IEEE Access
Multidisciplinary Rapid Review Open Access Journal

Received 8 November 2023, accepted 13 December 2023, date of publication 18 December 2023, date of current version 27 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3344032

RESEARCH ARTICLE

A Comparative Assessment of Unsupervised Keyword Extraction Tools

MOHAMMAD NADIM[1], DAVID AKOPIAN[2], (Senior Member, IEEE), AND ADOLFO MATAMOROS[3]
[1]Center for Cybersecurity Innovation, Texas A&M University-Central Texas, Killeen, TX 76549, USA
[2]Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA
[3]School of Civil and Environmental Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding author: Mohammad Nadim (mohammad.nadim@tamuct.edu)

ABSTRACT The extraction of keywords is a critical task in natural language processing and information retrieval. It has become increasingly important in a wide range of applications, from search engines and e-commerce platforms to news and social media analysis. However, the evaluation of keyword extraction methods remains a challenging task due to the diverse range of data types and contexts used, as well as the complexity of the methodologies and techniques involved. In this regard, this study reviews the prior surveys on keyword extraction methods to comprehend the fundamental principles, difficulties in keyword extraction, and benchmark datasets. The reliability of evaluation techniques and an examination of their flaws remain two of the largest problems. Hence, this study includes the literature that performed a comparative analysis of popular keyword extraction methods. Furthermore, in this paper, we present a comparative evaluation of open-source unsupervised keyword extraction tools, analyzing their performance across a range of data types and under different testing conditions. The experimental analysis shows that, in terms of f-score, KPminer performs most consistently for different text lengths while KeyBERT(mmr) outperforms other tools. Considering the execution time, RAKE and YAKE are the fastest tools. Though graph-based tools tend not to perform well on long text, TopicRank and MultipartiteRank perform very well on long text as they use topics as nodes of the graph, which is another finding of this study. By highlighting the key factors that influence the performance of keyword extraction tools, our analysis contributes to directing the reader in selecting suitable keyword extraction tools for different applications.

INDEX TERMS Information retrieval, keyword extraction, unsupervised learning.

I. INTRODUCTION
Data is the most precious resource in the modern world. The vast data holds valuable information that researchers can harness for a multitude of objectives. There are two main techniques to extract key insights or to present data insights for textual data. The first choice is manual analysis, which comprises processing the data manually and starting to log the information that seems important [1]. Automated text analysis is a different approach that uses computer tools to perform text analysis. The latter method is more appealing due to the vast amounts of data involved because it will be impossible to manually process a significant volume of text. Automatic Keyword/key-phrase Extraction (KE) is a frequently used method to annotate articles to reflect their main content [2]. It is extremely important in the information retrieval field because keywords from any piece of data can aid in determining whether the data source is appropriate and pertinent to other important domains and/or subjects.

We can categorize keyword extraction methods into unsupervised and supervised approaches, depending on the presence of a training set. The distinction between them is whether the learning process includes a labeled training set. The supervised methods convert the keyword extraction task into a classification problem [3] or regression problem [4]. It employs the trained model to ascertain whether a candidate word in a text is a keyword after training the model on the labeled training set. The supervised methods typically

The associate editor coordinating the review of this manuscript and approving it for publication was Fu Lee Wang.

outperform the unsupervised methods [5]. However, the supervised methods involve manually labor-intensive labeled training sets [6]. Unsupervised methods became more popular as they are free of human intervention. Considering the characteristics of unsupervised KE methods, we can broadly classify them into statistics-based, graph-based, and deep learning-based. Although there has been a significant amount of proposed KE methods, not all of them have an implementation as open-source software applications or tools to integrate them into other applications for automating the process of identifying important keywords. A well-implemented KE tool can improve the accuracy of downstream applications that rely on keyword extraction as a preprocessing step and can significantly enhance the productivity, accuracy, and scalability of developers working with text data. Yet it is incredibly challenging to pick the right KE tool for an application due to factors including specific task requirements, the text's domain and genre, the volume of the text, the quality of the text, the intended output, and the available computational resources. Moreover, fine-tuning the hyperparameters of KE tools can optimize the output. In this paper, we evaluate open-source unsupervised KE tools under different conditions and present their performance analysis across a range of data types. The effect of hyper-parameter tuning, data preprocessing, changes in the number of extracted keywords, execution time metric, and exact Vs. partial matching of keywords is experimented within this study. The purpose of this study is to introduce the widely used open-source unsupervised KE methods, as seen in [7], [8], and [9], but the distinct contribution of this study includes:

1) Surveying the survey studies. There is a significant interest in keyword extraction with many methods and tools. There were several attempts to survey the KE methods, and as the research evolves, "surveying surveys" would help to see a broader perspective.
2) A broad discussion of KE insights to give the readers a clear understanding of evaluation metrics, comparing supervised vs unsupervised methods, the impact of assigned keywords, the effect of execution time and data quality, simultaneous learning, and benchmark datasets.
3) A comparative assessment of contemporary open-source unsupervised KE tools that summarizes their underlying techniques, finds out their drawbacks, and evaluates them on different conditions. This study will direct the readers in selecting or developing a strategy that is suitable for the application they are aiming for.

This study builds upon the research and findings originally presented in the author's Ph.D. dissertation [10]. The rest of this paper is structured as follows; Section II presents the motivations and limitations of this study. The research methodology of this study is briefly discussed in Section III. Background in Section IV covers the existing survey studies on KE methods and state-of-the-art literature that comparatively evaluates the KE methods. In this section, the intensive knowledge gathered about the KE methods is briefly discussed. Section V briefly discusses the open-source unsupervised KE tools selected for this study and summarizes them. The experimental setup and results are discussed in Section VI. Finally, the concluding remarks are outlined in Section VII.

II. MOTIVATIONS AND LIMITATIONS OF THIS STUDY

Because there is so much data available today, researchers have dedicated their expertise and resources to the development of tools that can help the community when undertaking keyword extraction tasks. While some of these tools are general in nature, others are tailored to specific issues or domains. Evaluation of the KE tool is still a challenge. Numerous methods have been developed for and evaluated on a few specific categories of documents. Comparing the methodologies is difficult because the types of data used in each methodology differ. Prior studies on evaluating KE tools lack comparative investigating key factors that can influence the performance including hyperparameter tuning, data preprocessing, changes in the number of extracted keywords, execution time metric, and exact Vs. partial matching of the extracted keywords. Additionally, the studies do not provide a comprehensive summary of the evaluated KE tools. The main motivation of this study is to empirically evaluate the performance of unsupervised KE tools that are open-sourced and ready to use, and explicitly understand the underlying method of such tools. A properly implemented KE tool can increase the productivity, accuracy, and scalability of developers working with text data as well as the accuracy of downstream applications that depend on keyword extraction as a preprocessing step. In this study, the performance of several unsupervised KE tools implemented in the Python programming language is evaluated. Evaluating all KE methods previously proposed in the literature is out of the scope of this study. The study does not encompass a performance analysis of widely used commercial APIs, such as OpenAI GPT-3 and Amazon Comprehend, because they are not freely available to use, and their internal working principles are not known.

The presented research aims to identify whether there is an optimal KE tool with fast response time and high accuracy for different applications. Different KE tools are implemented in different programming languages. The experimental analysis presented in this paper, however, considers only the performance of open-source unsupervised KE tools implemented in the Python programming language. The performance comparison of diverse KE tools can also be a good research direction.

To summarize, even though there are numerous methods for keyword extraction, effectively extracting keywords from different data sources is a promising research direction. Section IV presents a summary of the survey studies on the KE methods. As mentioned in advance, this is a comparative experimental performance analysis of open-source unsupervised KE tools implemented in the Python programming language.

III. RESEARCH METHODOLOGY

Despite the substantial number of proposed KE methods, only a few of them have been developed as open-source software applications or tools that can be easily integrated into other applications to automate the process of extracting important keywords. The objective of this study is to evaluate open-source unsupervised KE tools across a range of data types and analyze the key factors that can impact their performance. This study will help the readers in developing a strategy or selecting the right KE tool for the application for which they are aiming. Before evaluating the tools, it is necessary to go through the state-of-the-art literature for a better understanding of the underlying techniques, challenges in the extraction process, and benchmark datasets. It is also essential to review the literature that evaluates the KE methods for finding the scope of improvement. The relevant research articles are retrieved by querying the popular online computing research repositories including Science Direct, IEEE Xplore, Google Scholar, and ACM Digital Library. The basic set of keywords for the query includes: 'Keyword Extraction', 'Key phrases Extraction', 'Automated Keyword Extraction Survey', and 'Evaluating Keyword Extraction Methods'. Query keywords are updated when a new set of seed keywords is found relevant to research needs. Research articles that focused on other genres other than text are screened out. Further, the research articles are filtered to select the survey studies as surveying the survey would help to see a broader perspective and the literature evaluating different KE methods to have a clear understanding of evaluation metrics, benchmark datasets, and scope of improvement. Initial query after screening in only text genre resulted in more than two hundred research articles including journal, conference paper, abstracts, and thesis. Then the filter is applied to select full-text published articles focusing on the most relevant, cited, and recent work. Applying these criteria, a total of thirty-eight research articles are selected that include KE survey, evaluating KE methods, and proposed methods with a developed software application or tool. The rest of the paper analyzes these research articles along with the evaluation of open-source unsupervised KE tools and key factors that can affect the performance of those tools.

IV. BACKGROUND

This section introduces the existing survey studies on KE methods and summarizes their outcomes. As well as the state-of-the-art literature on the comparative evaluation of KE methods is analyzed and concisely presented in this section. This section also includes a brief discussion of key insights gathered by reviewing and summarizing the above-mentioned literature.

A. INTRODUCTION TO KEYWORD EXTRACTION

One of the most promising applications of Natural Language Processing (NLP) and Information Retrieval (IR) is the extraction of keywords from textual sources. Years of research and development have gone into it to extract practical and helpful insights from textual documents. Keyword extraction can be defined as finding the most pertinent words or the group of words that best capture a document's main idea [11], [12]. A process known as Automatic Keyword Extraction (AKE) involves sending any document or documents as input to a system that will automatically evaluate the data and give significant words or segments that are immediately applicable to other NLP problem domains including text summarizing [13], [14], topic detection [6], event detection [15], document indexing [16], profile matching [17], opinion mining [18], document classification [19], and so on. In a nutshell, keyword extraction can be used for business purposes such as automatic question-and-answer systems, spam identification, rumor detection, fake information detection, reputation analysis, sentiment analysis, and many more.

