KRAKEN: A Novel Semantic-Based Approach for Keyphrases Extraction

Simone D'Amico[1][a], Lorenzo Malandri[2,3][b], Fabio Mercorio[2,3][c] and Mario Mezzanzanica[2,3][d]
[1]*Department of Economics, Management and Statistics, University of Milano-Bicocca, Milan, Italy*
[2]*Department of Statistics and Quantitative Methods, University of Milano-Bicocca, Milan, Italy*
[3]*CRISP Research Centre, University of Milan-Bicocca, Milan, Italy*

Keywords: Keyphrases Extraction, Keyphrases Evaluation, Keyphrases Benchmark Evaluation, Word Embeddings, Natural Language Processing.

Abstract: A research area of NLP is known as keyphrases extraction, which aims to identify words and expressions in a text that comprehensively represent the content of the text itself. In this study, we introduce a new approach called `KRAKEN` (**K**eyphrease ext**RA**ction ma**K**ing use of **E**mbeddi**N**gs). Our method takes advantage of widely used NLP techniques to extract keyphrases from a text in an unsupervised manner and we compare the results with well-known benchmark datasets in the literature. The main contribution of this work is developing a novel approach for keyphrase extraction. Both natural language text preprocessing techniques and distributional semantics techniques, such as word embeddings, are used to obtain a vector representation of the texts that maintains their semantic meaning. Through `KRAKEN`, we propose and design a new method that exploits word embedding for identifying keyphrases, considering the relationship among words in the text. To evaluate `KRAKEN`, we employ benchmark datasets and compare our approach with state-of-the-art methods. Another contribution of this work is the introduction of a metric to rank the identified keyphrases, considering the relatedness of both the words within the phrases and all the extracted phrases from the same text.

1 INTRODUCTION

Keyphrases refer to a set of relevant terms that provide a high-level description of a textual document. The Keyphrases Extraction task (KPE) defines a range of approaches and techniques that aim to identify keyphrases. The extraction of key phrases is of significant importance in the field of natural language processing (NLP), particularly in text summarization, content recommendation or topic modeling tasks.

The KPE task differs in *Keyphrases Assignment*, in which the most relevant phrases are identified based on a predefined dictionary of words or expressions, and *Keyphrases Extraction* involves directly identifying and extracting directly from the analyzed corpus. In this study, we define a new approach for extracting keyphrases directly from the text.

In the Keyphrases Extraction tasks context, we provide a formal definition of keyphrases: let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ the corpus consisting of n documents, $d_i = \langle t_1, t_2, \ldots, t_m \rangle$ a single document of length m consisting of a sequence of term t_j in the order of how they appear in the document itself. A keyphrases for a document d_i is a subsequence of terms in d_i: $kp_i = \langle t_l, t_{l+1}, \ldots, t_r : \forall\ 1 \leq l, l \leq r \leq m \rangle$, so in this context a keyphrase can also consist of only a single word.

The typical keyphrase extraction pipeline typically involves two steps:

1. Keyphrases extraction: In this step, specific algorithms are used to identify and extract a set of candidate keyphrases for a text.
2. Keyphrases ranking: Following the extraction step, a rank is applied to determine the best keyphrases among those extracted. This ranking can be used for evaluation purposes or to perform a specific task.

This paper presents KRAKEN (**K**eyphrease ext**RA**ction ma**K**ing use of **E**mbeddi**N**gs) a new approach for identifying and extracting keyphrases

[a] https://orcid.org/0009-0002-2820-0277
[b] https://orcid.org/0000-0002-0222-9365
[c] https://orcid.org/0000-0001-6864-2702
[d] https://orcid.org/0000-0003-0399-2810

D'Amico, S., Malandri, L., Mercorio, F. and Mezzanzanica, M.
KRAKEN: A Novel Semantic-Based Approach for Keyphrases Extraction.
DOI: 10.5220/0012179500003598
In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2023) - Volume 1: KDIR*, pages 289-297
ISBN: 978-989-758-671-2; ISSN: 2184-3228

from text as they appear in the text itself. A two-step evaluation process is also defined to rank the extracted keyphrases with the aim of identifying the most relevant ones. The results obtained are compared with other approaches for extracting keyphrases on five different datasets.

2 RELATED WORK

We provide an overview of the current state-of-the-art techniques in keyphrase extraction, highlighting their main characteristics. The works mentioned here will then be used as a term of comparison in evaluating the proposed method.

