obtained results are presented in the Table 2. Analysis of Table 2 reveals the following differences between the two groups of datasets in the distribution of keyphrases length 1 and 2 in references and candidates.

In reference keyphrase set we observe the following. *Datasets with author-assigned keyphrases*: in datasets where authors assigned keyphrases, the average number of single-word and two-word phrases per text in the references is roughly equal, with an average difference of 0.45 keyphrases. In the case of kp20k, this difference is greater than in PubMed and KPBiomed.

Datasets with reader-assigned keyphrases: in datasets where readers assigned keyphrases, the difference in the number of single-word and two-word phrases in the references is more pronounced, with an average difference of 2.86, indicating a higher prevalence of two-word phrases. However, for SemEval2017, this difference is somewhat smaller compared to INSPEC and SemEval2010.

In keyphrase candidates set we observe the following. The candidate phrase sets for

Table 2. Dataset Characteristics. Ref: the average number of reference keyphrases of a specified length per document. Cand: the average number of candidate phrases of a specified length per document. P: Precision. F1: F1-score. Reader keys: indicates datasets where readers assigned keyphrases. Author keys: indicates datasets where authors assigned keyphrases. SemEval2010: SemEval2010(TA) contains only titles and abstracts

	INSPEC reader keys				SemEval2010 reader keys				SemEval2017 reader keys			
len.	Ref	Cand	P	F1	Ref	Cand	P	F1	Ref	Cand	P	F1
1	1.31	11.85	8.64	13.43	1.99	16.18	4.87	8.02	3.95	16.23	17.04	25.04
2	5.16	8.58	40.64	46.99	5.54	11.38	16.09	19.70	5.13	11.765	31.98	40.55
3	2.44	2.82	40.56	40.18	1.98	3.49	12.06	12.57	2.78	4.175	34.43	35.82

	PubMed author keys				KPBiomed author keys				kp20k author keys			
len.	Ref	Cand	P	F1	Ref	Cand	P	F1	Ref	Cand	P	F1
1	3,93	18,7	6.30	10.56	2,18	22,75	5.16	8.79	1.75	14.08	5.19	7.12
2	3,34	12,66	6.42	10.46	2,11	16,58	6.01	9.91	2.44	10.86	9.13	13.17
3	1,15	4,34	4.77	6.54	0,71	5,5	4.89	6.90	0.78	3.87	7.02	8.86

all collections were extracted uniformly, and all datasets primarily consisted of titles and abstracts. Consequently, the proportion of single-word and two-word phrases in the candidate sets is similar across all datasets. In each dataset, the number of single-word phrases exceeds that of two-word phrases by a factor of 1.3 to 1.5. Despite the similar proportions of single-word and two-word phrases in the datasets, the evaluation of these sets using Precision and F1-score differs, which is partly a consequence of the differences in proportions between reference keyphrases of lengths 1 and 2.

Datasets with author-assigned keyphrases: the disparity between Precision and F1-score values for single-word and two-word phrases, when keyphrases are assigned by authors, is not as significant as in the case of reader-assigned keyphrases. Specifically, the difference in Precision is 0.12 for PubMed, 0.85 for KPBiomed, and 0.94 for kp20k. In terms of the F1-score, the differences are 0.1 for PubMed, 1.12 for KPBiomed, and 6.05 for kp20k.

Datasets with reader-assigned keyphrases: the disparity between Precision and F1-score values for single-word and two-word phrases, when keyphrases are assigned by readers, is very significant. Specifically, in terms of Precision, this difference is 32 for INSPEC, 11.22 for SemEval2010(TA), and 14.94 for SemEval2017. In terms of F1-score: 33.56 for INSPEC, 11.68 for SemEval2010(TA), and 15.51 for SemEval2017.

For datasets where readers assigned keyphrases, the number of keyphrases in the references is approximately twice as large as in the references of texts where authors assigned keyphrases. This difference is attributable to the greater number of multi-word phrases. Two-word phrases are less represented in the reference sets of author-annotated datasets than in reader-annotated datasets. Furthermore, for datasets where keyphrases were assigned by authors, Precision and F1-score for two-word candidate phrases are low: in two out of three cases, they are lower than the corresponding values for single-word phrases.

4.2 Single-Word Phrase Problem

The main points from the previous section are:

1. *Datasets with reader-assigned keyphrases*: in the reference sets of these datasets there are more multi-word phrases compared to single-word phrases.
2. In the candidate phrase sets, the number of single-word phrases is 1.3 to 1.5 times higher than that of two-word phrases.