B. SURVEY STUDIES ON KE METHODS

In this section, a concise overview of survey studies focused on the KE task is presented. These studies often cover a wide range of topics, including methodologies, datasets, commonly used language and word patterns, and different evaluation measures. Existing survey studies on KE methods are reviewed to better understand the underlying techniques, several types of errors, challenges in keyword extraction, and benchmark datasets. Table 1 summarizes the survey studies on KE methods. It includes the major focus of these survey studies along with their additional coverage, inclusion of research methodology, benchmark dataset, toolkits, summary table, and comparison with previous studies.

Hasan and Ng presented a state-of-the-art exploration of keyword extraction which first provides a brief overview of data feature factors that affect the difficulty of keyword extraction [20]. Further, they classified the major KE approaches into three different approaches. Later they provided a list of the most significant errors produced by KE systems with a set of recommendations. The survey study of Siddiqi and Sharan focused on the classification of KE methods, and they classified the KE methods into four major classes that include statistical methods, supervised, semi-supervised and, un-supervised approaches [21]. They also discussed various feature selection measures utilized to rank potential keywords and key phrases according to the weight they carry in the text under analysis.

Beliga et al. proposed a survey study that primarily focuses on graph-based techniques for KE tasks [22]. Regarding vertices and edge representations, the study examined different graph types. The study covers commonly used centrality measures to find suitable vertices inside a graph using ranking of vertices in addition to providing a taxonomy for graph-based methods. Results analyzed on a data set of Croatian news articles reveal that selectivity tends to produce superior keywords while the other methods choose stop-words as the top-ranking keywords.

TABLE 1. Summary of literature survey on KE methods.

Study	Major focus/ Distinction	Additional coverage	Research methodology	Benchmark dataset	Toolkit	Feature selection	Summary table	Comparison with previous study
[20]	General	Error analysis, data related difficulties	NI	PI	NI	CI	PI	NI
[21]	General	Feature selection	NI	NI	NI	CI	NI	NI
[22]	Graph-based methods	Measures of graph analysis	NI	PI	NI	NI	CI	NI
[23]	General	Text summarizing	NI	NI	NI	NI	CI	NI
[24]	Neural network-based methods	Text summarizing, Detailed benchmark data-set	CI	CI	PI	NI	CI	CI
[25]	Deep learning-based methods	Different use cases, Strengths and Weaknesses	CI	PI	NI	PI	CI	NI
[28]	General	Key-ness property of keywords, key-ness features	NI	CI	NI	CI	PI	PI
[27]	Deep learning-based methods	Empirical evaluation	CI	PI	PI	CI	CI	NI
[29]	Deep learning-based methods	Deep learning methods and their fields of application	NI	PI	NI	NI	CI	NI

* CI denotes Comprehensive Information, PI denotes Partial Information, and NI denotes No Information.

Since the process of text summarization heavily depends on automatic keyword extraction, Bharti et al. examined recent research on both topics [23]. Nasar et al. extensively reviewed both keyword extraction and textual summarizing and classified the corresponding literature based on approaches covering the recent advancement in deep learning approaches [24]. Their study findings suggest that most of the work is conducted using unsupervised methods. Since deep learning architectures can use a wide range of neural designs and parameters, the use of different deep learning architectures for keyword extraction and the effects they have on the outcomes need to be further studied. Another unexplored issue in this field is how different variables affect overall outcomes as well as hyperparameter tuning.

Merrouni et al. classified the task of KE into four major steps and gave a brief explanation of supervised, unsupervised, and deep learning techniques in the context of commonly employed KE systems [25], [26]. The study also emphasizes the significance of feature selection because it often has a significant impact on how well the supervised techniques perform. Papagiannopoulou and Tsoumakas presented another extensive review of both unsupervised and supervised KE methods, including recent deep learning methods, highlighting the strengths and weaknesses of each method [27]. Another KE survey by Firoozeh et al. focused on discussing the complexity of the KE task and categorized the main approaches based on the features and methods [28]. Ajallouda et al. presented a review of deep learning-based KE methods and highlighted their contribution to improving KE performance [29].

The prior survey study on keyword extraction raises important implications. First, keyword extraction is becoming increasingly necessary to extract useful information from the massive textual data generated by online blogs, magazines, social networks, and other online platforms. However, the absence of annotated benchmark datasets for social platforms has hindered progress in this field. This could limit the potential development of more better and accurate KE method. Second, the survey studies show that applications of deep learning methods are relatively under-addressed for automatic keyword extraction, and the effect of different deep learning architectures remains an open research direction. Third, there is still a challenge in the semantic-aware evaluation of keyword extraction generated results. This is a challenging research question as it requires an understanding of the meaning and context of the text. Overall, the survey studies emphasize the continued research and development of KE methods with a focus on the application of deep learning and the development of benchmark datasets.

C. LITERATURE ON EVALUATING KE METHODS

The reliability of evaluation techniques and approaches and an examination of their flaws are still two of the largest problems for keyword extraction. This section reviews the state-of-the-art literature that evaluates the KE methods. There has been a lack of research on evaluating the KE methods that focuses on various aspects like hyperparameter tuning, execution time, data preprocessing, changes in the number of extracted keywords, different matching options for evaluation, and a distinctive summary of KE methods. Table 2 shows the major distinction of this study from prior literature on evaluating KE methods.

Using several evaluation techniques and metrics, Papagiannopoulou and Tsoumakas provided an empirical study comparing cutting-edge commercial KE APIs as well as popular unsupervised KE methods [27]. They thoroughly

TABLE 2. The major distinction of this study from prior state-of-the-art studies.

Study	Prior comparative evaluation	Benchmark dataset	Summarized KE	Hyper parameter tuning	Execution time	Data pre processing	`Top_n` keyword	Partial Vs Exact match
[27]	PI	CI	NI	NI	NI	PI	CI	CI
[7]	NI	CI	NI	NI	NI	NI	NI	NI
[8]	PI	CI	NI	NI	NI	NI	NI	NI
[30]	NI	CI	NI	PI	NI	NI	NI	NI
[9]	NI	CI	NI	NI	CI	NI	NI	NI
[31]	NI	CI	PI	NI	NI	NI	NI	NI
[17]	PI	CI	PI	NI	CI	NI	NI	NI
This Study	CI	CI	CI	CI	CI	CI	CI	CI

* CI denotes Comprehensive Information, PI denotes Partial Information, and NI denotes No Information.

analyzed the exact and partial matching approaches in their evaluation study, recommending the one that considers their average and emphasizing the need for methods that consider the semantic similarity of predicted and golden keywords.

Gallina et al. performed a methodical, extensive examination of contemporary KE models using benchmark datasets from various sources and fields [7]. Additionally, they offered fresh perspectives on the implications of adopting author- or reader-assigned keywords as a stand-in for the gold standard and made suggestions for solid starting points and trustworthy benchmark datasets.

Giarelis et al. performed a comparative assessment of five different KE methods and experimented with the KE methods with different scientific and news articles [8]. They examined whether the number of terms in the texts and the language of each dataset affect the accuracy of the chosen approaches using the f1-score and a partial match evaluation framework. Their experimental findings provided several insights into the performance of the chosen approaches in datasets of various languages as well as the applicability of the methods in texts of various sizes.

Garg tried to study different dimensions for graph-based KE methods and conducted a comprehensive survey study to demonstrate these different dimensions and make inferences from the existing literature [30]. The experiment was implemented to compare results over 21 datasets including well-formed and ill-formed datasets. The author concluded that the performance of the KE technique decreases as the size of the document increases and the word distribution gets more complex.

In the context of keyword extraction, Ushio et al. conducted a thorough and extensive empirical evaluation of both statistical and graph-based term weighting methods [9]. Their evaluation of fifteen different keyword extraction datasets produced a range of insights on the several types of methods. Regarding tf-idf, their work highlights the benefits of the lesser-known lexical specificity method. The authors also found that the statistical models are faster than the graph-based models.

Sun et al. examined how various datasets affect KE performance because the characteristics of the datasets directly influence how well the approaches for keyword extraction perform [31]. The authors recommended that researchers should compute the position of each word starting from the beginning of the paragraph rather than using the complete manuscript as a reference. Additionally, they recommended paying attention to the function and position of conjunctions during the keyword extraction process, as this might enhance efficiency.

There is an attempt to evaluate KE tools on different benchmark datasets by Nadim et al. to show that the integration of KE tools with different applications can significantly increase productivity, accuracy, and scalability [17]. They conducted their experiment on San Antonio Research Partnership Portal as a use case to develop different web applications that can use KE tools to increase productivity and scalability [32]. Other literature worked on evaluating KE methods for different data domains like painting archives [33], new multi-document benchmark datasets [34], similarity in news articles [35], and indexing news articles [16].

One of the main implications of the literature on evaluating KE methods is that there is no one-size-fits-all approach to choosing a KE method because each method has its strengths and weaknesses and because the effectiveness of a method might vary based on the domain and type of the text being analyzed. Therefore, careful evaluation is needed before selecting the right method for a particular task. Another implication is that there is no single most right evaluation metric to use. For supervised methods, precision, recall, and f-score are the most used metrics. For unsupervised methods, along with these metrics' other evaluation metrics such as coverage, diversity, and novelty can be employed. Additionally, to enable a fair comparison of different methods, annotated benchmark datasets are needed that represent various domains and types of text. Finally, the preceding evaluation literature emphasizes the need for future research in areas including advanced unsupervised methods to manage noisy and unstructured data, the application of deep learning techniques for KE tasks, and the development of semantic-aware evaluation metrics. Overall, prior study on evaluating KE methods provides valuable guidance and insights for researchers working in this field.

D. KEY INSIGHTS OF KEYWORD EXTRACTION

An intensive knowledge of KE has been gathered by reviewing and summarizing the above-mentioned literature.

This section will discuss the insights of this intensive knowledge on keyword extraction like evaluation metrics, the effect of supervised vs unsupervised methods, the impact of assigned keywords, execution time, data quality, simultaneous learning, and benchmark datasets.