Keyphrase extraction techniques can be categorized into three main categories: *Deep Learning Techniques*, *Supervised Techniques* and *Unsupervised Techniques* keyphrase extraction. In these approaches, a keyphrase is treated as a sequence of tokens and incorporates information from previous ones (Merrouni et al., 2020). Supervised techniques involve training a classifier to identify keyphrases using labeled data. On the other hand, there are several unsupervised approaches, which can be categorized based on the selection of keyphrases: *text construction-based* or *relationship-based* approaches. In text construction-based approaches use statistical features such as TF-IDF (Term Frequency-Inverse Document Frequency), weighted co-occurrence matrices, or the semantic features of the text itself to identify keyphrases.

In relationship-based approaches, we still distinguish between *graph-based* or *topic-based* approaches, in these algorithms, it is assumed that the words that co-occur within a certain window in the text have some relationship. In graph-based approaches, graphs are created with words as nodes and two nodes are connected if the associated words co-occur within a fixed-size window. These graphs are then weighted, and node ranks are calculated using different algorithms like TextRank or PageRank. The highest-scoring word sequences are identified from these ranks to construct keyphrases. In topic-based approaches, words are assigned to topics, assuming that words within the same topic are related. Techniques such as Latent Dirichlet Allocation (LDA) or clustering techniques are commonly used in these methods. Typically, keyphrase identification begins by selecting words of specific parts of speech, in particular nouns and adjectives are taken as candidate sentences because they contain succinctly the key information (Kathait et al., 2017).

The selection of an appropriate approach for weighing and identifying keyphrases depends on the specific context and corpus being analyzed. For instance, in their work on a large corpus, Knittel et al. (Knittel et al., 2021) introduce a method called ELSKE. ELSKE efficiently extracts descriptive but potentially long sentences that occur unusually frequently. They extend the concept of TF-IDF and adapt it for large document collections. A stabilized version of TF-IDF is proposed to prevent the divergence of TF and IDF values for overly common words.

Chi et al. (Chi and Hu, 2021) propose ISKE, a PageRank-like method that weighs relationships between sentences by assuming strong causality between adjacent sentences. Rather than iterating through individual words, ISKE employs a graph-based method to weigh sentences based on the words within them. This approach reduces computational and temporal complexity.

Liu et al. (Liu et al., 2010) combine graph-based and non-graph-based algorithms, introducing TopicPageRank (TPR) that combines topic-based and graph-based ideas. TPR constructs a graph for each document using word co-occurrence statistics within a fixed window size, PageRank is then applied to the document's various topics to weigh words based on their importance within those topics.

Boudin (Boudin, 2018) introduces an approach that uses a multipartite graph structure (MUL) to encode topical information and keyphrase candidates. In this graph, nodes represent various keyphrase candidates, and edges connect nodes if the corresponding keyphrases belong to different topics. The edges are weighted based on keyphrase co-occurrence statistics. The TextRank algorithm ranks the nodes and identifies the most relevant keyphrases.

Campos et al. (Campos et al., 2018) propose YAKE, a feature-based method for keyphrase extraction. YAKE considers multiple features for each candidate keyphrase, such as its position or frequency within the text, whether it is capitalized, and the number of distinct words occurring before or after the term. These features are combined using a heuristic scoring approach to determine the best keyphrases.

Despite the widespread use of more recent models based on transformer architectures, such as BERT (Devlin et al., 2018), in NLP tasks, it was decided not to employ them in this specific work. The first reason is the intention to solely utilize the information present in the considered corpora. This led us to exclude all pre-trained models. A second motivation lies in the way the proposed approach identifies a keyphrase; it focuses on individual words during the keyphrase extraction without providing con-

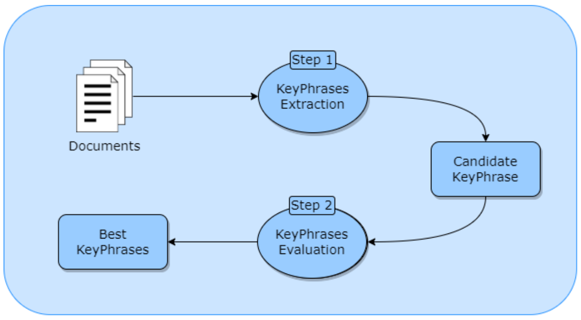


Figure 1: The keyphrases extraction task workflow.

textualization. Thus, the input for the model is not a sequence of tokens but a single token. In this regard, the use of the fastText word embeddings model proves to be more suitable. This is due to its training speed on the corpus and its efficiency in obtaining a vector representation of a single word.