3. *Datasets with reader-assigned keyphrases*: Precision and F1-score calculated for single-word candidate phrases relative to the set of single-word keyphrases in the references are significantly lower than those for two-word and three-word phrases.

The essence of the single-word phrase problem for datasets with reader-assigned keyphrases: the initial set of candidate phrases contains a significant number of single-word phrases, and for single-word phrases in the candidate set, the true positive percentage may be significantly lower than for multi-word phrases. This problem is dataset-dependent. Since some of the main datasets in the KE domain present this problem it is worth paying attention to. The consequences of the problem: it is more difficult to extract true positive single-word keyphrases than the same for multi-word phrases. Hypotheses based on this problem: KE algorithms can achieve higher performance not only by selecting well-suited linguistic features but also by minimizing the number of extracted single-word keyphrases.

4.3 Unsupervised KE Algorithms Evaluation and Single-Word Phrase Problem

To demonstrate the connection between the total number of single-word phrases in the resulting keyphrase set and the algorithm's performance, we conducted the following experiment. We ran 10 unsupervised KE algorithms on all six datasets and collected the following:

– The average number of single-word keyphrases extracted per text by the algorithm.

– The quality of the algorithm's performance, evaluated using the F1-score, without any restrictions on the algorithm's operation.

– The quality of the algorithm's performance, evaluated using the F1-score, when the algorithm is restricted to extracting only multi-word keyphrases.

The results are presented in the Table 3. In the I-1 phr. columns, which show the average number of single-word phrases extracted per text by the algorithm, five cases with the highest number of such phrases are highlighted in bold. Corresponding to these highlights, in the rows, instances where prohibiting the extraction of single-word phrases led to an improvement in the algorithm's performance are also highlighted in bold. The top five scores obtained for each specific dataset are highlighted in italics. In the Table: I-1 phr.is an average number of single-word phrases extracted per text by the algorithm; all phr. is evaluation score when there were no restrictions on keyphrase extraction; multi - evaluation score when algorithms were restricted from extracting single-word keyphrases

The results indicate the following. We observe **two groups of algorithms**. The first group of algorithms tends to minimize the number of extracted single-word keyphrases (TextRank, SingleRank, PositionRank, Topical PageRank). The second group comprises algorithms that extract the highest number of single-word keyphrases (FirstPhrase, TopicRank, MultipartiteRank, KP-miner). Additionally, we highlight two intermediate algorithms: the first, YAKE, is closer and more likely to belong to the first group, while the second, TfIdf, is closer and more likely to belong to the second group of algorithms.

Algorithms from the second group demonstrate the best results compared to other algorithms on datasets with author-assigned keyphrases. However, these algorithms perform worse on datasets with reader-assigned keyphrases. If algorithms from the second group are restricted from extracting single-word phrases, their performance quality improves on datasets with reader-assigned keyphrases. This quality improvement is more noticeable the more pronounced the single-word phrases problem is in the dataset.

For datasets where keyphrases were assigned by readers, algorithms from the first group demonstrated higher performance compared to algorithms in the second group. However, this difference begins to diminish once we prohibit