1) EVALUATION METRICS

To figure out the most effective KE method, it is a legitimate aim to have evaluation metrics that offer a fair comparison of the KE methods. However, creating an evaluation metric that can capture the benefits and drawbacks of a method is a difficult job. A method can be evaluated more precisely and thoroughly using multiple metrics. Most works on KE methods used Precision, Recall, and F-score as metrics to evaluate the performance. The evaluation metrics are founded on the notion that keywords are independent of one another, but more important keywords ought to be placed higher. Other measures used to evaluate the performance of the KE method that reflect the ranking features between keywords are Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Binary Preference Measure.

a: PRECISION

Precision metric is used to evaluate the accuracy of a model's prediction and it is defined as the ratio of the true positive predictions to the total number of predictions made by the model.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Here, True Positive (TP) refers to the number of correct keywords extracted by the model and False Positive (FP) refers to the number of incorrect keywords extracted by the model. Precision is a useful metric to use in situations where the goal is to minimize false positives.

b: RECALL

Recall metric is used to evaluate the effectiveness of a model in identifying all relevant keywords and is defined as the ratio of true positive prediction to the total number of actual keywords in the data.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Here, False Negative (FN) refers to the number of keywords that the model failed to extract. Recall is a useful metric to use in situations where the goal is to minimize false negatives.

c: F-SCORE

The F-score is used to evaluate the overall performance of a model and it is the harmonic mean of precision and recall that balances the trade-off between them.

$$F - score = \left(1 + \beta^2\right) \frac{precision * recall}{\beta^2 * precision + recall} \tag{3}$$

When $\beta = 1$, it is called f1-score. F1-score is a good metric to use in situations where both precision and recall are important, and where we need to balance the trade-off between the two metrics to optimize overall performance. Unfortunately, the f-score has the drawback of not accounting for the ranks of the correct keywords.

d: MEAN AVERAGE PRECISION (MAP)

Mean Average Precision is another metric for evaluating a list of ranked keywords and is defined as follows:

$$MAP = \frac{1}{n}\sum_{i=1}^{n} AP_i = \frac{\sum_{n=1}^{|N|} P(n)\, gd(n)}{|LN|} \tag{4}$$

where AP_i is the average precision of the extracted keywords list. $|N|$ denotes the length of the list, $|LN|$ is the number of relevant keywords, $P(n)$ is the precision, and $gd(n)$ equals 1 if the nth keyword is a gold keyword and 0 otherwise. We get the MAP by averaging the AP over a set of n documents. Though MAP is not a commonly used metric for KE model evaluation, Jiang et al. leverages MAP to evaluate the KE models [36].

e: MEAN RECIPROCAL RANK (MRR)

In the list of keywords that have been extracted using a KE method, Mean Reciprocal Rank assesses how a document's first accurate keyword ranks. MRR is defined as follows:

$$MRR = \frac{1}{|D|}\sum_{d\, \in D} \frac{1}{rank_d} \tag{5}$$

where D is the set of documents and $rank_d$ is the rank of the first accurate keyword with all extracted keywords in a document d. Liu et al. used MRR to evaluate the KE method, though it is not very common [6].

f: BINARY PREFERENCE MEASURE (BPERF)

It is desirable to use the Binary Preference Measure to assess the performance of KE methods while taking the ranking of the extracted keywords into account. Bperf assesses the number of bad keywords that appear higher in the ranking than the good ones [37] and defined as follows:

$$Bperf = \frac{1}{R}\sum_{r\, \in\, R} 1 - \frac{|n \text{ ranked higher than } r|}{M} \tag{6}$$

where R is the number of correct keywords within M extracted keywords by a method, in which n is an incorrect keyword and r is a correct keyword.

2) SUPERVISED VS. UNSUPERVISED METHODS

The supervised method typically outperforms the unsupervised method [5]. However, large volumes of labeled data must be made readily available for supervised KE methods to function well [6]. The amount of data needed to reach optimal performance is still up for debate. Gallina et al. showed that as more training data is supplied, the model's performance slowly increases, indicating that the size of the existing datasets may not be sufficient [7]. Although supervised KE methods outperform their unsupervised counterparts in f1-score, they are unable to manage huge document collections without predetermined keywords, mostly because of the enormous amount of manual labor required by human annotators.

3) EXECUTION TIME

The execution time of KE methods on a given dataset could be an important performance parameter and can have several effects. Kumar et al. used time as an efficacy metric to conduct an empirical analysis of three different KE methods for text extraction and summarizing [38]. The method's ability for scaling can be affected by the time needed for keyword extraction. Its applicability is constrained if the method's execution time makes it impractical to use it on big datasets. The execution time becomes a crucial element when keyword extraction is done in real time. In these circumstances, the method must be created to deliver quick results without sacrificing the effectiveness of the keywords. The quantity of memory and processing power that KE methods use can be quite high. High execution times can influence the system's total resource usage. The execution time may influence the user experience if the keyword extraction process is part of a user-facing application. Frustration and disengagement can result from slow reaction times. Designing methods that can deliver quick and precise results while using the least number of computational resources is therefore crucial.

4) BENCHMARK DATASET

Benchmark datasets are vital for keyword extraction because they offer a consistent framework for assessing and contrasting various KE methods. Researchers can compare the results of various methods on the same collection of documents and keywords by using the same dataset, allowing them to figure out which methods are most successful. Additionally, benchmark datasets offer a means to confirm that KE methods can be applied to a variety of text genres and domains. A method is more likely to be successful on other datasets if it works well on a benchmark dataset. Benchmark datasets can also be used to pinpoint areas where KE methods need to be improved. For instance, if a specific dataset presents difficulties for current methods, researchers can concentrate on creating new methods that manage these difficulties. Benchmark datasets offer a means to advance state-of-the-art keyword extraction and raise the standard of automated KE methods. Table 3 shows the widely used and publicly available benchmark datasets for evaluating KE methods. The benchmark datasets are mostly English language-dependent, and the annotator of these benchmark datasets can be grouped into three classes: Author, Reader, and Professional Indexer. There is a wide range of mean words per document and mean keywords per document for these benchmark datasets. Fig. 1 shows a percentage comparison of mean words per document and mean keywords per document for all benchmark datasets. The short-text datasets have a higher ratio of mean keywords

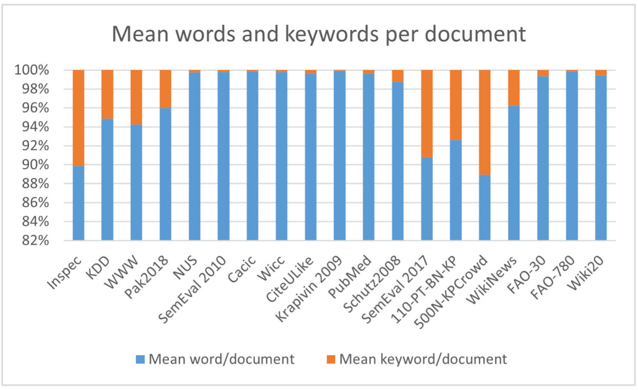


FIGURE 1. Mean words and keywords per document for all benchmark datasets.

per document. Author-annotated datasets tend to have a lower number of mean keywords per document.

5) IMPACT OF ASSIGNED KEYWORDS

The use of gold keywords can influence keyword extraction in positive as well as negative ways. On the one hand, offering a collection of gold keywords can aid in assessing the effectiveness of a KE method and comparing it with different methods. Additionally, it can be used to optimize the method's parameters for better efficiency. On the other hand, using gold keywords as the method's input can skew the results in favor of these terms, possibly causing the method to overlook other important words that weren't in the gold collection. Author-assigned keywords are usually used as the gold standard in datasets that include scientific articles or abstracts. Keywords assigned by readers in datasets appear to create gold-standard annotations through extractive means. There might be more false negatives during evaluation because both annotations may use distinct keywords to describe the same documents. For datasets like Inspec, the gold standard given by expert indexers produces higher performance scores during evaluation [17].

6) DATA QUALITY

When it comes to evaluating keyword extraction methods, data quality is a critical factor that cannot be overlooked. The quality of the data used for keyword extraction can greatly influence the accuracy and relevance of the results. High-quality data that is properly labeled, categorized, and annotated can help KE methods to better understand the context and meaning of the text. On the other hand, low-quality data that is poorly labeled or has errors, noise, or bias can lead to inaccurate and irrelevant keyword extraction results. Therefore, it is essential to carefully select and prepare the data for evaluation to ensure that the KE method is evaluated under realistic and reliable conditions. Additionally, it is important to continuously check and improve the data quality during the evaluation process to prevent any biases or errors from affecting the results.

TABLE 3. Summary of benchmark datasets used for evaluating KE methods.

Category	Title	Annotator	Language	Number of documents	Mean words per document	Mean keywords per document	Study	Description
Paper abstract	Inspec	PI	English	2000	124	14	[42]	Abstracts of scientific journal papers from Inspec database.
	KDD	A	English	755	74	4	[43]	Scientific paper abstract from ACM conference on Knowledge Discovery and Data Mining
	WWW	A	English	1330	82	5	[43]	Scientific paper abstract from World Wide Web Conference
	Pak 2018	A	Polish	50	96	4	[44]	Paper abstract collected from Measurement, Automation and Monitoring journal.
Full text paper	NUS	A+R	English	209	5122	12	[45]	Full text scientific papers collected using Google SOAP API.
	SemEval 2010	A+R	English	243	8032	16	[46]	Conference and workshop papers from ACM digital library.
	Cacic	A	Spanish	888	3893	4	[47]	Scientific papers published in Argentine Congress of Computer Science.
	Wicc	A	Spanish	1640	1917	4	[47]	Scientific papers published in Workshop of Researchers in Computer Science.
	CiteULike	R	English	183	4597	17	[48]	Scientific papers from CiteULike.org with pre-set filter.
	Krapivin 2009	A+R	English	2304	7856	5	[49]	Full text scientific papers from ACM database.
	PubMed	PI	English	500	3869	14	[50]	Full text biomedical literature scientific papers from PubMed central.
	Schutz 2008	A	English	1231	3551	45	[51]	Full text biomedical literature scientific papers from PubMed central distributed across 254 different journals.
Paper para-graph	SemEval 2017	R+PI	English	493	168	17	[52]	Paragraph of scientific publications from ScienceDirect.
News article	110-PT-BN-KP	R	Portuguese	110	301	24	[53]	News articles from European Portuguese ALERT BN database.
	500N-KPCrowd	R	English	500	394	49	[54]	Broadcast news stories on 10 different categories.
	WikiNews	R	French	100	282	11	[55]	News articles from French version of WikiNews.
Full-text article	FAO-30	PI	English	30	4793	32	[56]	Documents from United Nations Food and Agriculture Organization.
	FAO-780	PI	English	779	4863	8	[56]	Documents from United Nations Food and Agriculture Organization.
	Wiki20	R	English	20	6018	35	[57]	Technical research report covering different aspect of computer science.

A = Author, R = Reader, PI = Professional Indexer

7) DATA TYPE

The length of the input document increases the difficulties of the KE process because longer documents produce more candidate keywords and a larger search space [39]. Therefore, compared to abstracts, emails, and news articles, it is more difficult to extract keywords from scientific papers, technical reports, and meeting transcripts. The structural consistency of a document is also likely to ease keyword extraction because of the standard format as there are certain locations where a keyword is most likely to appear. The presence of uncorrelated topics in documents may also increase the difficulties of keyword extraction.

8) SIMULTANEOUS LEARNING

Simultaneous learning is a technique used in keyword extraction to improve the quality of extracted keywords. This technique involves jointly learning the representation of the document and selecting the most relevant keywords. Researchers hypothesized that text summarization and keyword extraction may perhaps benefit from each other if both processes were carried out simultaneously since keywords represent a dense summary of a document [40], [41].