3 KRAKEN: A NOVEL APPROACH TO KPE

KRAKEN introduces a novel approach for keyphrase extraction that leverages word embedding techniques. The method begins with preprocessing the text and removing stop words. The focus is then placed on nouns and adjectives, which are taken as candidate phrases because they succinctly contain the key information (Kathait et al., 2017). These words play a crucial role in defining the keyphrases. The approach operates iteratively by constructing a window around each noun or adjective. This window includes the preceding and succeeding words, forming potential keyphrases. In this context, a keyphrase corresponds to the window constructed around a specific word. To determine the construction of the window, KRAKEN utilizes a word embeddings model. It evaluates the proximity between the vectors of the current window and the words that can be added to the window. This proximity analysis aids in determining the relevant words for the keyphrase construction.

As shown in figure 1, the keyphrase extraction task implementation follows a two-step process: the initial step involves identifying and extracting candidate keyphrases from the text, while the subsequent phase focuses on selecting the most suitable keyphrases using specific metrics for evaluation purposes, comparing them with the baseline.

We provide two versions of KRAKEN: one employing the Pearson correlation index for constructing and evaluating the windows, and another version utilizing the Cosine similarity. Both versions are compared with state-of-the-art results in the baseline section.

3.1 Step 1: Phrases Identification and Extraction

In the keyphrase extraction phase, two main objectives have been identified: (i) performing preprocessing to reduce the noise present in the texts and provide a clean corpus for the word embedding model, and (ii) actually extracting the keyphrases.

We apply state-of-the-art preprocessing steps. Firstly, all letters in the text are converted to lowercase. Then, stop words, numbers, punctuation, accent marks, diacritics, and HTML tags are removed. Additionally, lemmatization is performed on the corpus to obtain the base form of each word by removing inflectional endings. As the last preprocessing step, to increase the capacity of the word embedding model, *n*-gramms have been identified in the text, in particular uni-grams, bi-grams and tri-grams. By applying these preprocessing steps, KRAKEN aims to enhance the quality of the corpus and improve the keyphrase extraction process (Mezzanzanica et al., 2015; Mezzanzanica et al., 2012; Boselli et al., 2014).

Once the corpus has been cleaned, a word embedding model is trained, which will be utilized for creating the windows. Part-of-speech (POS) tagging is applied to identify nouns and adjectives, referred to as *anchor* words, that are used for keyphrases identification.

The algorithm starts by analyzing the preceding part of the text before the anchor, forming phrases around it. Then, the same procedure is applied to the subsequent part of the text. This iterative process extends to the words preceding and following the current window. In the base case, when the window consists only of the anchor, the relatedness between the vectors of the two words is calculated. If the relatedness exceeds a fixed threshold, the first word is included in the window. However, if the relatedness is less than the threshold, the process stops.

For the words following the first one, the window constructed thus far is taken into consideration. For each new word encountered, the relatedness between its vector and that of the current window is calculated. If the relatedness is greater than the value obtained in the previous iteration, the word candidate is added to the window, and the process continues to the next word. By iteratively incorporating relevant words based on their relatedness to the window, KRAKEN aims to construct meaningful phrases around the anchor words identified through POS tagging.

Let $\vec{v}_{kp_{i-1}}$ the embedding representation of the keyphrase kp_{i-1} constructed in iteration $i-1$, $\vec{v}_{w_i}$ is the embedding representation of a word w_i considered in iteration i, and α_i represent the measure of relatedness between kp_{i-1} and w_i. At i-th iteration, the keyphrase is constructed as follows:

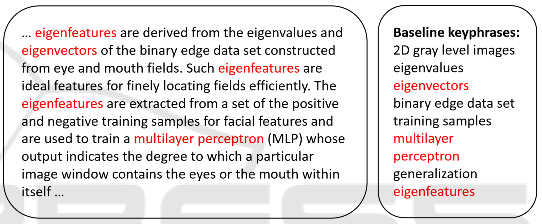
1. Calculate the relatedness α_i between $\vec{v}_{kp_{i-1}}$ and $\vec{v}_{w_i}$.
2. If α_i is greater than the relatedness obtained in the previous iteration α_{i-1}, proceed to the next step. Otherwise, stop constructing the keyphrase.
3. Add w_i to the keyphrase.
4. Repeat steps 1-3 for the subsequent words, considering the updated keyphrase at each iteration.
5. Stop constructing the keyphrase when a word is encountered that decreases the relatedness or when there are no more new words to add to the windows.

By following these steps, the algorithm constructs the keyphrases iteratively, incorporating words that increase the relatedness and stopping when the relatedness decreases or the desired length is reached. The definition of the relatedness α depends on the version of `KRAKEN` used: in the case of Pearson's correlation then α is equal to the coefficient $\rho_{\vec{v}_{kp_{i-1}},\vec{v}_{w_i}}$ and in the case of the cosine similarity α is the cosine of the angle between these vectors.

```
Data: a document $d$
Result: the set of keyphrases for the input
        document;
$kp_d \leftarrow \emptyset$;
for $word$ in $d$ do
    if $word$ is nouns or $word$ is adjective then
        $kp_L \leftarrow extract\_kp(word, d)$;
        // Extract kp on the left
         side;
        $kp_R \leftarrow extract\_kp(word, d)$;
        // Extract kp on the right
         side;
        $kp_{word} \leftarrow kp_L \cup word \cup kp_R$;
        $kp_d \leftarrow kp_d \cup kp_{word}$
    end
end
return $kp_d$
```

Algorithm 1: Extract kps for a text.

There are two stopping criteria for the construction of the windows: (i) if the new relatedness is below a certain threshold t no first word is added to the window at the beginning and it remains composed of only the anchor, and (ii) if the new relatedness is lower than the previous iteration. In this second scenario, this means that when a word w is encountered that decreases the relatedness, the construction of the phrase is halted, even if adding subsequent words after w could result in a higher relatedness. To account for this, the stop condition (ii) has been modified: even if the relatedness between a word w_i and the current window is lower than the relatedness in the previous iteration, w_i is still included in the windows and the process continues. However, if another subsequent word with lower relatedness is encountered, the process is stopped. Therefore, only one word is allowed which does not increase the closeness during the construction process of the phrase. The algorithm 1 shows how the keyphrases are extracted from a text: for each noun or adjective in the text, the keyphrase is built around it iterating first on the previous part of the text and then on the next one. The result kp_d is the set of keyphrases for the document d in input. Figure 2 shows the result of identifying keyphrases from a text.



… eigenfeatures are derived from the eigenvalues and eigenvectors of the binary edge data set constructed from eye and mouth fields. Such eigenfeatures are ideal features for finely locating fields efficiently. The eigenfeatures are extracted from a set of the positive and negative training samples for facial features and are used to train a multilayer perceptron (MLP) whose output indicates the degree to which a particular image window contains the eyes or the mouth within itself …

Baseline keyphrases:
2D gray level images
eigenvalues
eigenvectors
binary edge data set
training samples
multilayer
perceptron
generalization
eigenfeatures

Figure 2: An example of keyphrases extracted from a text.

3.2 Step 2: Phrases Evaluation

After the keyphrase extraction step, a set of candidate phrases is obtained for each examined text. The next step involves evaluating each individual window and assigning it a score. Only the phrases with the highest scores are selected for evaluation with the baseline. The scoring process utilizes the same word embedding model that was used to create the windows.

The candidate phrases are divided into two categories: *short keyphrases* composed of a single word and *long keyphrases* composed of multiple words. Different measures are applied depending on the type of keyphrase, namely within-window score and between-window score.

The within-window score (WW) is applied only to windows longer than one word to avoid considering windows too long. The average of the relatedness between all the pairs of words in the window is calculated, if this average is higher than a fixed threshold, the keyphrase proceeds to the next evaluation step; otherwise, it is discarded. The within-window score measures the internal cohesion of phrases in terms of the coherence among their words. The threshold value also determines the number of long keyphrases

Table 1: Details of datasets.

Dataset	# doc	Avg. text words	Avg. keyphrases	Avg. single word KP	Avg. multiple word KP
Inspec	2000	124.36	14.11	2.32	11.79
KDD	755	190.7	4.1	1.04	3.05
SemEval2010	243	8032.55	15.58	3.12	12.45
Nguyen2007	209	5121.67	12.01	3.31	8.69
WWW	1330	82.04	4.82	1.65	3.16

that will be filtered out: the higher the threshold the more keyphrases will be eliminated. This is an advantage in that it reduces the number of keyphrases with terms with low relatedness, but it may also exclude some windows present in the baseline. The within-window score for a keyphrase i is defined as:

$$WW_{kp_i} = \frac{1}{|kp_i^{(2)}|} \sum_{\substack{w_n, w_m \in kp_i \\ (w_n, w_m) \in kp_i^{(2)}}} \alpha(w_n, w_m) \qquad (1)$$

Where $kp_i^{(2)}$ is the set of all two-word combinations obtained from the set of words that make up kp_i, α can be either the Cosine similarity or the Pearson's correlation index and w_n and w_m are two words belonging to kp_i.

The long keyphrases that have passed the within-window score (WW) evaluation are combined with the single-word phrases. These merged phrases are then evaluated using the between-window score (BW). For each candidate keyphrase in a specific document, the average correlation with all other windows in the same document is calculated. This allows assigning a score to each window and selecting the most relevant ones.

The between-window score (BW), denoted as BW_{kp_i}, represents the ability of a keyphrase i to have a similar meaning to all other keyphrases extracted from the same text. A higher BW score indicates that the keyphrase effectively captures the essence of the text from which it is extracted. The BW score for a keyphrase i is defined as:

$$BW_{kp_{i,d}} = \frac{1}{|kp_d| - 1} \sum_{\substack{kp_{j,d} \in kp_d \\ kp_{i,d} \neq kp_{j,d}}} \alpha(kp_i, kp_j) \qquad (2)$$

Where kp_d is the set of all candidates keyphrases for the document d and, again, α can be the Cosine similarity or the Pearson's correlation index.

The algorithm 2 shows how to order the keyphrases by the between-window score, after filtering the long keyphrases with a within-window score lower than the threshold th_{WW}.

4 BASELINE EVALUATION

The obtained results from applying the two versions of `KRAKEN` to the benchmark datasets are presented and compared with the results of other approaches proposed in the literature for keyphrase extraction.

The baseline datasets used for evaluation are widely used in the NLP literature for assessing various NLP tasks. Specifically, five English datasets were considered, which are available on GitHub[1]. Each dataset contains a set of documents, and for each document, a list of keyphrases, manually identified by human annotators, is provided. Table 1 presents some information about each dataset.

```
Data: a list ok keyphrases $kp_d$ for a
      document $d$, the threshold $th_{WW}$ to
      filter multiple words keyphrases
Result: the ranking of keyphrases based on
        between-window score;
for $kp$ in $kp_d$ do
    if $kp$ is a multi-words keyphrases then
        $WW_{kp} \leftarrow$ Compute Eq. (1);
        if $WW_{kp} \geq th_{WW}$ then
            $BW_{kp} \leftarrow$ Compute Eq. (2);
        end
    else
        $BW_{kp} \leftarrow$ Compute Eq. (2)
    end
end
// Ordered according to BW score
   for each kp;
$sorted\_kp \leftarrow oreder(kp_d)$ ;
return $sorted\_kp$
```

Algorithm 2: Calculation of the score for each keyphrases.

SemEval2010 (Kim et al., 2010) consists of 244 scientific papers extracted from the ACM Digital Library. The papers belong to four different computer science research areas: distributed systems; information search and retrieval; distributed artificial in-

[1]See at https://github.com/LIAAD/KeywordExtractor-Datasets

telligence and social and behavioral sciences. The keyphrases were provided by the authors or by the editors themselves. The *KDD* (Gollapalli and Caragea, 2014) collection is based on the abstracts of papers collected from the ACM Conference on Knowledge Discovery and Data Mining (KDD) published during the period 2004-2014, with a total of 757 documents. The keywords of these papers are author-labeled terms. The *Nguyen2007* (Nguyen and Kan, 2007) is a dataset composed of 211 scientific conference papers. The gold keywords were manually assigned by student volunteers who were each given three papers to read. Similarly to the KDD, the *WWW* (Gollapalli and Caragea, 2014) collection of 1330 documents based on the abstracts of papers collected from the World Wide Web Conference (WWW) published during the period 2004-2014. The *Inspec* (Hulth, 2003) dataset consists of 2000 abstracts of scientific journal papers in computer science collected between the years 1998 and 2002. The keywords in this dataset are of two types: keywords that appear in the Inspec thesaurus but may not appear in the document and keywords that are freely assigned by the editors. Table 1 shows some characteristics of the datasets such as the average length of texts and the average number of keyphrases per text.

For the evaluation, the performance measures considered are *precision@k*, *recall@k*, *F_1-measure@k* where k indicates the number of best keyphrases considered, we use k $= 5, 10$. *precision* measures the accuracy of the system in identifying keyphrases, *recall* indicates how complete the system is in extracting known keyphrases, and F_1-measure is the harmonic mean of precision and recall. Two variants of KRAKEN are proposed: KRAKEN$_{cos}$, which utilizes Cosine similarity for extracting and ranking phrases, and KRAKEN$_{pear}$, which uses the Pearson correlation index.

4.1 Thresholds Optimization

KRAKEN relies on several values that impact its outcomes, including the training parameters of the word embedding models used for constructing windows. The proposed architecture utilizes fastText (Bojanowski et al., 2017) with the CBOW algorithm, a learning rate of 0.1, 300 as vector size and 100 train epochs. These parameter choices are based on the findings of (Giabelli et al., 2020; Giabelli et al., 2022), where job advertisements were processed to identify occupations and job skills. Although the scenario differs, as we focus on generic key phrases instead of skills, the similarity in the nature of the tasks justifies the selection of these parameters.

Other parameters that have a direct impact on performance are the threshold values for window construction (th_{win}) and within-window threshold for selecting the long keyphrases (th_{WW}). The various datasets were divided into two portions: a validation set containing 15% of the texts from the original dataset was used to optimize the values of the thresholds, and the remainder was used for the final comparison. To identify the optimal threshold values, a grid search was performed, considering values from 0 to 0.9 with a step of 0.1 for both thresholds. The evaluation metrics used for this search were F_1@5 and F_1@10. The results of the grid search are presented in Table 2. For each dataset and value of k, the threshold values that produce the highest F_1-scores are identified. These parameters are then used to compare with other state-of-the-art methods. The performances of both versions are very similar, both in terms of F_1-scores and optimal threshold values. It can be observed that in all datasets, except for WWW, the best value of th_{win} is very low. This implies that it will be easier to create keyphrases that contain more than one word. On the other hand, for the WWW dataset, the best threshold value is 0.9, indicating that many

Table 2: Results of the grid search with the best threshold values and their corresponding F1@k.

		KRAKEN$_{cos}$			**KRAKEN$_{pear}$**		
		best th_{win}	best th_{WW}	F_1@k	best th_{win}	best th_{WW}	F_1@k
k=5	Inspec	0.1	0.1	**11.92**	0.1	0.1	11.9
	KDD	0.1	0.2	17.30	0.1	0.2	**17.44**
	Nguyen2007	0.1	0.9	**7.2**	0.1	0.9	7.16
	SemEval2010	0.1	0.9	**3.93**	0.1	0.9	**3.93**
	WWW	0.9	0.1	27.2	0.9	0.1	**27.5**
k=10	Inspec	0.1	0.1	**18.72**	0.1	0.1	18.69
	KDD	0.1	0.1	23.1	0.1	0.1	**23.65**
	Nguyen2007	0.1	0.9	8.97	0.1	0.9	**9.16**
	SemEval2010	0.1	0.9	**5.1**	0.1	0.9	5.04
	WWW	0.9	0.1	**29.3**	0.9	0.1	**29.3**

Table 3: Performance@5 on datasets.

Dataset	**Eval**	**MUL**	**TPR**	**ISKE**	**Yake**	**KRAKEN**$_{cos}$	**KRAKEN**$_{pear}$
Inspec	P@k	3.9	2.7	4	1.4	**13.2**	**13.2**
	R@k	4.9	3.7	5.3	2	**9.3**	**9.3**
	F1@k	4.1	2.9	4.2	1.5	**10.5**	**10.5**
KDD	P@k	10.1	8.1	12	3.1	15.0	**15.3**
	R@k	12.2	9.7	14.3	4	15.0	**15.2**
	F1@k	10.7	8.5	12.3	3.4	15.0	**15.3**
Nguyen2007	P@k	**14.1**	10.2	12.2	10.1	11.9	11.9
	R@k	**17.7**	12.8	15	12.6	7.3	7.3
	F1@k	**15.3**	11	13.1	10.9	8.7	8.7
SemEval2010	P@k	**8.7**	5.9	7.9	4.3	7.1	7.1
	R@k	**11.9**	8.4	11.3	6.1	3.5	3.5
	F1@k	**9.7**	6.7	9	4.9	4.6	4.6
WWW	P@k	12.2	9.4	12.8	4.4	24.3	**24.4**
	R@k	12.9	10.2	13.9	5	24.3	**24.4**
	F1@k	12	9.3	12.7	4.5	24.3	**24.4**

keyphrases will be composed of a single word. Regarding the threshold value th_{WW}, in the case of Inspec, KDD, and WWW, it is low. Therefore, in the ranking of keyphrases, more keyphrases composed of multiple words will be considered. On the other hand, for Nguyen2007 and SemEval2010, the threshold value is high, resulting in a larger number of these keyphrases being filtered out. In summary, the choice of threshold values has an impact on the composition and ranking of keyphrases. Lower values favor the inclusion of multi-word keyphrases, while higher values result in the filtering of more keyphrases. However, it is important to note that the optimal threshold values vary across datasets, indicating the need for dataset-specific parameter tuning.

The results indicate that there is a relationship between the best threshold values of th_{WW} and the average number of words in each document. Specifically, the analysis reveals a direct proportional relationship between these two variables. This relationship is supported by the calculation of Spearman's rank correlation coefficient. For $F_1@5$, the coefficient yielded a value of $\rho = 0.94$ with a p-value of 0.01. This indicates a strong positive correlation between the variables, suggesting that as the average number of words in a document increases, the best threshold value for achieving optimal performance also tends to increase. These findings allow us to reject the null hypothesis (H_0) of non-correlation with a 95% confidence level. In the case of $F_1@10$, the coefficient was $\rho = 0.86$ with a p-value of 0.057. Although the value of ρ is high, it is important to note that the p-value is still greater than 0.05.

The use of a high threshold, in this case, allows having very cohesive phrases, that are formed by terms that are very correlated with each other; in this way, it is possible to better filter the noise produced by the numerous non-relevant phrases. On the contrary, in short texts, the relevant expressions are limited and therefore having a lower threshold avoids discarding those phrases with low within-window correlation. The formal analysis of the correlation between average document length and th_{WW} values provides further support for the logical explanation of this phenomenon. It is observed that as the length of the text increases, the possibility of having a lower number of relevant terms compared to the total number of words also increases. This implies that the document contains more noise.

In such cases, using a higher threshold value for th_{WW} allows for the formation of very cohesive phrases. These phrases are composed of terms that have high relatedness to each other. Using a high threshold, it becomes possible to effectively filter out the noise generated by numerous non-relevant phrases. On the other hand, in shorter texts, the number of relevant expressions is limited. Therefore, using a lower threshold value prevents the exclusion of phrases with low within-window relatedness. In conclusion, the analysis suggests that the characteristics of the datasets, specifically the average number of words, play a role in determining the optimal threshold values for achieving high performance in keyphrase extraction.

5 RESULT

KRAKEN, with both its versions and the optimal values for the thresholds, was compared with four other State-of-the-Art approaches presented in section 2: MUL, TPR and ESKE, three graph-based methods

Table 4: Performance@10 on datasets.

Dataset	Eval	MUL	TPR	ISKE	Yake	$\texttt{KRAKEN}_{cos}$	$\texttt{KRAKEN}_{pear}$
Inspec	P@k	2.9	2.6	3.3	1.3	**16.2**	**16.2**
	R@k	7.1	6.5	8	3.5	**15.4**	15.3
	F1@k	3.9	3.6	4.4	1.8	**15.7**	**15.7**
KDD	P@k	7.4	7.4	9.1	3.5	21.6	**21.9**
	R@k	16.8	16.4	22	8.8	21.6	**21.9**
	F1@k	9.7	9.7	12.2	4.8	21.6	**21.9**
Nguyen2007	P@k	9.3	8.4	10.8	9.4	11.4	**11.5**
	R@k	23.4	20.5	**25.4**	23.5	10.4	10.5
	F1@k	13.1	11.7	14.9	**18**	10.8	10.8
SemEval2010	P@k	5.9	5.7	6.1	3.7	**6.6**	6.5
	R@k	**15.7**	14.9	**15.7**	10.7	5.8	5.7
	F1@k	8.4	8	**8.6**	5.4	6.2	6.0
WWW	P@k	8.7	8.5	10.2	3.9	**28.6**	**28.6**
	R@k	17.1	16.7	19.8	8.6	**28.6**	**28.6**
	F1@k	10.8	10.5	12.6	5.1	**28.6**	**28.6**

and Yake, a feature-based method.

Based on the evaluation of performance@5, as shown in Table 3, the version of KRAKEN using Pearson's correlation index performs the best overall for the Inspec, KDD, and WWW datasets, while the version using cosine similarity is the second best in terms of performance. However, for the other two datasets, the precision of KRAKEN is in line with the other approaches but fails to achieve the results of MUL.

Moving on to performance@10, shown in Table 4 the values improve further. For the Inspec, KDD, and WWW datasets, the performance of KRAKEN is clearly superior to the other approaches. In the case of the Nguyen2007 and SemEval2010 datasets, KRAKEN achieves the highest precision, and this improvement in precision also leads to an increase in the F1-measure for these two cases.

In summary, based on performance@5, KRAKEN using Pearson's correlation index emerges as the best option for the Inspec, KDD, and WWW datasets, while cosine similarity performs well as the second-best choice. For performance@10, KRAKEN excels in the Inspec, KDD, and WWW datasets, and achieves the highest precision in the Nguyen2007 and SemEval2010 datasets, resulting in improved F1-measure as well.

6 CONCLUSION

In this work, we presented KRAKEN, a novel approach for keyphrase extraction from texts. We propose and design a new method that exploits word embeddings to identify keyphrases, taking into account the relationship among words in the text. We have introduced two different versions, namely $\texttt{KRAKEN}_{cos}$ and $\texttt{KRAKEN}_{pear}$, which use cosine similarity and Pearson correlation index, respectively, to identify and evaluate the key phrases.

For the evaluation, we utilize benchmark datasets and compare our approach with the other four methods using performance@k metrics. Additionally, we introduce a metric to rank the keyphrases, considering the correlation of words within the phrases and among all the extracted phrases from the same text. By combining word embeddings and correlation measures, our approach aims to improve the accuracy and effectiveness of keyphrase extraction, offering a comprehensive and robust method for extracting meaningful keyphrases from texts.

One of the key features of KRAKEN is the separation of the keyphrase extraction step from the ranking step. Unlike other approaches, where the ranking of phrases is determined at the time of extraction, KRAKEN allows for modularity and interchangeability, in KRAKEN the weighing process is independent of the extraction phase. This means that you can identify phrases using window techniques and their correlation, and then weigh them using co-occurrence statistics of the words within the phrases. Alternatively, you can first identify phrases using other methods and then weigh them using the two correlation measures defined in KRAKEN.

From the evaluation, both proposed versions of KRAKEN achieve similar performance, with the version utilizing Pearson's correlation index being the best in some cases. KRAKEN in both its versions attains the highest performance on the Inspec, KDD, and WWW datasets. Additionally, it achieves the best precision@10 on the remaining two datasets. These findings highlight the effectiveness of KRAKEN in keyphrase extraction tasks across various datasets.

REFERENCES

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2014). A policy-based cleansing and integration framework for labour and healthcare data. In Holzinger, A. and Jurisica, I., editors, *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics - State-of-the-Art and Future Challenges*, volume 8401 of *Lecture Notes in Computer Science*, pages 141–168. Springer.

Boudin, F. (2018). Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., and Jatowt, A. (2018). Yake! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval*, pages 806–810. Springer.

Chi, L. and Hu, L. (2021). Iske: An unsupervised automatic keyphrase extraction approach using the iterated sentences based on graph method. *Knowledge-Based Systems*, 223:107014.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Nobani, N. (2022). Embeddings evaluation using a novel measure of semantic similarity. *Cognitive Computation*, 14(2):749–763.

Giabelli, A., Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2020). NEO: A tool for taxonomy enrichment with new emerging occupations. In Pan, J. Z., Tamma, V. A. M., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., and Kagal, L., editors, *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 568–584. Springer.

Gollapalli, S. D. and Caragea, C. (2014). Extracting keyphrases from research papers using citation networks. In *Twenty-eighth AAAI conference on artificial intelligence*.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223.

Kathait, S. S., Tiwari, S., Varshney, A., and Sharma, A. (2017). Unsupervised key-phrase extraction using noun phrases. *International Journal of Computer Applications*, 162(1):1–5.

Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. (2010). Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Knittel, J., Koch, S., and Ertl, T. (2021). Elske: Efficient large-scale keyphrase extraction. *arXiv preprint arXiv:2102.05700*.

Liu, Z., Huang, W., Zheng, Y., and Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 366–376.

Merrouni, Z. A., Frikh, B., and Ouhbi, B. (2020). Automatic keyphrase extraction: a survey and trends. *Journal of Intelligent Information Systems*, 54(2):391–424.

Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2012). Data quality sensitivity analysis on aggregate indicators. In Helfert, M., Francalanci, C., and Filipe, J., editors, *DATA 2012 - Proceedings of the International Conference on Data Technologies and Applications, Rome, Italy, 25-27 July, 2012*, pages 97–108. SciTePress.

Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2015). A model-based approach for developing data cleansing solutions. *Journal of Data and Information Quality (JDIQ)*, 5(4):1–28.

Nguyen, T. D. and Kan, M.-Y. (2007). Keyphrase extraction in scientific publications. In *International conference on Asian digital libraries*, pages 317–326. Springer.