Table 3. Algorithms performance

DataSet	INSPEC (reader)			SemEval2010 (reader)			SemEval2017 (reader)		
Type	I-1 phr.	all phr.	multi	I-1 phr.	all phr.	multi	I-1 phr.	all phr.	multi
F@5									
FirstPhrase	**2.20**	**25.38**	***34.57***	**2.07**	**12.35**	***14.48***	**2.33**	**13.71**	**17.38**
TextRank	0.12	30.84	31.06	0.04	11.02	11.02	0.01	16.50	16.53
SingleRank	0.33	30.72	31.83	0.30	13.69	*13.73*	0.17	18.78	18.46
TopicRank	**2.3**	**25.73**	***34.17***	**2.29**	**11.38**	**12.83**	**2.12**	**16.32**	**17.08**
MultipartiteRank	**2.22**	**25.92**	***34.68***	**2.30**	**12.10**	**13.20**	**2.12**	**16.94**	**17.73**
PositionRank	0.65	30.20	32.19	0.48	14.12	*14.10*	0.41	*18.91*	18.92
TopicalPageRank	0.42	30.48	32.00	0.33	13.60	13.67	0.17	*18.80*	18.56
Tf-Idf	**1.63**	**29.73**	***34.69***	**2.15**	**12.32**	***13.75***	**1.95**	*18.39*	17.90
KP-miner	**1.99**	**28.74**	***34.39***	**2.36**	**12.12**	**15.04**	**2.61**	**18.34**	***18.44***
YAKE	0.82	31.96	34.04	0.95	*15.12*	*14.65*	0.55	*18.36*	18.18
F@10									
FirstPhrase	**4.8**	**30.76**	***41.92***	**4.9**	**12.35**	**16.54**	**4.71**	**21.01**	**25.02**
TextRank	1.34	*37.78*	40.27	0.39	15.70	15.52	0.42	24.27	24.25
SingleRank	1.88	37.19	40.81	1.44	17.04	16.78	1.02	*26.51*	26.07
TopicRank	**4.97**	**30.20**	**40.29**	**5.18**	**13.51**	**14.57**	**4.85**	**21.83**	**22.70**
MultipartiteRank	**4.90**	**31.05**	**41.58**	**5.25**	**14.12**	**15.91**	**4.72**	**22.29**	**24.58**
PositionRank	2.52	35.50	40.81	2.12	*17.78*	*17.58*	1.71	*26.52*	25.93
TopicalPageRank	2.03	36.98	40.80	1.55	17.14	16.96	1.14	*26.66*	*26.26*
Tf-Idf	**3.03**	**36.99**	**41.76**	**3.30**	**15.47**	**16.20**	**3.14**	*25.72*	25.40
KP-miner	**3.44**	**35.78**	**41.73**	**3.66**	**16.41**	**16.90**	**4.33**	**24.96**	**25.36**
YAKE	2.84	36.56	41.20	2.97	16.98	17.21	2.20	25.42	*25.78*
Dataset	kp20k (author)			PubMed (author)			KPBiomed (author)		
F@3									
FirstPhrase	**1.05**	*13.91*	13.64	**1.29**	*13.84*	11.20	**1.20**	*14.10*	12.66
TextRank	0.00	5.40	5.40	0.00	2.93	2.93	0.00	3.20	3.20
SingleRank	0.04	8.69	8.59	0.04	6.54	6.21	0.04	6.11	5.99
TopicRank	**1.15**	*12.23*	11.56	**1.37**	*12.65*	9.82	**1.54**	*12.67*	10.10
MultipartiteRank	**1.06**	*13.65*	12.56	**1.29**	*13.97*	11.00	**1.36**	*13.93*	10.98
PositionRank	0.09	10.86	10.61	0.11	9.96	8.87	0.07	9.34	8.80
TopicalPageRank	0.05	9.35	9.24	0.04	7.38	6.94	0.04	6.74	6.60
Tf-Idf	**1.03**	*14.17*	13.03	**1.47**	*17.33*	*12.20*	**1.60**	*15.13*	11.90
KP-miner	**1.30**	*14.45*	14.19	**1.58**	*17.46*	*13.20*	**1.67**	*15.26*	13.13
YAKE	0.24	12.65	12.09	0.31	11.83	10.36	0.35	12.04	10.46
F@5									
FirstPhrase	**1.96**	*15.32*	13.79	**2.28**	*16.11*	12.04	**2.16**	*17.09*	13.47
TextRank	0.04	7.93	7.91	0.00	5.19	5.19	0.01	5.01	4.97
SingleRank	0.20	11.60	11.15	0.17	9.41	8.53	0.16	8.68	8.07
TopicRank	**2.12**	*13.07*	11.64	**2.36**	*14.31*	9.85	**2.58**	*14.08*	10.29
MultipartiteRank	**2.04**	*14.31*	13.11	**2.28**	*15.72*	10.96	**2.41**	*15.75*	11.83
PositionRank	0.36	13.51	12.88	0.39	13.02	10.95	0.26	12.02	10.72
TopicalPageRank	0.25	11.92	11.57	0.20	10.18	9.23	0.18	9.14	8.45
Tf-Idf	**1.54**	*14.71*	12.88	**2.31**	*18.45*	11.41	**2.43**	*16.70*	12.34
KP-miner	**2.05**	*15.11*	14.30	**2.66**	*18.52*	12.79	**2.63**	*16.95*	13.34
YAKE	0.70	14.87	13.73	0.76	*14.60*	11.90	0.85	15.21	12.37

algorithms in the second group from extracting single-word phrases.

For datasets with authors – assigned keyphrases, the single-word phrase problem does not exist. Prohibiting the extraction of single-word keyphrases does not enhance the performance quality of algorithms from either group; on the contrary, it degrades their performance.

5 Conclusion

The study includes a series of experiments that overcome several shortcomings in unsupervised KE. We formulated the dataset-dependent single-word phrase problem and demonstrated its cause. We analyze how different KE algorithms rank keyphrase candidates and explore one of the reasons that influence quality evaluation.

Results show that an algorithm can perform better or worse depending on whether it allows fewer or more one-word phrases in the extracted keyphrase set. Removing single-word candidate phrases before the ranking stage can improve performance for datasets with reader-assigned keyphrases. This is critical for algorithms that efficiently rank phrases but allow too many single-word phrases among the extracted keyphrases. We also note that prohibiting the extraction of single-word keyphrases does not apply to datasets with authors-assign keyphrases.

References

1. **Augenstein, I., Das, M., Riedel, S., Vikraman, L., McCallum, A. (2017).** SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Association for Computational Linguistics, Vancouver, Canada, pp. 546–555. DOI: 10.18653/v1/S17-2091.
2. **Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M., Jaggi, M. (2018).** Simple unsupervised keyphrase extraction using sentence embeddings. Proceedings of the 22nd Conference on Computational Natural Language Learning, pp. 221–229. DOI: 10.18653/v1/k18-1022.
3. **Bernardini, A., Carpineto, C., D'Amico, M. (2009).** Full-subtopic retrieval with keyphrase-based search results clustering. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, IEEE, pp. 206–213. DOI: 10.1109/wi-iat.2009.37.
4. **Blei, D. M., Ng, A. Y., Jordan, M. I. (2003).** Latent dirichlet allocation. The Journal of Machine Learning Research, Vol. 3, pp. 993–1022.
5. **Boudin, F. (2016).** pke: an open source python-based keyphrase extraction toolkit. Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations, pp. 69–73.
6. **Boudin, F. (2018).** Unsupervised keyphrase extraction with multipartite graphs. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Vol. 2, pp. 667–672. DOI: 10.18653/v1/N18-2105.
7. **Bougouin, A., Boudin, F., Daille, B. (2013).** TopicRank: Graph-based topic ranking for keyphrase extraction. Proceedings of the 6th International Joint Conference on Natural Language Processing, Asian Federation of Natural Language Processing, pp. 543–551.
8. **Brin, S., Page, L. (1998).** The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, Vol. 30, No. 1–7, pp. 107–117. DOI: 10.1016/s0169-7552(98)00110-x.
9. **Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A. (2020).** YAKE! Keyword extraction from single

documents using multiple local features. Information Sciences, Vol. 509, pp. 257–289. DOI: 10.1016/j.ins.2019.09.013.

10. **Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., Jatowt, A. (2018).** A text feature based automatic keyword extraction method for single documents. Springer International Publishing, pp. 684–691. DOI: 10.1007/978-3-319-76941-7_63.

11. **Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., Jatowt, A. (2018).** YAKE! Collection-Independent Automatic Keyword Extractor. Springer International Publishing, pp. 806–810. DOI: 10.1007/978-3-319-76941-7_80.

12. **Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019).** BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

13. **Ding, H., Luo, X. (2021).** AttentionRank: Unsupervised keyphrase extraction using self and cross attentions. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 1919–1928. DOI: 10.18653/v1/2021.emnlp-main.146.

14. **Ding, H., Luo, X. (2022).** AGRank: Augmented graph-based unsupervised keyphrase extraction. Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing, Association for Computational Linguistics, Vol. 1, pp. 230–239.

15. **El-Beltagy, S. R., Rafea, A. (2010).** KP-miner: Participation in SemEval-2. Proceedings of the 5th International Workshop on Semantic Evaluation, Association for Computational Linguistics, pp. 190–193.

16. **Florescu, C., Caragea, C. (2017).** Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Vol. 1, pp. 1105–1115. DOI: 10.18653/v1/p17-1102.

17. **Gutwin, C., Paynter, G., Witten, I., Nevill-Manning, C., Frank, E. (1999).** Improving browsing in digital libraries with keyphrase indexes. Decision Support Systems, Vol. 27, No. 1–2, pp. 81–104. DOI: 10.1016/s0167-9236(99)00038-x.

18. **Houbre, M., Boudin, F., Daille, B. (2022).** A large-scale dataset for biomedical keyphrase generation. Proceedings of the 13th International Workshop on Health Text Mining and Information Analysis (LOUHI), Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid), pp. 47–53. DOI: 10.18653/v1/2022.louhi-1.6.

19. **Hulth, A. (2003).** Improved automatic keyword extraction given more linguistic knowledge. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Vol. 10, pp. 216–223. DOI: 10.3115/1119355.1119383.

20. **Kim, S. N., Medelyan, O., Kan, M.-Y., Baldwin, T. (2010).** SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. Proceedings of the 5th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Uppsala, Sweden, pp. 21–26.

21. **Kim, S. N., Medelyan, O., Kan, M. Y., Baldwin, T. (2010).** SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. Proceedings of the 5th International Workshop on Semantic Evaluation, Association for Computational Linguistics, pp. 21–26.

22. **Le, Q. V., Mikolov, T. (2014).** Distributed representations of sentences and documents.

Proceedings of Machine Learning Research, Vol. 32, No. 2, pp. 1188–1196. DOI: 10.48550/ARXIV.1405.4053.

23. **Li, T., Hu, L., Li, H., Sun, C., Li, S., Chi, L. (2021).** TripleRank: An unsupervised keyphrase extraction algorithm. Knowledge-Based Systems, Vol. 219, pp. 106846. DOI: 10.1016/j.knosys.2021.106846.

24. **Liang, X., Wu, S., Li, M., Li, Z. (2021).** Unsupervised keyphrase extraction by jointly modeling local and global context. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 155–164. DOI: 10.18653/v1/2021.emnlp-main.14.

25. **Liu, Z., Huang, W., Zheng, Y., Sun, M. (2010).** Automatic keyphrase extraction via topic decomposition. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 366–376.

26. **Liu, Z., Li, P., Zheng, Y., Sun, M. (2009).** Clustering to find exemplar terms for keyphrase extraction. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 257–266.

27. **Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y. (2017).** Deep keyphrase generation. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, pp. 582–592. DOI: 10.18653/v1/P17-1054.

28. **Mihalcea, R., Tarau, P. (2004).** TextRank: Bringing order into text. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 404–411.

29. **Moghadasi, M. N., Zhuang, Y. (2020).** Sent2Vec: A new sentence embedding representation with sentimental semantic. IEEE International Conference on Big Data, IEEE, pp. 4672–468. DOI: 10.1109/bigdata50022.2020.9378337.

30. **Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018).** Deep contextualized word representations. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Vol. 1, pp. 2227–2237. DOI: 10.18653/v1/N18-1202.

31. **Popova, S., Kovriguina, L., Mouromtsev, D., Khodyrev, I. (2013).** Stop-words in keyphrase extraction problem. Proceedings of the Conference of Open Innovation Association, FRUCT, pp. 113–121. DOI: 10.1109/FRUCT.2013.6737953.

32. **Rose, S., Engel, D., Cramer, N., Cowley, W. (2010).** Automatic keyword extraction from individual documents. Text Mining, pp. 1–20. DOI: 10.1002/9780470689646.ch1.

33. **Schutz, A. (2008).** Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods.

34. **Song, M., Xu, P., Feng, Y., Liu, H., Jing, L. (2023).** Mitigating over-generation for unsupervised keyphrase extraction with heterogeneous centrality detection. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 16349–16359. DOI: 10.18653/v1/2023.emnlp-main.1017.

35. **Sun, Y., Qiu, H., Zheng, Y., Wang, Z., Zhang, C. (2020).** SIFRank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. IEEE Access, Vol. 8, pp. 10896–10906. DOI: 10.1109/access.2020.2965087.

36. **Turney, P. D. (2000).** Learning algorithms for keyphrase extraction. Information Retrieval, Vol. 2, No. 4, pp. 303–336. DOI: 10.1023/a:1009976227802.

37. **Wan, X., Xiao, J. (2008).** Single document keyphrase extraction using neighborhood knowledge. Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI Press, Vol. 2, pp. 855–860.

38. **You, W., Fontaine, D., Barthès, J. P. (2012).** An automatic keyphrase extraction system for scientific documents. Knowledge and Information Systems, Vol. 34, No. 3, pp. 691–724. DOI: 10.1007/s10115-012-0480-2.

39. **Zeng, H. J., He, Q. C., Chen, Z., Ma, W. Y., Ma, J. (2004).** Learning to cluster web search results. , pp. 210–217DOI: 10.1145/1008992.1009030.

40. **Zhu, M., Ahuja, A., Wei, W., Reddy, C. K. (2019).** A hierarchical attention retrieval model for healthcare question answering. The World Wide Web Conference, Vol. 242, pp. 2472–2482. DOI: 10.1145/3308558.3313699.

Article received on 16/06/2024; accepted on 28/08/2024.
**Corresponding author is Svetlana Popova.*