V. KEYWORD EXTRACTION TOOLS

This section provides a brief discussion about the selected open-source unsupervised KE tools. A wide range of KE tools including Statistical-based, Graph-based, and Deep Learning-based are selected for this study. Table 4 summarizes the selected KE tools including their features, drawbacks, and summary. It also includes information about the language dependencies like the use of stop words, part of speech (PoS) tags, and word stemming.

A. STATISTICAL-BASED TOOLS

The method of statistical-based keyword extraction finds the most important words or phrases in a text by performing statistical analysis on it. To assess the significance of each word or phrase in the text, the approach includes computing various statistical measures, including term frequency, inverse document frequency, and others. These statistical measurements are used to rank the words or phrases in the document according to how relevant they are to the general subject matter, with higher-ranked terms considered as keywords. This approach is effective in identifying statistically significant words or phrases within the text.

1) TF-IDF

Term Frequency-Inverse Document Frequency (Tf-Idf) is a popular numerical statistical method used in natural language processing for extracting keywords from a text document [58]. This method works by assigning a weight to each term in the document based on its frequency in the document (TF) and its inverse frequency across all documents in the corpus (IDF).

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

$$IDF(\omega) = \log(\frac{N}{df_i})$$

$$TF - IDF = TF_{i,j} * \log(\frac{N}{df_i}) \tag{7}$$

where N denotes the total number of documents, $TF_{i,j}$ is the number of occurrences of word i in document j, and df_i denotes the number of documents containing word i. The intuition behind this is that terms that appear frequently in a document but rarely in the corpus are more likely to be important keywords that represent the main themes or topics discussed in the document.

2) KPMINER

El-Beltagy and Rafea proposed an unsupervised KE method named KPMiner that uses a modified version of Tf-Idf and works with n-gram [59]. The proposed method is divided into three major steps including a selection of candidate keywords, weight calculation of candidate keywords, and refining the selected keywords. After removing punctuation and stop words in the candidate keywords selection stage, two new statistical features were added by KPMiner. i) For a word to be a candidate keyword, it must occur at least n times in the document from which keywords are to be extracted and this is called the least allowable seen frequency ($lasf$) factor. ii) A word will not be considered a keyword and will be filtered out if it occurs in a long document after a specific cutoff threshold position. In the candidate keywords weight calculation stage, KPMiner uses a boosting factor for compound keywords to balance the bias of Tf-Idf towards single keywords as single keywords tend to achieve higher scores because of potential multiple presence. In the final stage, keywords are refined to return top n keywords.

3) YAKE

Yet Another Keyword Extractor (YAKE) is another well-known unsupervised KE method that uses statistical features from a single document without depending on any corpus to extract the most important keywords [44]. The significant difference between KPMiner and YAKE is that it introduces a new diverse feature set having five features to capture the characteristics of each word [60]. i) Term casing (T_{Case}) reflects the case sensitivity of candidate words. ii) Term position ($T_{Position}$) reflects that the words that appear in the early sentences of a text have higher values than the words that appear later. iii) Term frequency normalization (T_{FNorm}) represents that a candidate word's significance increases with the candidate word frequency. However, it requires normalization to prevent the bias towards high frequency in long documents. iv) Term relatedness to context (T_{Rel}) shows how many distinct words are present on either side of a candidate word. v) Term different sentence ($T_{Sentence}$) represents the idea that candidate words that appear in a variety of sentences are more likely to be significant. After calculating all feature scores, the unique word score is calculated using the following formula:

$$S(t) = \frac{T_{Rel} * T_{Position}}{T_{Case} + \frac{TF_{Norm}}{T_{Rel}} + \frac{T_{Sentence}}{T_{Rel}}} \tag{8}$$

The final word score of each candidate word is calculated using the $n - gram$ model as follows:

$$S(kw) = \frac{\prod_{t \in kw} S(t)}{KF(kw) * (1 + \sum_{t \in kw} S(t))} \tag{9}$$

where kw represents a candidate keyword of $n-gram$ and KF is the frequency of the candidate keyword. The smaller the score of a candidate keyword, the more relevant the candidate keyword will be.

4) RAKE

Rapid Automatic Keyword Extraction (RAKE) is also a domain-independent and language-independent unsupervised KE method that can extract keywords from individual documents [61]. RAKE first selects the candidate keywords by using a stop word list and specified word delimiters. These candidate keywords are then ranked based on their frequency and co-occurrence with other keywords in the text, using a statistical measure called degree centrality. Additionally, RAKE searches for groups of stop word-containing keywords that appear next to one another in the same sequence and at least twice in the same document. Then the top one-third scoring candidate words are selected as keywords for the document.

B. GRAPH-BASED TOOLS

Graph-based keyword extraction idea originated from Google's PageRank algorithm [62] with the basic assumption that more edge connections of a graph showed more significant candidate words. A general approach for graph-based keyword extraction begins with text preprocessing, where the text is cleaned, tokenized, and divided into words or phrases. A co-occurrence graph is then constructed, with nodes representing words or phrases and edges denoting co-occurrence relationships. Nodes in the graph are scored to identify their importance and the top-ranked nodes are considered as keywords. Graph-based methods capture the semantic relationships between words in the document, so they can provide a more comprehensive and contextualized understanding of the content of a document.

1) TEXTRANK

TextRank is the first method to rank the relevance of sentences or keywords in a text document using the graph-based algorithm PageRank [63]. The method starts by creating a graph of the text document. Each sentence or keyword is represented as a node in the graph, and the edges between the nodes show how similar the associated sentences or keywords are to one another. To prevent the excessive growth of the graph, it only considers 1-gram for the graph and eventually reconstructs the multi-word keyword in the post-processing phase. Unlike PageRank, it considers a weight between two nodes as the graphs are built from a natural language context. It defines a formula that integrates the weights to calculate the weight score (WS) of the node V_i as follows:

$$WS\left(V_i\right)=(1-d)+d*\sum_{V_j\,\in\, In(V_i)}\frac{w_{ji}}{\sum_{V_k\,\in\, Out(V_j)} w_{jk}}WS(V_j) \tag{10}$$

where d is a dumping factor, w is the weight, $In(V_i)$ is the set of nodes that point to node V_i, and $Out(V_i)$ is the set of nodes that node V_i points to. It is a powerful and efficient method for identifying important information in large text documents.

2) SINGLERANK

Wan and Xiao proposed a graph-based method named SingleRank which is essentially like TextRank with some major differences [64]. The method works by constructing a bipartite graph where one set of nodes represents the documents, and the other set represents the words in the documents. The edges between the nodes represent the co-occurrence of words in the documents. SingleRank then uses an iterative process to calculate the importance scores for each document based on the scores of the words that appear in the document and the scores of the documents that have those words.

3) POSITIONRANK

PositionRank is a graph-based KE method that introduces location information with the idea that the important keywords appear earlier in the document [65]. The method employs a modified PageRank approach, where nodes in the graph indicate keywords, and edges represent the similarity between pairs of keywords based on their position and co-occurrence. It assigns a weight based on the position of words and calculates the score of nodes as follows:

$$\widetilde{p}=\left[\frac{p_1}{p_1+p_2+\ldots+p_{|V|}},\right.$$
$$\left.\ldots,\frac{p_{|V|}}{p_1+p_2+\ldots+p_{|V|}}\right] \tag{11}$$

$$S\left(v_i\right)=(1-d)\ .\,\widetilde{p_i}$$
$$+d*\sum_{V_j\,\in\, Adj(V_i)}\frac{w_{ji}}{\sum_{V_k\,\in\, Adj(V_j)} w_{jk}}S(V_j) \tag{12}$$

where p is a vector of length $|V|$ and it indicates, being in a node v_i, the random walk can jump to any other node in the graph with equal probability.

4) TOPICRANK

TopicRank is a graph-based KE method that uses the topical representation of the document to extract important keywords from the document [55]. The method first identifies candidate topics by clustering related keywords in the document, using a variant of the TextRank method to identify the most important keywords in each cluster. It constructs a graph where nodes represent the identified topics and edges represent the degree of similarity between topics. The similarity between topics is calculated based on the similarity between the keywords that make up each topic. Hierarchical Agglomerative Clustering (HAC) algorithm [66] is used to automatically group similar keywords into topics. TopicRank is particularly useful for identifying important topics that may not be captured by individual keywords.

5) TOPICAL PAGERANK

Topical PageRank is a graph-based keyword extraction method that combines the graph-based approach of PageRank with topical information from an external corpus [67]. The method first constructs a graph of the document, where nodes represent words and edges represent co-occurrence relationships between words. Next, Topical PageRank uses Latent Dirichlet Allocation (LDA) [68] for the topical information from an external corpus to assign each word to one or more topic categories. The method then applies a variant of the PageRank algorithm that considers both the importance of each word within the document and its relevance to each of the topic categories. For topic z, Topical

TABLE 4. **Summary of selected open-source unsupervised KE tools.**

Tool	Study	Method	Stop word	PoS	Stem	Feature	Drawbacks	Summary
Tf-Idf	[58]	Statistical based	✓	X	X	Word frequency	Have a bias toward single keywords as they can appear in both single and compound keywords.	Words that appear frequently in a document but rarely in the corpus are more likely to be important keywords in that document.
KPMiner	[59]	Statistical based	✓	X	✓	Word frequency with condition and boosting factor.	Cutoff threshold is only useful in certain types of documents.	Keywords must appear at least n times and not after a cutoff threshold position in a long document. Boosting factor for compound keywords to balance the bias of TF-IDF toward single keyword.
YAKE	[60]	Statistical based	✓	X	X	Sentence structure, word frequency, and co-occurrence.	Sensitive to the length and complexity of the input text.	After text preprocessing, diverse feature set is used to calculate unique word score. Final word score is calculated using n-gram model.
RAKE	[61]	Statistical based	✓	X	X	Word frequency and co-occurrence.	If the stop word list is not exhaustive, it would consider continuous long text as a keyword. N-gram containing stop word could be missed.	After removing stop words, candidate keywords are ranked based on their frequency and co-occurrence with other keywords in the text using degree centrality. Top one third scoring candidate words are selected as keywords.
SpaCy	[72]	Deep learning based	✓	✓	*	Word frequency and position.	May not be suitable for specialized domains.	Deep learning model trained on large, annotated dataset that uses combination of linguistic rules and statistical models.
TextRank	[63]	Graph based	X	✓	X	Graph of word nodes and co-occurrence between them.	May not be well-suited for short texts. Does not consider the context or meaning of the text.	Each word represents a node in the graph and the edges between the nodes indicate the similarity between the corresponding words.
SingleRank	[64]	Graph based	X	✓	X	Bipartite graph with documents and words in the document.	Has difficulties to handle noisy or sparse data.	One set of nodes represents the documents, and the other set represents the words in the documents. The edges between the nodes represent the co-occurrence of words in the documents.
TopicRank	[55]	Graph based	X	✓	✓	Graph of topics and similarities between them.	May not perform well to very short text with very few meaningful topics.	Words are clustered into topics and each topic represents a node in the graph. The edge represents the similarity between topics.
Topical PageRank	[67]	Graph-based	✓	✓	✓	Graph of word nodes and co-occurrence between them with a probability score based on topic.	Requires external corpus.	Each word represents a node in the graph and the edges between the nodes indicate the similarity between the corresponding words. Using LDA, each word is assigned to one or more topics. The relevant word has larger probability score.
Position Rank	[65]	Graph-based	✓	✓	✓	Graph of word nodes. Position and co-occurrence between them.	Requires a large amount of text data to be effective.	Each word represents a node in the graph and the edges between the nodes indicate the similarity based on position and co-occurrence between the corresponding words.
Multipartite Rank	[69]	Graph-based	X	✓	✓	Graph of topics and similarities between them along with word position.	Complex computation. May not be effective for individual document.	Words are clustered into topics and each topic represents a node in the graph. The edge represents the similarity between topics. Edge weights are adjusted to capture position information.
KeyBERT	[73]	Deep-learning-based	✓	✓	✓	Cosine similarity between embedding vector of word and the document.	Requires pre-trained models. Performance depends on the quality of the pre-trained model.	Words which have a vector representation highly like the one of the documents, are the keywords representing the document.

∗ Instead of stemming SpaCy uses lemmatization, which is the process of reducing words to their base form by applying morphological analysis of the word.

PageRank calculates the score R as follows:

$$R_z(w_i) = (1-d)\ .\ p_z(w_i) + d * \sum_{j:w_j \to w_i} \frac{w_{ji}}{\sum_{j:w_i \to w_j} w_{jk}} R_z(w_j) \qquad (13)$$

where $p_z(w)$ is topic specific preference value for a specific topic z. The assigned probability will be larger for relevant words.

6) MULTIPARTITERANK

MultipartiteRank is a graph-based keyword extraction method that constructs a multipartite graph, where nodes represent both individual documents and phrases that appear in the documents [69]. MultipartiteRank applies a modified version of the PageRank algorithm that considers the bipartite structure of the graph, as well as the importance of each phrase within its respective document. The result is a set of keywords that are representative of the content of the entire document collection, rather than individual documents. This method is more complicated because it includes a step where edge weights are modified to consider positional information. As a result, there is a bias in favor of potential keywords that first appear in the text.

C. DEEP LEARNING-BASED TOOLS

Recent developments in deep learning have made it possible for academics to enhance traditional KE methods, which only use graph and statistical measures, by using word embedding to better capture the semantic relationships between words in the text. The typical process for deep learning-based keyword extraction begins with data preparation and the utilization of pre-trained deep learning models, such as word embeddings or advanced contextual models like BERT and GPT. These models are employed to derive features from the text, effectively capturing both semantic and contextual information by leveraging the power of deep learning. While there has been a substantial amount of research on keyword extraction using deep learning models [70], [71], our focus remains on open-source unsupervised KE tools that are ready to integrate into various applications.

1) SPACY

Spacy is a popular Python library for natural language processing, which uses deep learning techniques to build statistical models for performing a wide range of tasks. Spacy's models are trained on substantial amounts of annotated text data and designed to be both accurate and efficient, so they can be applied to large-scale text data in real-time [72]. Spacy's keyword extraction method then uses a simple heuristic to score each candidate keyword based on its frequency in the text and its position in the sentence. One advantage of using Spacy for keyword extraction is that it is fast and easy to use. However, since the method relies on simple heuristics, it may not be as accurate or effective as more sophisticated methods. Additionally, Spacy's keyword extraction may not be suitable for all types of text, such as highly technical or specialized domains.

2) KEYBERT

KeyBERT, a state-of-the-art keyword extraction library developed by Maarten Grootendorst, uses pre-trained word embeddings models to find the most important words or phrases in a document [73]. First, using Scikit-learn's CountVectorizer class, the model generates a collection of potential keywords for each document. This class provides a straightforward bag-of-words implementation that counts how often each keyword appears. Second, a document embedding vector based on the text's words is created, as well as an embedding vector for each potential keyword. The sentence-transformer package is used to create these embeddings. KeyBERT offers the choice to use different embedding models, including BERT, RoBERTa, and DistilBERT. Thirdly, following the creation of the necessary embedding vectors, a pairwise cosine similarity score is computed between each potential keyword and the document's embedding vector. After that, the keywords are ranked according to how close they are, using the similarity score. KeyBERT can include an extra diversification step by using the Maximal Marginal Relevance (MMR) or Max Sum Distance (MaxSum) measure.

a: MAXIMUM MARGINAL RELEVANCE (MMR)

MMR considers the similarity of keywords with the document, along with the similarity of already selected keywords to address the flaws of highly identical outcomes. As a result, the keywords chosen are as diverse as possible within the context of the document.

b: MAX SUM DISTANCE (MAXSUM)

It takes twice the number of `top_n` most similar words to the document. The combination of words that are the least similar to each other by cosine similarity is then extracted from all of the `top_n` combinations of twice the number of `top_n` words. For a large number of `top_n` words, it is not advisable to use as it requires more computational time.

The statistical-based tools usually use word frequency and co-occurrence to find the keywords from a text. Short text data might not have enough statistical data to distinguish between keywords and non-keywords, which might have affected the performance of the statistical-based tools. In graph-based tools, either word or topic is used to construct a graph first, and then co-occurrence or similarities are used to extract the keywords. The position of a word can also be considered in graph-based tools. The abundance of information in long text data might hamper the performance of the graph-based tools. Deep learning-based tools can use the embedding vector of word and cosine similarity to find keywords that represent the document. A concern of deep learning-based tools could be that the performance depends on the quality of the pre-trained model. The inclusion of several types of tools

TABLE 5. Parameter configuration of selected unsupervised KE tools.

Tool	Parameter
Tf-Idf	language= 'en', grammar= 'NP: {<ADJ>*<NOUN|PROPN>+}'
KPMiner	language= 'en', grammar= 'NP: {<ADJ>*<NOUN|PROPN>+}'
YAKE	lan= 'en', n=3, windowsSize=3
RAKE	-
SpaCy	en_core_web_md, merge_noun_chunks, not is_stop and not is_punct and not like_num
TextRank	en_core_web_md
SingleRank	pos={'NOUN', 'PROPN', 'ADJ', 'ADV'}, language= 'en', window=3
TopicRank	language= 'en', pos={'NOUN', 'PROPN', 'ADJ', 'ADV'}
Topical PageRank	pos={'NOUN', 'PROPN', 'ADJ'}, language= 'en', grammar= 'NP: {<ADJ>*<NOUN|PROPN>+}'
Position Rank	pos={'NOUN', 'PROPN', 'ADJ', 'ADV'}, language= 'en', maximum_word_number=5, window=3
Multipartite Rank	pos= {'NOUN', 'PROPN', 'ADJ'}, language= 'en'
KeyBERT	keyphrase_ngram_range= (1,3), stop_words= 'english'
KeyBERT (mmr)	keyphrase_ngram_range= (1,3), stop_words= 'english', use_mmr=True
KeyBERT (maxsum)	keyphrase_ngram_range= (1,3), stop_words= 'english', use_maxsum=True

and methods for comparative analysis provides insight into selecting appropriate application-specific KE tools.

VI. EXPERIMENT

In this experiment, the commonly used unsupervised KE tools that have a Python implementation are evaluated using several benchmark datasets. Python programming language is selected because most of the unsupervised KE tools have an implementation in Python programming language and Python programming language is now widely used to develop AI applications. This experiment aims to find the best KE tool with the fastest response time and higher accuracy. So that the KE tool can be integrated with different applications.

A. EXPERIMENTAL SETUP

An Intel Core i7 2.9 GHz Quad-Core processor with 16GB 2133 MHz LPDDR3 memory on a MacOS machine is used for this experiment. All KE tools are implemented for Python programming language using Python program manager, pip. All KE tools are configured to generate n-grams with sizes ranging from 1 to 3. Table 5 shows the parameter configuration of selected unsupervised KE tools. The top 10 keywords are extracted from all KE tools. The extracted keywords are compared with golden keywords for partial matching using a Fuzzy matching algorithm where the partial matching threshold is set to 80%.

B. BENCHMARK DATASETS

The distribution of the number of words in a text in a dataset can provide important insights into the characteristics of the dataset and can impact the analysis and modeling of the data. The length of the text can affect the results of these

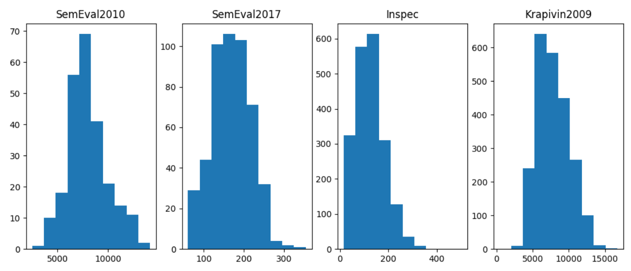


FIGURE 2. Histogram of the number of words for SemEval2010, SemEval2017, Inspec, Krapivin2009.

preprocessing steps, so understanding the distribution of the number of words in the text can help to choose appropriate preprocessing techniques. Also, understanding the distribution of the number of words in the text can help to choose an appropriate method and set appropriate parameters. Fig. 2 and 3 show the number of word distributions for different benchmark datasets. Four benchmark datasets have been selected for this experiment that has a good distribution of the number of words in a text. The selected benchmark datasets are SemEval2010, SemEval2017, Inspec, and Krapivin2009.

The Inspec dataset is the collection of short-text scientific articles in the field of computer science, and it has been widely used in the literature due to its rich domain-specific vocabulary and well-defined gold standard keywords [24]. The SemEval2010 contains a collection of scientific articles from multiple domains and is often used for general-domain documents. The SemEval2017 dataset was recently introduced and is an extension of the SemEval2010 dataset, which includes a larger set of documents covering a wider range of domains. This dataset is especially useful in evaluating KE methods that can work across multiple domains [28]. The Krapivin2009 dataset contains long-text scientific articles from the computer science domain [49]. Thus, the selection of these benchmark datasets for evaluating KE methods will cover a wide range of domains and both short and long-text documents. One setback of this selection of benchmark datasets might be language dependence. All the selected datasets are English language-based. Creating a high-quality language-independent benchmark dataset is still an open research challenge. Also, there is a lack of widely used well-annotated benchmark datasets of different emerging domains including social media text, web articles, and news articles. Section VI-C9 briefly discusses the available benchmark dataset repositories. For this comparative analysis, benchmark datasets are collected from the repository described in [60].

C. RESULTS AND DISCUSSION

In this section, the result of evaluating selected unsupervised KE tools is discussed. In this experiment the execution time, number of matched keywords, precision, recall, and f-score are calculated to evaluate the KE tools. Table 6 shows the result summary of this experiment. Furthermore,

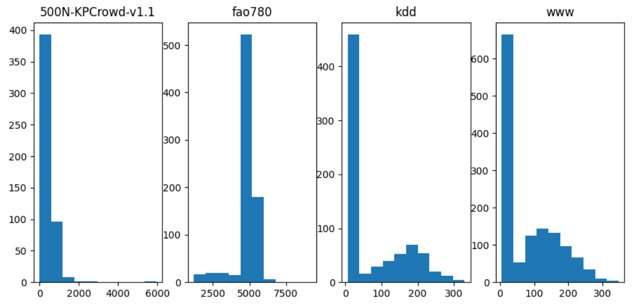


FIGURE 3. Histogram of the number of words for 500N-KPCrowd, Fao780, KDD, WWW.

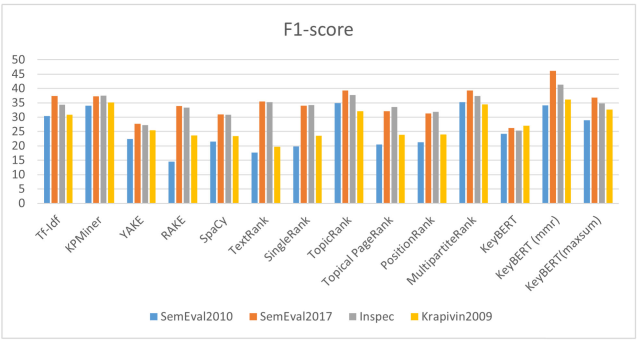


FIGURE 4. F1-score of all selected KE tools on four benchmark datasets.

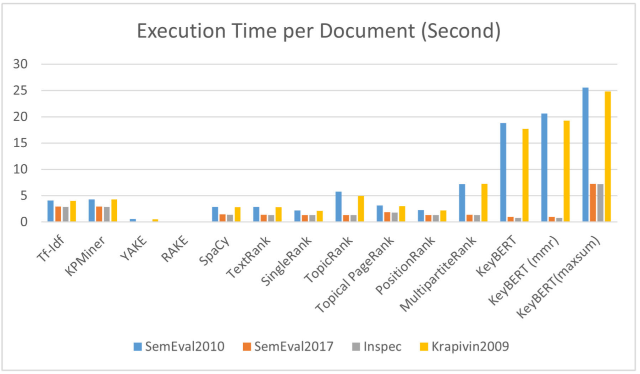


FIGURE 5. Execution time per document in seconds of all selected KE tools on four benchmark datasets.

the key factors that can impact the performance of KE tools like hyperparameter tuning, number of extracted keywords, exact Vs. partial matching, and data preprocessing are experimented with. In addition, with other metrics diversity metric is used to measure the performance of selected KE tools.

1) DATA TYPE

The type of data used for evaluating KE tools has a profound impact on evaluation results. Distinct types of data may require different approaches and techniques for effective keyword extraction, and some tools may perform better on certain types of data than others. Since full-text scientific publications contain enough statistical data to distinguish between keywords and non-keywords, the results of f1-scores on datasets from those full-text articles should support the superiority of statistical methods over graph-based ones. The detection of keywords using graph-based approaches might be further hampered by the abundance of information. Furthermore, when it comes to extracting keywords from short scientific papers, graph-based methods should perform better than statistical methods. Fig. 4 shows the f1-score of all selected KE tools on four benchmark datasets.

From Table 6, we can see that for all selected KE tools, the SemEval2017 dataset has the highest precision score and the Krapivin2009 dataset has the highest recall score. This is because the SemEval2017 dataset has the highest mean number of keywords per document and Krapivin2009 has the lowest mean keywords per document. To optimize the overall performance of KE tools it is better to investigate the f-score that balances the trade-off between the precision and recall.

Fig. 4 shows that the most consistent KE tools over four benchmark datasets in terms of percentage change in f1-score are: KPMiner (9.18%), KeyBERT (10.54%), and MultipartiteRank (12.37%). It also shows that the top KE tools in terms of average f1-score are: KeyBERT(mmr) (39.43%), MultipartiteRank (36.57%), TopicRank (36.0%), and KPMiner (35.96%). It is also clear from Table 6 and Fig. 4 that KeyBERT(mmr) outperforms all KE tools on three benchmark datasets: SemEval2017, Inspec, and Krapivin2009. Except in SemEval2010 dataset both MultipartiteRank and TopicRank perform slightly better than KeyBERT(mmr). Additionally, short-text datasets SemEval2017 and Inspec annotated by professional indexer show better results on all KE tools than long-text datasets SemEval2010 and Krapivin2009 annotated by author and reader.

The outcome demonstrates that statistical-based tools, such as Tf-Idf and KPMiner, outperform graph-based tools, such as TextRank, PositionRank, and SingleRank. Additionally, graph-based tools find it challenging to extract keywords from large text databases. Surprisingly, topic-based KE tools like TopicRank and MultipartiteRank, where topics are viewed as nodes in a graph, do well on datasets with both short and long texts. Moreover, KE tools that incorporate word position as a feature, such as PositionRank, TextRank, and YAKE, tend to yield suboptimal results when applied to datasets where keywords are not evenly distributed throughout the text [74].

2) EXECUTION TIME

The type of data being analyzed can also have an impact on the execution time of a KE tool. The complexity and size of the data set can affect the performance of the tool, with larger and more complex data sets generally requiring more processing time. Fig. 5 shows the average execution time per document in seconds for all selected KE tools on four benchmark datasets.

The short-text datasets SemEval2017 and Inspec take considerably less amount of time to execute for all KE tools.

TABLE 6. **The evaluation result of selected unsupervised KE tools on four benchmark datasets.**

Tool	Benchmark Dataset	Execution Time (Sec)	Execution Time per Document (Sec)	Matched Keyword per Document	Precision (%)	Recall (%)	F1-score (%)
Tf-Idf	SemEval2010	991.09	4.0786	3.89	38.93	24.99	30.44
	SemEval2017	1448.28	2.9377	**5.11**	**51.1**	29.53	**37.43**
	Inspec	5682.86	2.8414	4.14	41.4	29.34	34.34
	Krapivin2009	9270.33	4.0236	2.36	23.63	**44.28**	30.82
KPMiner	SemEval2010	1048.99	4.3168	4.35	43.5	27.93	34.02
	SemEval2017	1435.96	2.9127	**5.09**	**50.85**	29.39	**37.25**
	Inspec	5729.67	2.8648	4.49	45.52	31.82	**37.46**
	Krapivin2009	9869.34	4.2836	2.69	26.94	**50.47**	35.13
YAKE	SemEval2010	131.69	**0.5419**	2.87	28.68	18.41	22.42
	SemEval2017	18.88	**0.0383**	3.78	37.79	21.84	27.68
	Inspec	76.21	**0.0381**	3.29	32.88	23.3	27.27
	Krapivin2009	1212.65	**0.5263**	1.95	19.54	36.61	25.48
RAKE	SemEval2010	7.52	**0.0309**	1.86	18.56	11.92	14.52
	SemEval2017	0.43	**0.0009**	4.63	46.27	26.74	33.89
	Inspec	1.32	**0.0007**	4	40.23	28.37	33.27
	Krapivin2009	62.48	**0.0271**	1.81	18.1	33.91	23.6
spaCy	SemEval2010	691.65	2.8463	2.75	27.49	17.65	21.50
	SemEval2017	703.6	1.4272	4.22	42.23	24.41	30.94
	Inspec	2812.67	1.4063	3.72	37.33	26.35	30.89
	Krapivin2009	6454.75	2.8015	1.8	17.96	33.66	23.42
TextRank	SemEval2010	692.5	2.8498	2.26	22.55	14.48	17.63
	SemEval2017	668.86	1.3567	4.84	48.36	27.95	35.42
	Inspec	2658.01	1.329	4.23	42.81	30.0	35.28
	Krapivin2009	6490.39	2.817	1.51	15.12	28.33	19.72
SingleRank	SemEval2010	538.52	2.2161	2.54	25.39	16.3	19.85
	SemEval2017	660.0	1.3387	4.63	46.35	26.79	33.95
	Inspec	2640.24	1.3201	4.11	41.47	29.01	34.20
	Krapivin2009	4927.71	2.1388	1.81	18.08	33.88	23.58
TopicRank	SemEval2010	1402.66	5.7723	4.46	44.57	28.61	34.85
	SemEval2017	649.85	1.3182	**5.37**	**53.65**	31.00	**39.30**
	Inspec	2643.44	1.3217	4.51	46.1	31.94	**37.73**
	Krapivin2009	13689.23	4.9415	2.46	24.63	**46.15**	32.12
Topical PageRank	SemEval2010	761.21	3.1326	2.63	26.26	16.86	20.53
	SemEval2017	906.96	1.8397	4.38	43.75	25.29	32.05
	Inspec	3639.06	1.8195	4.02	40.82	28.50	33.56
	Krapivin2009	6977.81	3.0286	1.83	18.32	34.32	23.89
PositionRank	SemEval2010	543.46	2.2365	2.72	27.20	17.46	21.27
	SemEval2017	658.49	1.3357	4.27	42.72	24.69	31.29
	Inspec	2631.48	1.3157	3.82	38.79	27.08	31.89
	Krapivin2009	5090.87	2.2096	1.84	18.41	34.5	24.01
MultipartiteRank	SemEval2010	1754.23	7.2191	4.51	45.06	28.93	35.24
	SemEval2017	664.94	1.3488	**5.36**	**53.63**	31.0	**39.29**
	Inspec	2644.37	1.3222	4.47	45.44	31.68	**37.33**
	Krapivin2009	16749.26	7.2696	2.64	26.4	**49.46**	34.43
KeyBERT	SemEval2010	4573.35	18.8204	3.09	30.95	19.87	24.20
	SemEval2017	470.96	**0.9553**	3.58	35.84	20.72	26.26
	Inspec	1501.01	**0.7505**	3.05	30.48	21.61	25.29
	Krapivin2009	40789.16	17.7036	2.07	20.74	38.86	27.05
KeyBERT (mmr)	SemEval2010	5016.92	20.6458	4.36	43.62	28.01	34.11
	SemEval2017	470.37	**0.9541**	**6.3**	**62.96**	36.39	**46.12**
	Inspec	1504.31	**0.7522**	4.98	49.78	35.28	**41.29**
	Krapivin2009	44458.65	19.2963	2.77	27.75	**51.99**	36.19
KeyBERT (maxsum)	SemEval2010	6209.97	25.5554	3.71	37.08	23.80	28.99
	SemEval2017	3572.86	7.2472	**5.03**	**50.28**	29.07	36.84
	Inspec	14360.47	7.1802	4.2	42	29.77	34.84
	Krapivin2009	57255.09	24.8503	2.28	25.19	**46.42**	32.66

The KE tools that have an execution time of less than a second for all benchmark datasets are RAKE (average 0.0149 s) and YAKE (average 0.286 s). Though both KeyBERT and KeyBERT(mmr) execute in less than a second for shorter text, they take a significantly higher amount of time for longer text. KeyBERT(maxsum) takes the highest amount of execution time as expected for all benchmark datasets. Our experimental result shows that statistical-based KE tools like Tf-Idf, and KPMiner take more time to execute than graph-based tools for the Inspec dataset, which aligns with and supports the findings of Ushio et al. [9]. Among the graph-based tools, TopicRank and MultipartiteRank perform very well for long text datasets, but they take a higher amount of time to execute as well compared to other graph-based

TABLE 7. Effect of exact and partial matching on KE tool's performance.

RAKE / Inspec dataset	Partial Matching	Exact Matching
Precision	40.23	15.84
Recall	28.37	11.17
F-score	33.27	13.1
Matching Time per Document	0.0063	0.0001

tools. So, for long text, we need to balance the trade-off between f-score and execution time.

3) EXACT VS. PARTIAL MATCHING

The common method for evaluating KE system output is to use an exact match to map the keywords in the gold standard to those in the system output, and then to score the output using evaluation metrics. This approach can be effective if the set of keywords is well-defined and covers all relevant terms in the text. A predicted keyword may be a variation of a gold keyword, in that case, the precise match may be an excessively strict requirement. On the other hand, partial matching allows the method to identify terms that are semantically similar to the pre-defined set of keywords. Partial matching is a solution to this problem. Zesch and Gurevych proposed a generalized framework for evaluating KE methods based on approximate keyword matching [75]. Partial matching can improve the accuracy of keyword extraction by capturing important terms that may not be included in the set of keywords. However, it may also result in more false positives and require more computational resources compared to exact matching. The RAKE tool is selected to evaluate the effect of partial matching and exact matching in terms of precision, recall, f1-score, and matching time on the Inspec dataset. Table 7 shows that there is a drastic drop in the performance of the tool as expected for the exact matching. The F-score for partial matching is more than double that of exact matching and the result is consistent with the result of Papagiannopoulou and Tsoumakas [27]. Additionally, it demonstrates that exact matches take less time to complete per document than partial matches because partial matches require extra steps to compute the matches.

4) HYPERPARAMETER TUNING

Hyperparameter tuning is a crucial step for optimizing the performance of any keyword extraction tool. Most keyword extraction tools come with hyperparameters that can be fine-tuned to achieve the best performance on specific datasets or tasks. These hyperparameters can include settings related to tokenization, stop-word removal, part-of-speech (PoS) tagging, stemming, thresholding, and scoring. The best values for these hyperparameters can depend on factors such as the language of the input text, the length of the documents, and the type of keywords desired. Two MultipartiteRank systems with different hyperparameters are selected to evaluate the effect of hyperparameter tuning on the KE task. In the first system, the default threshold of

TABLE 8. Effect of hyper-parameter tuning on KE tool's performance.

System	Hyperparameter	Precision	Recall	F1-score
Multipartite Rank (System 1)	pos= {'NOUN', 'PROPN', 'ADJ'}	45.44	31.68	37.33
Multipartite Rank (System 2)	pos= {'NOUN', 'PROPN', 'ADJ'}, stoplist= list(string.punctuation), threshold=0.74	53.63	31.0	39.29
% change		18.02	-2.15	5.25
TopicRank (System 1)	pos= {'NOUN', 'PROPN', 'ADJ', 'ADV'}	53.65	31.00	39.30
TopicRank (System 2)	pos= {'NOUN', 'PROPN', 'ADJ'}, stoplist= list(string.punctuation), threshold= 0.8	56.03	32.37	41.03
% change		4.44	-4.42	4.40

minimum similarity for clustering is 0.25. In the second system, a punctuation stop-list is added and a threshold of minimum similarity for clustering is set to 0.74. Table 8 shows the result of these two systems, where the f1-score of the second system is improved by more than 5%. The precision also increases in the second system though there is a slight decrease in recall for the second system.

The impact of hyperparameter tuning of the KE tool is further investigated with different hyperparameter settings of the TopicRank tool. In the first system of TopicRank, the default threshold of minimum similarity for clustering is 0.74. In the second system of TopicRank the PoS tags are changed, a punctuation stop-list is added, and a threshold of minimum similarity for clustering is set to 0.8. The result shows that the change in hyperparameter settings increases the f1-score by more than 4%, while there is a decrease in recall. The KE tool's performance won't always improve if hyperparameter changes are made from the default values. So before integrating the KE tool with other applications, it is crucial to appropriately tune the hyperparameters.

5) DATA PREPROCESSING

Data preprocessing is an essential step in any keyword extraction process. It involves cleaning and transforming raw text data to a format that is suitable for KE methods. This step helps to remove irrelevant information from the text, such as stop words, punctuation, and special characters, which may affect the accuracy of the keyword extraction process. Additionally, document preprocessing can also involve techniques like stemming and lemmatization to reduce words to their root form, making it easier for the methods to identify and extract meaningful keywords. Proper document preprocessing ensures that the input data is consistent and reduces noise, which can improve the performance of the KE methods [76], [77]. Several document preprocessing tools can be used for keyword extraction in

TABLE 9. Effect of data pre-processing on KE tool's performance.

	Without Data Pre-processing	With Data Pre-processing	% change
RAKE / SemEval 2017 dataset			
Precision	46.27	51.08	10.39
Recall	26.74	29.52	10.40
F1-score	33.89	37.42	10.42
Execution time per document	0.0009	0.0007	-22.22
Average matched keywords	4.63	5.11	10.37
YAKE / SemEval 2010 dataset			
Precision	28.68	30.21	5.33
Recall	18.41	19.39	5.32
F1-score	22.42	23.62	5.30
Execution time per document	0.5419	0.3961	-26.90
Average matched keywords	2.87	3.02	5.23

Python. Some of the most used tools include NLTK (Natural Language Toolkit), spaCy, Gensim, TextBlob, and Stanford CoreNLP. These document preprocessing tools can help in cleaning and preparing the text data before applying KE methods. Table 9 shows the effect of data preprocessing of RAKE on the SemEval2017 dataset in terms of precision, recall, f1-score, execution time per document, and average matched keywords. In this experiment, the data preprocessing step includes removing punctuation, word tokenization, and lemmatizing words.

The result shows that using data preprocessing steps increases the performance of RAKE on the SemEval2017 dataset. The f1-score increases by over 10% if the data is preprocessed. Also, the execution time decreases by almost 22% for the preprocessed data. The impact of data preprocessing is further investigated by YAKE and evaluated with the SemEval2010 dataset. The result shows that data preprocessing highly impacts the execution time per document and the execution time decreases by about 29% for the preprocessed data, while the precision, recall, f1-score, and average matched keywords increase by more than 5%. It is important to keep in mind that certain KE tools incorporate data preprocessing steps into their technique, such as filtering out stop words, part of speech (PoS) tagging, stemming, or lemmatizing words, therefore adding more data preprocessing may not help to increase the performance overall. Additional data preprocessing might increase the overall execution time of such tools, slowing the application's keyword extraction process.

6) NUMBER OF EXTRACTED KEYWORDS

The selection of number of the `Top_n` keywords also plays a significant role in the performance score. Usually, a lower number of extracted keywords can lead to a higher precision score and a lower recall score. This relation between precision and recall is called the precision-recall trade-off. The number of extracted keywords and their relevance to the text should be balanced to optimize the performance score of a KE tool. Factors such as the target audience, specific application domain, and the length and complexity of the text should be considered while selecting the optimal number of extracted keywords. Three unsupervised KE tools are evaluated to observe the effect of changing the number of extracted keywords. Fig. 6 shows the changes in precision for KPMiner, TopicRank, and MultipartiteRank for four benchmark datasets. It supports the claim that a lower number of extracted keywords leads to a higher precision score. It is better to investigate the f-score that balances the trade-off between the precision and recall score and gives an optimized overall performance score of KE tools. Table 10 shows the changes in the f1-score when the number of extracted keywords changes.

The f1-score of all three KE tools increases as the number of extracted keywords increases. There is a significant increase in the f1-score when the `Top_n` increases from 5 to 10, while there is a moderate increase in the f1-score when `Top_n` increases from 10 to 15. This increasing trend is followed by three datasets except Krapivin2009. This is because the mean keyword per document for the Krapivin2009 dataset is only 5, which means the increase in

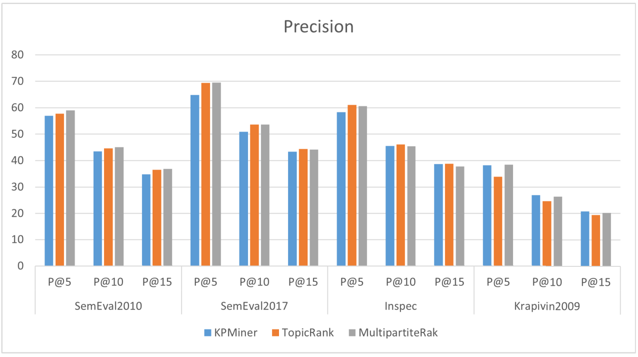


FIGURE 6. Precision of selected KE tools with the change in number of extracted keywords.

TABLE 10. Effect of changing the number of extracted keywords on the KE tool's performance.

Tool	Dataset	F1-score (`Top_n` = 5)	F1-score (`Top_n` = 10)	F1-score (`Top_n` = 15)
KPMiner	SemEval2010	27.68	34.02	34.18
	SemEval2017	29.09	37.25	40.26
	Inspec	30.53	37.46	39.09
	Krapivin2009	37.03	35.13	30.57
TopicRank	SemEval2010	28.08	34.85	35.8
	SemEval2017	31.12	39.30	41.19
	Inspec	31.87	37.73	38.69
	Krapivin2009	32.76	32.12	28.66
Multipartite Rank	SemEval2010	28.68	35.24	36.18
	SemEval2017	31.18	39.29	41.04
	Inspec	31.65	37.33	38.05
	Krapivin2009	37.2	34.43	29.84

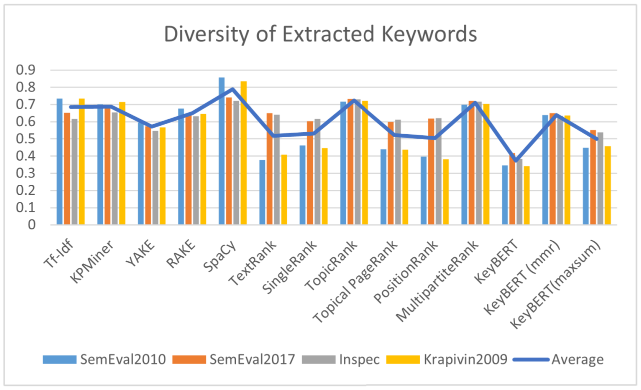


FIGURE 7. **Diversity of extracted keywords of all selected KE tools on four benchmark datasets.**

the number of extracted keywords will only drop the precision score and this leads to a lower f1-score. So, it is also important to understand the dataset before evaluating the KE tools. By selecting the optimal number of extracted keywords, it is possible to optimize the performance of KE methods.

7) DIVERSITY OF EXTRACTED KEYWORDS

Precision, recall, and f-score are the most well-known evaluation metrics and have been used in most of the KE task [27]. These metrics might not be adequate to evaluate the performance of unsupervised KE tools, as the unsupervised KE tools do not rely on ground truth or specific keywords. We introduce another metric to evaluate the performance of unsupervised KE tools referred to as diversity. Diversity measures how distinct the extracted keywords are from one another, indicating whether the KE tool can identify a broad range of keywords or concepts in the text [78]. A high diversity score means that the extracted keywords cover a wide range of topics and keywords found in the text and are not limited to just a few. On the other side, a low diversity score shows that the extracted keywords have a narrow scope and do not adequately cover the text's whole spectrum of keywords and topics. Therefore, diversity can provide a better understanding of the document's content. Diversity can be calculated using the following equation:

$$Diversity = 1 - \frac{\text{Sum of pairwise similarities}}{\text{Total number of pairwise comparisons}} \tag{14}$$

Each extracted keyword is used to make a pair with other extracted keywords to find the similarity between them. Partial fuzzy matching is used to measure the similarity here. Fig. 7 shows the diversity score of extracted keywords of all selected KE tools on four benchmark datasets along with their average. SpaCy scores the highest average diversity score of 0.7880 while there is a change in diversity score for different data types. The most consistent KE tools in terms of diversity across different data types are TopicRank, MultipartiteRank, KeyBERT(mmr), and KPMiner. Among them, TopicRank and MultipartiteRank can achieve more than 70% average diversity, while KeyBERT(mmr) and KPMiner score a little less average diversity. An interesting fact is that statistical-based tools tend to extract more diverse keywords for long-text data while graph-based tools tend to extract more diverse keywords for short-text data.

8) IMPLEMENTATION SOURCE

Python-based keyword extraction tools are available both as open-source and proprietary software. The open-source libraries are freely available and can be easily installed using the Python package manager, pip. Additionally, these libraries come with detailed documentation, which makes it easy for developers to integrate keyword extraction functionality into their code.

Some of the popular sources for KE tools are:

1) PKE (Python Keyphrase Extraction): It is a simple yet effective Python-based open-source toolkit for natural language processing tasks, including keyword extraction. It provides an easy-to-use interface for implementing a variety of statistical and graph-based unsupervised models as well as supervised models for keyword extraction. Additionally, PKE offers preprocessing tools for cleaning and normalizing text data, as well as utilities for evaluating and benchmarking the performance of keyword extraction models.
2) spaCy: A popular Python library for natural language processing tasks, including keyword extraction. Spacy's keyword extraction functionality uses statistical methods based on the frequency and distribution of words and phrases in the input text. It also includes several built-in models for different languages and domains, as well as the ability to customize and train user's models.
3) Gensim: A Python library that specializes in the topic modeling, but also has functionality for keyword extraction. Gensim's keyword extraction module offers several methods for keyword extraction, including TextRank and TF-IDF. Gensim's keyword extraction module also provides options for controlling the number of keywords returned, filtering stop-words and punctuation, and specifying the part-of-speech tags of interest.
4) KeyBERT: A lightweight and easy-to-use Python library for keyword extraction based on BERT embeddings. It can be fine-tuned on domain-specific data to improve the quality of the extracted keywords. Additionally, it provides options to customize the length of the extracted keywords and the number of candidate keywords. It also provides pre-trained models for different languages, making it accessible to non-English language processing tasks.
5) KEX (Keyword Extraction): Another Python library for keyword extraction that uses statistical and graph-based methods. It provides a simple and consistent interface for performing keyword extraction and supports multiple languages. Kex also allows for

customization of the methods by adjusting the hyper-parameters and selecting different stop word lists.

9) BENCHMARK DATASET REPOSITORIES

Benchmark datasets for keyword extraction are essential to evaluate and compare the performance of different KE methods. There are various repositories available for benchmark datasets of keyword extraction. One of the commonly used repositories is the ACL Anthology Network, which is a collection of natural language processing papers and resources, including benchmark datasets for keyword extraction [79]. Another repository is the Text Analysis Conference (TAC) Knowledge Base Acceleration (KBA) track, which provides benchmark datasets for keyword extraction and related tasks [80]. Additionally, several GitHub repositories store the commonly used benchmark dataset for keyword extraction [81], [82], [83].

10) COMMERCIAL API'S

There are several commercial APIs available for keyword extraction that can be used to extract keywords from a text. These APIs usually provide a web-based interface or a REST API that can be used to submit text data for keyword extraction. Some popular commercial APIs for keyword extraction include Google's Natural Language AI, Amazon Comprehend, Microsoft Azure Text Analytics, IBM Watson, TextRazor, MonkeyLearn, OneAI, OpenAI, and many more. These APIs are powered by advanced machine learning algorithms and offer several additional features such as sentiment analysis, entity recognition, and more [84]. However, the use of these APIs usually comes with a cost, and the pricing varies depending on the usage and features provided. Nonetheless, these APIs can be a convenient and efficient way to extract keywords from text data, especially for businesses and organizations with large-scale keyword extraction needs. Unfortunately, as the internal working principle of these commercial APIs' are not known, it is difficult to interpret the performance of the dataset. So, evaluating these commercial APIs is out of the scope of this study.

VII. CONCLUSION

A comparative evaluation of KE tools is crucial for selecting the most appropriate tool for a given task. The performance of keyword extraction tools can be evaluated using various measures such as precision, recall, F1-score, and accuracy. Benchmark datasets such as Inspec, NUS, and SemEval can be used to evaluate the performance of these tools. While open-source tools such as PKE, Gensim, SpaCy, and KeyBERT are readily available, commercial APIs like TextRazor, and OpenAI GPT-3 also provide keyword extraction services. Document preprocessing is also essential in improving the performance of keyword extraction tools. Techniques like stop-word removal, stemming, and part-of-speech tagging can be used to preprocess documents. Hyperparameter tuning is another important aspect of optimizing the performance of KE tools. Therefore, a careful selection of tools and optimization of their parameters can lead to improved performance in keyword extraction tasks.

In this study, existing surveys on KE methods and state-of-the-art literature on KE tool evaluation are reviewed and summarized to have a better understanding of underlying techniques, challenges in keyword extraction, and associated benchmark datasets. The study also includes a broad discussion of KE insights that will give the readers a clear understanding of evaluation metrics, supervised vs unsupervised methods, the impact of assigned keywords, the effect of execution time and data quality, simultaneous learning, and benchmark datasets.

The empirical evaluation of KE tools shows the effect of types of data that are being analyzed to extract keywords, hyperparameter tuning, exact Vs. partial matching, data preprocessing, number of extracted keywords, and execution time. The purpose of this study is to direct the readers in selecting or developing a strategy that is suitable for the application they are aiming for. This study shows that professional indexer-annotated datasets tend to achieve higher performance scores than author or reader-annotated datasets. Another finding of this study is that the graph-based KE tools have difficulties extracting keywords from long text datasets. But TopicRank and MultipartiteRank perform very well on long text datasets because they use topics as nodes of the graph. KeyBERT (mmr) performs better than other tools on all benchmark datasets, still, it has the drawback of taking a significantly higher amount of time for long text datasets. Based on both performance score and execution time, the tools that perform well on short text datasets are: KeyBERT (mmr), TopicRank, MultipartiteRank, and KPMiner. The study also shows that by considering partial matching over exact matching for evaluation, tuning hyperparameters, preprocessing data, and selecting the appropriate number of extracted keywords, KE tools can be optimized to have better performance scores.

REFERENCES

[1] S. A. Babar and P. D. Patil, "Improving performance of text summarization," *Proc. Comput. Sci.*, vol. 46, pp. 354–363, Jan. 2015.

[2] Z. Liu, C. Liang, and M. Sun, "Topical word trigger model for keyphrase extraction," in *Proc. COLING*, Dec. 2012, pp. 1715–1730.

[3] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "KEA: Practical automatic keyphrase extraction," in *Proc. 4th ACM Conf. Digit. Libraries*, Aug. 1999, pp. 254–255.

[4] C. E. Gutierrez, P. M. R. Alsharif, M. Khosravy, P. K. Yamashita, P. H. Miyagi, and R. Villa, "Main large data set features detection by a linear predictor model," *AIP Conf.*, vol. 1618, no. 1, pp. 733–737, Oct. 2014.

[5] S. N. Kim and M.-Y. Kan, "Re-examining automatic keyphrase extraction approaches in scientific articles," in *Proc. Workshop Multiword Expressions Identificat., Interpretation, Disambiguation Appl. (MWE)*, 2009, pp. 9–16.

[6] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2010, pp. 366–376.

[7] Y. Gallina, F. Boudin, and B. Daille, "Large-scale evaluation of keyphrase extraction models," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries*, Aug. 2020, pp. 271–278.

[8] N. Giarelis, N. Kanakaris, and N. Karacapilidis, ''A comparative assessment of state-of-the-art methods for multilingual unsupervised keyphrase extraction,'' in *Proc. 17th IFIP WG 12.5 Int. Conf. Artif. Intell. Appl. Innov. (AIAI)*. Hersonissos, Greece: Springer, Jun. 2021, pp. 635–645.

[9] A. Ushio, F. Liberatore, and J. Camacho-Collados, ''Back to the basics: A quantitative analysis of statistical and graph-based term weighting schemes for keyword extraction,'' 2021, *arXiv:2104.08028*.

[10] M. Nadim, ''Machine learning for empowering community applications and security,'' Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Texas at San Antonio, San Antonio, TX, USA, 2023.

[11] W. D. Abilhoa and L. N. De Castro, ''A keyword extraction method from Twitter messages represented as graphs,'' *Appl. Math. