Rapid Unsupervised Keyphrase Extraction from Single Document

Svetlana Popova, John Cardiff
TU Dublin
Dublin, Ireland
miracle.metaverse@gmail.com,
John.cardiff@tudublin.ie

Vera Danilova
Uppsala University
Uppsala, Sweden
vera.danilova@idehist.uu.se

***Abstract*—Keyphrases offer a concise representation of a document's content. They are valuable for improving web search results and enhancing tasks such as document tagging, text classification, or summarization. This makes keyphrase extraction is an essential component of text mining. Among the widely used constraints and features in existing keyphrase extraction methods, we identified several effective techniques that have not yet been used together: Part-of-Speech (PoS) restrictions, extended stop-word lists, and position-based features. To address this gap, we propose an approach that leverages automatically extracted extended stop word lists combined with PoS restrictions in keyphrases, and incorporates positional criteria. The main goal of the work was to develop a fast keyphrase extraction algorithm, which was built upon the three mentioned features. Experimental results on the INSPEC and SemEval 2010 datasets demonstrate the effectiveness of the proposed method.**

I. Introduction

Keyphrases offer a concise representation of a document's content and automatic keyphrase extraction involves "the automatic selection of important and topical phrases from the body of a document" [1]. Being more informative than single keywords, keyphrases enhance the efficiency of text mining processes. They can improve the functionality of information retrieval systems [2]–[6] and are useful across various natural language processing tasks, e.g., document categorization [7], text summarization [8], [9], opinion mining [10], web tagging [11]. Thus, automatic keyphrase extraction is an important research task. In practice, keyphrases usually contain from one to five words. Table I contains the example of the text and keyphrases from the INSPEC dataset [7].

This work is centered on the development of a rapid unsupervised keyphrase extraction approach from single documents that require no external resources such as WordNet, Wikipedia, or pre-trained models, including LDA models or transformer-based language models. These external resources can be used in addition to the proposed approach for further improvement. We exploit an iterative method to generate stop words (stoplist) for keyphrase extraction and introduce a keyphrase extraction algorithm that combines an extended stoplist, Part-of-Speech (PoS) restrictions, and a position feature. The results of experiments obtained on several well-known datasets (with short and long texts) affirm the effectiveness of the proposed method in comparison with baseline algorithms.

TABLE I. Text and keyphrases from the INSPEC dataset [7]

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.

II. Related Work

A. Classification of methods for keyphrase extraction

The existing keyphrase extraction approaches can be divided in several ways. Traditionally they are grouped as supervised and unsupervised. Another way is to distinguish them between two approach types:

- The keyphrase approach *Type 1* (word-based approaches) consists of two stages: (1) word weighing, followed by the ranking/classification, and (2) keyphrase construction based on the results of the previous stage. This approach mainly employs: *unsupervised* graph-based models for word weighing and ranking (e.g. [12]) or *supervised methods* - mainly neural networks (e.g. LSTM-RNN) and conditional random fields – that produce a decision for each word regardless of whether this word is part of a keyphrase or not and of its position within a phrase [11], [13]–[15].
- The keyphrase approach *Type 2* (candidate-based approaches) also includes two stages: (1) keyphrase candidates extraction, and (2) keyphrase selection based on the ranking/classification of the candidates. The following techniques have been proposed for candidate phrase selection: 1) extraction of n-grams from text or its parts, as in [16]–[18]; 2) extraction of noun phrases or PoS sequences according to predefined patterns [16], [19]; 3) subsequences with the highest frequency of occurrence in the collection [3], [4]; 4) longest continuous sequences within phrase delimiters [20]–[22]; and their combinations. In the second stage, the ranking or classification is performed (1) using *unsupervised ranking approaches* including graph-based [19], [21], tf-idf based [18], [23], embedding/transformer-based methods [24]–[30]. or, (2)

supervised methods including Naive Bayes [22], Support Vector Machines [31], Random Forest [32], Conditional Random Field [15], Neural Networks [33], etc. Candidates in these methods are commonly extracted as noun phrases or sequences of nous, and adjectives. These methods are based on the calculation of the distance between the embedding of a candidate and the entire document, sentence, or topic, as well as between candidates themselves.

There are important word and phrase-based features that are commonly taken into account in the keyphrase extraction domain: PoS feature, the position of the first occurrence of a given phrase with respect to the document start point, the length feature, words co-occurrence, the tf and idf of the phrase, keyphraseness (encodes the number of times a phrase appears in the training set as a keyphrase), the sequences of word prefixes and suffixes in the phrase, word embeddings, the nearest context enriched e.g. with PoS tags, external resource-based features extracted e.g. from Wikipedia and WordNet.

Also, keyphrase extraction approaches can be split based on the involvement of corpus-based features. One set of approaches including RAKE [20], TextRank [12], SingleRank [21], and YAKE [34], [35] searches for keyphrases in an individual document without using statistics based on the whole dataset. Another set of approaches produces corpus-based statistical features (e.g., tf-idf, as in [18], [23]). Moreover, the methods can be divided into those that do not need external resources to perform keyphrase extraction and those that use knowledge bases, such as WordNet or Wikipedia (e.g., [19], [36]).

This research is based on the *unsupervised approach Type 2* that extracts keyphrases from a *single document* and *requires neither corpus information nor external information or pre-trained models, including LDA-models or transformer-based language models*.

B. Positioning of the study and baseline Approaches

In this Section, we define the scope of our work and position it with respect to the representative literature related to different research directions within keyphrase extraction. As in TextRank [12], SingleRank [21], YAKE [34], [35] and RAKE [20], we focus on the unsupervised methods that can be efficiently applied to single documents and require neither corpus information nor external resources. For this reason, the mentioned methods are our main baseline approaches and we do not make comparisons with LDA-based or embedding—transformer-based approaches. We aimed to propose an algorithm that would be as simple and fast as possible. In selected baselines, the first one relates to the approach Type 1, and the other - to Type 2. TextRank and SingleRank are graph-based methods but they use graphs in different steps. YAKE and RAKE are statistical methods.

TextRank. TextRank [12] employs a graph-based approach to rank words and select a part of them for further keyphrase construction. In this method, a word graph is built, where words are vertices and the edges represent the fact of word co-occurrence in a given text in a window of size n (edges are not weighted). The vertices of the graph (words) are weighed and ranked, and one-third of the number of words with the highest rank are selected as keywords. Sequences of adjacent keywords are merged into multiword keywords. To calculate word weight, a modification of the PageRank formula [37] is used. The authors report that the best results are achieved with n=2 when only nouns and adjectives are allowed in the keyword set.

SingleRank. In this approach [21], keyphrase candidates are extracted and then a graph-based approach is used to rank them. The candidates are extracted as the longest continuous sequences of nouns and adjectives from a given text given the condition that phrases ending with an adjective are not allowed. The score of a candidate phrase is computed by summing the scores of the words contained in the phrase. The scores for words are counted recursively similar to PageRank [37] and exploit a local graph for a given document. The vertices of this graph include only nouns and adjectives, the edges are weighted based on word co-occurrence.

RAKE. RAKE [20] can be considered as one of the most rapid algorithms. First, it extracts keyphrase candidates followed by the evaluation of the phrase weight as the sum of its member word scores. The candidates are extracted as longest sequences of contiguous words split at phrase delimiters and stop word positions.

The sequences are not required to match predefined patterns as in [19] and, as opposed to SingleRank, there are no PoS restrictions. Instead, the authors of RAKE propose a method for the automatic construction of a $stoplist$ (list of stop words). The obtained stop words are used as delimiters between phrases. To construct this stoplist the algorithm uses a document set where keyphrases are labelled for each text. The frequency of each word occurring adjacent to a keyphrase is accumulated across these documents. The words with a term frequency higher than a predefined value that occurs more frequently as adjacent to keyphrases than within them are included in the stoplist. Once the stoplist is built it can be used for different datasets from the same area.

In RAKE, a candidate keyphrase is weighed as the sum of the weights of the constituent words. For weight estimation, a graph of word co-occurrences within the extracted candidates is built. The graph is used to calculate the degree of a word vertex, which equals to counting the number of different words this word co-occurs with within the candidates extracted from a given text. Naturally, the words with higher scores tend to occur often and in longer candidates. The authors use the ratio between a given word‘s degree and its frequency in a text to calculate the word‘s scores and report that in this case, the words that predominantly occur in longer candidates are more likely to have higher scores.

YAKE. In [34], [35], YAKE is experimentally shown to improve the results of other methods: RAKE, TextRank, and SingleRank. It uses a sliding window of 3-g generating a contiguous sequence of 1, 2, and 3-g candidate keywords. Keyphrase candidates beginning or ending with a stopword

TABLE II. BASELINE APPROACHES AND THE FEATURES: POS RESTRICTIONS, POSITION FEATURE, EXTENDED STOPLISTS

	RAKE	YAKE	TextRank	SingleRank
PoS			+	+
Extended StopList	+			
Position feature		+		

are not allowed. That is, as opposed to RAKE, TextRank, and SingleRank, there is a constraint on the maximum length of phrases, and the overall number of generated candidates is expected to exceed the number of candidates produced by RAKE or SingleRank. Furthermore, as compared to RAKE, YAKE uses a more sophisticated and expensive implementation of the ranking step. The weight calculation in YAKE is based on the following features: (1) Casing; (2) Word Position; (3) Word Frequency; (4) Word Relatedness to Context; and (5) Word DifSentence (see [34], [35] for details). All these statistical features are taken from single documents and do not involve the entire document collection.

Summary. Unlike TextRank and SingleRank, RAKE and YAKE do not use $PoS\ constraints$ and outperform the results reported in [12], [21] without using a PoS tagger. RAKE is mostly based on the extraction of domain-dependent *extended stop words* that are used as delimiters between keyphrases. In YAKE, a sophisticated ranking procedure is implemented that employs multiple document-based statistical features. YAKE is the only one among the described baseline algorithms that exploit the *position feature*. In RAKE and SingleRank, the keyphrase weight is estimated as the sum of the weights of its constituents (the words that make up a given keyphrase). In YAKE, the keyphrase weight is calculated based on the equation suggested in [34], [35] that takes into account the values of the mentioned statistical features. TextRank considers only word weights and the keyphrase weight estimation is not performed.

We summarized the information about the following features of the algorithms: PoS restrictions, position feature, and stoplists exploitation in Table II to show that these features are not used all at once. We assume that they can complete each other and we combine all of them in our approach.

The authors of RAKE report that TextRank is over 6 times slower than RAKE. Both SingleRank and TextRank exploit the ranking algorithm based on the modification of PageRank. YAKE, as expected, generates a sufficiently large candidate set and uses a much more complex ranking algorithm. We can expect that it will not work faster than RAKE. As in RAKE, speed is one of the priorities in our approach. Our research extends previous studies in that it combines different features from baseline algorithms. It relies on extended stoplists, position features, and PoS information bringing a new perspective to the ideas that underlie the RAKE algorithm. To our knowledge, none of the existing approaches incorporates these features in a unified unsupervised model independent from external resources and corpus-based information.

III. PROBLEM STATEMENT

Keyphrase Extraction is defined as follows. Let $D = \{d_1, d_2, ..., d_n\}$ be a set of n documents. Each document $d_i \in D$ has reference phrases (also called gold standard) – a set of keyphrases predefined by the experts $C_i = \{c_{i_1}, c_{i_2}, ..., c_{i_m}\}$. The goal of an unsupervised keyphrase extraction approach is for each text $d_i \in D$ automatically extract a list of keyphrase candidates, score them, create a ranked list, and select k ($@k$) top-ranked phrases as keyphrases $G_i = \{g_{i_1}, g_{i_2}, ..., g_{i_k}\}$ that should match the set of reference phrases as precisely as possible in terms of exact match micro-average $F1_{score}$ - $F1$ [16]. $F1$ allows to integrate information about the $Precision$ and $Recall$ of the extracted phrases:

$$Precision = \frac{|(C \cap G)|}{|G|} \tag{1}$$

$$Recall = \frac{|(C \cap G)|}{|C|} \tag{2}$$

$$FScore = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

where $|C \cap G|$ - is the number of correctly extracted phrases when processing all the texts of the collection, $|G|$ - is the total number of phrases automatically extracted by the algorithm from all the texts of the collection, $|C|$ - is the number of all phrases in the "gold standard". The gold standard (reference phrases) for each text includes an ideal list keyphrases manually tagged by an expert.

IV. PROPOSED APPROACH

The proposed approach consists of two stages: (1) keyphrase candidate extraction, (2) candidate scoring, ranking, and selection of k, if defined, or half top-ranked candidates as keyphrases. The algorithm takes the following input parameters: the list of delimiters (punctuation, except hyphens), the list of allowed PoS, and the stoplist. We named this approach SWaP – a **S**top**W**ords **a**nd **P**oS restriction-based approach to keyphrase extraction.

A. Keyphrase candidates extraction

Following RAKE, we extract candidates as word sequences separated by phrase delimiters and stop words. As delimiters, we use punctuation marks (except hyphens). Complementing TextRank and SingleRank, we introduce PoS restrictions so that the words with disallowed PoS tags act as delimiters between keyphrases. In the field, it is observed that nouns and adjectives are appropriate to be allowed within keyphrases and this fact is widely exploited. We allow these PoS in SWaP. The approach described above allows extracting a candidate set that is quite small and contains a relatively high percentage of true-positive keyphrases.

B. Candidate scores and ranking

As in RAKE and SingleRank, the keyphrase weight equals to the sum of the weights of its components. The following features are used for the calculation of word weights.

Word frequency feature. The proposed ranking algorithm uses the idea exploited in RAKE to weigh the candidates using the information on the word frequency in the obtained word sequences and not in the whole text. Additionally, we make use of the observation that, as opposed to single words, long phrases are more often contextually significant. As it is noticed, only a small group of keyphrases are single-word [19]. For this reason, we calculate word frequency as the frequency of its occurrence in multiword candidates that belong to a given text.

Position feature. The importance of considering the first occurrence of a keyphrase in a text is shown in the domain. It is intuitively reasonable that this feature should be particularly useful in the processing of long texts, such as research papers. It is not surprising, because it is in the title, abstract, and introduction where the main ideas and most keyphrases are [15]. Hence, the hypothesis is quite justified that the closer a keyphrase lies to the beginning of a document, the more important it is and it helps to give priority to keyphrases extracted from the beginning of a document. Actually, the works that achieved one of the top results in the SemEval 2010 competition (e.g. [15]) also exploit information about parts of texts from which phrase candidates are extracted. YAKE is reported to outperform RAKE, SingleRank and TextRank and it takes this feature into account as opposed to the mentioned algorithms.

We also use this feature so that SWaP is capable of processing both long and short texts. Opposite to other works in the area, we assume that the phrases that occur at the beginning of a text have the same priority over each other and over the keyphrases from the rest of the document. For the phrases that occur close to the starting point of the second part, the priority can be determined by measuring the distance from the document start or using a special coefficient.

C. Ranking algorithm.

Note that a keyphrase's weight equals the sum of its word weights. The two features above are combined as follows to calculate word v weight:

$$\begin{cases} weight(v) = tf(v)*x,\ ind(ph) < y \\ weight(v) = tf(v),\ ind(ph) > (y\text{-}1). \end{cases}$$

where $ind(ph)$ is the index for the phrase that contains the word, x and y are input parameters, and $tf(v)$ is the frequency of occurrence of a word v in multi-word candidates. We engage frequency data only from the obtained candidates and combine it with the observation that only a small group of keyphrases are single-word [19].

We introduce an additional condition for long and noisy documents (containing names of author affiliations, links to grants, etc.) that we apply for the processing of SemEval2010: if for at least one word belonging to a keyphrase, $tf(v) < 2$, then the weight of the entire keyphrase equals to 0. This condition does not apply to short texts, e.g. from the INSPEC dataset. Moreover, INSPEC dataset does not contain names of authors, affiliations, etc.

At the end of the ranking step, k top-ranked candidates are selected as keyphrases. If k is not specified then half the top-ranked candidates form the resulting keyphrase set.

D. Extended stoplist generation

We hold to the idea expressed in RAKE that defines phrase delimiters as high-frequency words that are often adjacent to keyphrases and rarely occur within them. These words are mostly represented by function words and we will denote them as stop words. The authors of RAKE find stop words using keyphrases from the gold standard of the training set. The algorithm proved its efficiency and is patented.

In the present study, we exploited and tested an alternative approach for the construction of extended stop-word lists. It involves finding the words that are often incorrectly included in keyphrase candidates. We use the same training set as in [20] to find these words.

Define two algorithms: SWEA (stop words Extraction Algorithm) and KE algorithm (Keyphrase Extraction Algorithm, which exploits stop words). SWEA iterates over the words in the train collection and measures the increase in performance quality of the KE algorithm produced by the inclusion of each specific word in the stoplist. If this improvement exceeds the threshold p, the word is labeled as a stopword. In the final step, all words labeled as stop words are added to the initial stoplist. Since the use of new stop words can change the specific features of the phrases extracted by the KE algorithm and it may produce wrong results for other words, SWEA proceeds to a new iteration. At the i-th iteration, SWEA iterates again over all words, except stop words, from the dictionary of the train collection and assesses whether the inclusion of each specific word contributes to the improvement in performance quality. For this purpose, KE algorithm uses a stoplist that includes all words obtained at the previous iterations.

E. Algorithm complexity.

Assuming that the extended stoplist is generated and the dataset is PoS-tagged, the time complexity of SWaP is $O(n * log(n))$, n – the number of words in a document. On one pass through the text, the following is accomplished with $O(n)$: a) construction of all candidates, b) word frequency calculation, and c) assignment of phrase position feature. In the ranking step, the weight estimation for each keyphrase requires $O(n)$ and the sorting of all keyphrases is $O(m * log(m)$ where m is the number of phrases. Since $m < n$, the overall complexity of this step and of the whole algorithm is denoted as $O(n * log(n))$.

F. Comparison with the baseline methods

The proposed SWaP method incorporates several effective properties of other algorithms. As in TextRank and SingleRank, we introduce the constraints on the PoS (nouns and

adjectives) that are allowed in the retrieved keyphrases. Also, similar to YAKE, SWaP takes into account the closeness to the beginning of the document during keyphrase weighting.

The keyphrase candidates extraction algorithm in SWaP is similar to the one used in RAKE. RAKE's difference is that it does not employ PoS restrictions and involves adjacent keywords [20]. As in RAKE, we generate and employ an extended list of stop words that act as delimiters between keyphrases. SWaP and RAKE use different stop-word list generation methods. The ranking procedure in both algorithms is comparable. Assuming that the stoplist is an input parameter and the texts are previously PoS-tagged, our approach is quite rapid and close to RAKE. Notice that when RAKE was compared to TextRank and the authors reported that it worked 6 times faster, the PoS tagging step in the TextRank algorithm was not taken into consideration [20].

V. DataSets

INSPEC. INSPEC [16] is one of the most famous datasets in the domain of keyphrase extraction. The subsets of the INSPEC dataset of research abstracts that we use in the experiments are as follows: INSPEC-Trial (1000 documents) and INSPEC-Test (500 documents).The abstracts are related to the following research fields: Computers and Control, Information Technology. For each abstract, there are two sets of keyphrases: (1) a controlled set of terms assigned by an expert indexer (terms restricted to the INSPEC thesaurus); (2) a set of uncontrolled terms (any suitable terms). The keyphrases from these sets can either appear or not in the abstracts. We rely on the uncontrolled set as the gold standard following many other works in the domain. Most of the phrases from this set are present in the abstracts as compared with the controlled set (76.2% as opposed to 18.1%) [16]. The following data provides a statistical information for the INSPEC-Test: document length in words (avg.:122, min.:23, max.:338), number of keyphrases (avg.:10, min.:2, max.:31). Following [20], we use the INSPEC-Trial dataset (1000 documents) to create stopword lists and INSPEC-Test to evaluate the proposed approach

SemEval 2010. The SemEval 2010 dataset [38] consists of 244 full scientific papers from the computer science domain collected from ACM (avg. 8,020 tokens per document). The dataset combines documents from 4 categories: C2.4 (Distributed Systems), H3.3 (Information Search and Retrieval), I2.11 (Distributed Artificial Intelligence-Multiagent Systems), and J4 (Social and Behavioral Sciences-Economics). This collection includes a total of 244 documents divided into three parts: Trial (40 documents, that are also included in TRAIN), Train (144 documents), and Test (100 documents). Each article in the dataset has several human-assigned gold standards (author-assigned, reader-assigned, and combined). We use the Test dataset during the evaluation process and the combined gold standard: reader-assigned and author-assigned keyphrases where the number of keyphrases on average is: 15 (min.:9, max.:29).

VI. Evaluation

We use micro-averaged exact match $F1$ to evaluate the quality of the proposed approach as well as the $Precision$ and $Recall$. An exact match means that an automatically extracted keyphrase is considered a true positive if the gold standard contains exactly the same phrase. If there is a semantically equivalent but visually distinct phrase, it is considered a false positive. This is an evaluation error as defined by [39]. It is one of four types of errors described in [39] that causes low-performance quality in keyphrase extraction algorithms. Here, it should be noticed that despite the fact that the domain has existed for a long time the quality performance of keyphrase extraction algorithms is still poor in terms of $F1$.

The specific aims of this study do not include reimplementation of the algorithms that are considered for comparison purposes. We refer to the results achieved by these algorithms that are available in the previous literature. Both comparison methods (reimplementation of the algorithms and comparison to the available published results) have their respective drawbacks that should be taken into account. Particularly, the reimplementation results can disagree with those available in publications as shown in [40]. In some cases, it is possible to find the cause of the divergence, and sometimes, additional investigation is needed [40].

If we perform a comparison with the results available in the literature, we face the problem that a correct comparison may not be always possible due to the below-listed circumstances that are often not considered by the authors in the domain. On the basis of the INSPEC collection, let us outline the main aspects that should be considered when comparing the results with those available in the literature:

- the INSPEC gold standard dataset contains keyphrases that may not occur in the texts. Some authors (e.g., [12], [20]) employ the default full version of the gold standard, other authors (e.g., [16], [21]) remove keyphrases that do not occur in the texts and use the short version.
- one set of studies presents the results obtained on the subset of 500 documents from INSPEC-Test. In another set, these 500 documents are taken not only from INSPEC-Test but from the whole pool of the INSPEC dataset. For instance, the authors of [41] pick 500 documents out of 2000 texts where all the gold standard phrases occur in the selected documents.
- most studies employ the uncounter gold standard set and some authors prefer to merge the counter and uncounter sets as in [42].
- most works employ the micro-average $F1$ as the quality measure, however, some studies (e.g., [11], [36] report on the use of the macro-average $F1$.

Additionally, it should be noted that some studies impose a constraint on the number of keyphrases that should be extracted (often k=5, 10, 15). In another set of works, there is no such limitation and, for each text, any number of keyphrases can be retrieved.

TABLE III. EVALUATION ON INSPEC DATASET. RESULTS FOR TEXTRANK AND SINGLERANK IN THE COLUMN " F RE-IMPL" REIMPLEMENTATION ARE TAKEN FROM [40]

INSPEC	F re-impl.	P	R	F
SWaP	-	**0.355**	**0.470**	**0.404**
RAKE	-	0.337	0.415	0.372 [20]
YAKE	-	-	-	0.316 [34]
TextRank	0.330	0.312	0.431	0.362 [12]
SingleRank	0.353	-	-	-

TABLE IV. EVALUATION ON SEMEVAL 2010 DATASET FOR 10-TOP KEYPHRASES. RESULTS FOR RAKE, TEXTTANK AND SINGLERANK ARE TAKEN FROM [34], [35]

SemEval 2010 @10	P	R	F
SWaP	**0.180**	**0.117**	**0.142**
YAKE	0.153	0.103	0.123
RAKE	0.007	0.004	0.005
TextRank	0.101	0.067	0.081
SingleRank	0.035	0.022	0.027

To avoid inaccuracies and perform an efficient comparison, we take the results of the reimplementation of the main algorithms described in [40] that are achieved with the best parameters. Here, the author's quality evaluation parameters coincide with ours: INSPEC-Test collection, the full uncontrolled version of the gold standard, no additional stemming of gold standard and extracted phrases, micro-average exact-match $F1$. These results are taken from [40] and presented in Table III column "re-impl." (this Table is in section 4 Experiments). In addition, it provides the results from the original papers.

In [40], the results obtained on the SemEval-2010 collection are not presented. They are given in [34], [35] and the reimplementation of the algorithms is available in the YAKE project. For each text, the top 10 keyphrases were selected. The results obtained in [34], [35] are presented in Table IV (this Table is in section 4 Experiments).

VII. EXPERIMENTS

A. Experiment description and results

The experiments are conducted in the following order and with the following settings:

- allowed PoS: nouns and adjectives.
- extended stoplists extraction using INSPEC-Trial dataset: the number of iterations i =2; parameter p=0.0001 in the first iteration, p =0.0005 in the second iteration;
- keyphrase extraction from the INSPEC-Test dataset with the obtained extended stoplist, undefined evaluation parameter k[1] and full uncountr gold standard, position feature parameters x=10, y=3000;
- keyphrase extraction from the SemEval-Test dataset with the obtained extended stoplist, evaluation parameter k=10, reader and author combined gold standards, position feature parameters x=10, y=3000.

Table III shows the results achieved on the INSPEC-Test dataset where the following notation is used. The column "re-implementation" tabulates the results with the best-performing setup reported in [40] for the re-implementation of the indicated algorithms. In Table IV, the results for SemEval 2010 (combined reader-author gold standard) are shown: for the case where 10 keyphrases should be extracted from a given text. The results achieved on both collections demonstrate that SWaP outperforms all baseline methods.

[1]Remind: k - is the number of keyphrases that the algorithm can extract. If k is not specified then the half top-ranked candidates form the resulting keyphrase set

B. Discussion

The results produced by SWaP improve the results achieved by the baseline algorithms. Moreover, SWaP uses a simple keyphrase extraction procedure that allows extracting a minimum number of keyphrase candidates, and a simple and computationally cheap ranking method. This is achieved through the incorporation of several important components: PoS constraints, position feature and stoplist. Table V summarizes the processing of each collection by SWaP and indicates the role each component plays in the extraction process.

Position. This feature is particularly useful for the processing of full research papers. It allows selecting candidates primarily from the starting parts of a document, such as the title, abstract, and introduction where keyphrase density is much higher than in the main body of the article. The latter increases the number of true positives among the candidates and, consequently, in the resulting keyphrase set. In this respect, YAKE has an advantage over RAKE, TextRank, and SingleRank and expectedly works better on long texts. For short texts, this position feature plays a less important role. It is easy to observe in the results reported in [34], [35] that the advantage of YAKE on short texts is less prominent. In the case of the INSPEC processing, the use of this feature does not have any effect on the quality, while for SemEval, it significantly impacts the performance quality.

The authors of [39], [40] point out that the increase in document length substantially affects the quality of the retrieved phrases and complicates the task. In the first place, it can be explained by the fact that a long text generates a large number of candidates, which negatively influences the results of the subsequent keyphrase selection. The authors of [43] remark that the quality of keyphrase candidates significantly impacts the results of the final ranking and, subsequently, determines the quality of the final keyphrase set. Following this idea, our task is to build a small set of high-quality candidates for a given text. For scientific articles, we achieve this by focusing on titles, abstracts, and introductions. The position feature helps us in it.

PoS. The application of PoS constraints affects less the results obtained on SemEval as compared to INSPEC, which in our opinion is due to the differences in document lengths. SemEval includes full texts, not only abstracts as INSPEC, which allows accumulating the necessary statistics on the

TABLE V. INFLUENCE OF THE MAIN FEATURES ON SWAP PROCESSING

	INSPEC			SemEval		
	P	R	F	P	R	F
All features	0.355	0.470	**0.404**	0.180	0.117	**0.142**
no position feature	0.355	0.470	0.404	0.046	0.031	0.037
no PoS	0.256	0.453	0.327	0.132	0.087	0.104
no stoplist	0.320	0.467	0.380	0.165	0.107	0.130
no stop & no PoS	0.222	0.419	0.291	0.114	0.076	0.091

occurrence of terms in multiword candidates. We assume that this statistic helps to increase the weight of the phrases consisting of nouns and adjectives. The data that can be collected from the abstracts on word frequency in candidate keyphrases is limited. We can state that the introduction of PoS constraints has a higher influence on the processing of short texts as compared to full texts.

StopList. The application of stoplists in RAKE and SWaP allows finding words that function as delimiters between keyphrases on the one hand, and on another hand, it enables to removal of the terms that are commonly present in texts and do not reflect the specifics of a given document. For scientific publications, these are such words as $paper$, $author$, $study$, $suppose$ and others. The use of a similar stoplist improves the quality of the algorithms. The results show that the constructed stoplist possesses the property of universality within one research field and in this case, both datasets are related to scientific publications. The latter is confirmed by the fact that the same stoplist built using INSPEC-Train is applied to both collections and in both cases, its application makes it possible to improve the quality of the keyphrase extraction algorithm.

To sum up, let us point out that this combination of features seems promising and justified, which is proved by the achieved results. Earlier these components functioned separately and now we incorporate all of them into the SWaP approach and show the advantages of this combination. This allows us to suggest a simple and effective keyphrase extraction algorithm, specifically, a simple and rapid approach to candidate keyphrase extraction and ranking.

VIII. CONCLUSION

In this paper, a keyphrase extraction algorithm is proposed that allows combining PoS restrictions with an automatically generated extended stoplist. The incorporation of these components together with a position feature into a simple candidate extraction and ranking procedure enables the enhancement of the state-of-the-art in the field. The algorithm extracts keyphrases from individual documents in an unsupervised manner and depends neither on the whole dataset nor on external knowledge bases, such as WordNet, Wikipedia, or pre-trained models. The algorithm affirms its effectiveness on two well-known datasets. Moreover, we show what features play a key role in processing short texts and which are particularly important when working with full texts.

REFERENCES

[1] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information Retrieval Journal*, vol. 2, pp. 303–336, 2000.

[2] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank, "Improving browsing in digital libraries with keyphrase indexes," pp. 81–104, 1999.

[3] B. Andrea, C. Claudio, and D. Massimiliano, "Full-subtopic retrieval with keyphrase-based search results clustering," *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp. 206–213, 2009.

[4] Z. Hua-Jun, H. Qi-Cai, C. Zheng, M. Wei-Ying, and M. Jinwe, "Learning to cluster web search results." *Proceedings of SIGIR'04*, pp. 210–217, 2004.

[5] F. Boudin, Y. Gallina, and A. Aizawa, "Keyphrase generation for scientific document retrieval," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 1118–1126. [Online]. Available: https://aclanthology.org/2020.acl-main.105

[6] H. Du, S. Thudumu, A. Giardina, R. Vasa, K. Mouzakis, L. Jiang, J. Chisholm, and S. Bista, "Contextual topic discovery using unsupervised keyphrase extraction and hierarchical semantic graph model," *Journal of Big Data*, vol. 10, 10 2023.

[7] A. Hulth and B. B. Megyesi, "A study on automatically extracted keywords in text categorization," p. 537–544, 2006. [Online]. Available: https://doi.org/10.3115/1220175.1220243

[8] J. Steve, L. Stephen, and P. G. W., "Interactive document summarization using automatically extracted keyphrases," *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pp. 1160–1169, 2002.

[9] E. D'Avanzo and B. Magnini, "A keyphrase-based approach to summarization: the lake system at duc-2005," 01 2005.

[10] G. Berend, "Opinion expression mining by exploiting keyphrase extraction," 2011.

[11] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," p. 1318–1327, 2009.

[12] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, D. Lin and D. Wu, Eds. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 404–411. [Online]. Available: https://aclanthology.org/W04-3252

[13] L. Wang and S. Li, "PKU_ICL at SemEval-2017 task 10: Keyphrase extraction with model ensemble and external knowledge," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, S. Bethard, M. Carpuat, M. Apidianaki, S. M. Mohammad, D. Cer, and D. Jurgens, Eds. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 934–937. [Online]. Available: https://aclanthology.org/S17-2161

[14] E. Marsi, U. Sikdar, C. Marco, B. Barik, and R. Sætre, "Ntnu-1@scienceie at semeval-2017 task 10: Identifying and labelling keyphrases with conditional random fields," 01 2017.

[15] A. Prasad and M.-Y. Kan, "Wing-nus at semeval-2017 task 10: Keyphrase identification and classification as joint sequence labeling," 08 2017.

[16] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '03. USA: Association for Computational Linguistics, 2003, p. 216–223. [Online]. Available: https://doi.org/10.3115/1119355.1119383

[17] T. Zesch and I. Gurevych, "Approximate matching for evaluating keyphrase extraction," 09 2009, pp. 484–489.
[18] S. Danesh, T. Sumner, and J. Martin, "Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction," 01 2015, pp. 117–126.
[19] L. Zhiyuan, L. Peng, Z. Yabin, and S. Maosong, "Clustering to find exemplar terms for keyphrase extraction." *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 257–266, 2009.
[20] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining: Applications and Theory*, pp. 1–20, 2010.
[21] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'08. AAAI Press, 2008, p. 855–860.
[22] E. Frank, G. Paynter, I. Witten, C. Gutwin, and C. Nevill-Manning, "Domain-specific keyphrase extraction," 07 1999.
[23] S. R. El-Beltagy and A. Rafea, "KP-miner: Participation in SemEval-2," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, K. Erk and C. Strapparava, Eds. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 190–193. [Online]. Available: https://aclanthology.org/S10-1041
[24] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," 2018.
[25] X. Liang, S. Wu, M. Li, and Z. Li, "Unsupervised keyphrase extraction by jointly modeling local and global context," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 155–164. [Online]. Available: https://aclanthology.org/2021.emnlp-main.14
[26] H. Ding and X. Luo, "Attentionrank: Unsupervised keyphrase extraction using self and cross attentions," pp. 1919–1928, 01 2021.
[27] T. Li, L. Hu, C. Sun, S. Li, and L. Chi, "Triplerank: An unsupervised keyphrase extraction algorithm," *Knowledge-Based Systems*, vol. 219, p. 106846, 02 2021.
[28] Y. Sun, H. Qiu, Y. Zheng, Z. Wang, and C. Zhang, "Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model," *IEEE Access*, vol. 8, pp. 10 896–10 906, 2020.
[29] H. Ding and X. Luo, "AGRank: Augmented graph-based unsupervised keyphrase extraction," in *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 (Volume 1: Long Papers)*, Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang, Eds. Online only: Association for Computational Linguistics, Nov. 2022, pp. 230–239. [Online]. Available: https://aclanthology.org/2022.aacl-main.19
[30] M. Song, P. Xu, Y. Feng, H. Liu, and L. Jing, "Mitigating over-generation for unsupervised keyphrase extraction with heterogeneous centrality detection," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 16 349–16 359. [Online]. Available: https://aclanthology.org/2023.emnlp-main.1017
[31] X. Jiang, Y. Hu, and H. Li, "A ranking approach to keyphrase extraction," 07 2009, pp. 756–757.
[32] A. John, L. Di Caro, and G. Boella, "A supervised keyphrase extraction system," 09 2016, pp. 57–62.
[33] J. Villmow, M. Wrzalik, and D. Krechel, "Automatic keyphrase extraction using recurrent neural networks," 07 2018, pp. 210–217.
[34] C. Ricardo, M. Vítor, P. Arian, J. A. Mário, N. Célia, and J. Adam, "A text feature based automatic keyword extraction method for single documents," *ECIR, Lecture Notes in Computer Science*, vol. 10772, pp. 684–691, 2018.
[35] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, "Yake! collection-independent automatic keyword extractor," in *European Conference on Information Retrieval*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4640960
[36] G. Tsatsaronis, I. Varlamis, and K. Nørvåg, "Semanticrank: Ranking keywords and sentences using semantic graphs." vol. 2, 01 2010, pp. 1074–1082.
[37] B. Sergey and P. Lawrence, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 1–7, 1998.
[38] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 21–26. [Online]. Available: https://aclanthology.org/S10-1004
[39] K. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art," vol. 1, 06 2014, pp. 1262–1273.
[40] K. S. Hasan and V. Ng, "Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art," in *Coling 2010: Posters*, C.-R. Huang and D. Jurafsky, Eds. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 365–373. [Online]. Available: https://aclanthology.org/C10-2042
[41] R. Wang, W. Liu, and C. Mcdonald, "Using word embeddings to enhance keyword identification for scientific publications," 06 2015, pp. 257–268.
[42] D. Mahata, J. Kuriakose, R. R. Shah, and R. Zimmermann, "Key2Vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 634–639. [Online]. Available: https://aclanthology.org/N18-2100
[43] W. You, D. Fontaine, and J.-P. Barthès, "An automatic keyphrase extraction system for scientific documents," *Knowledge and Information Systems*, vol. 34, 03 2012.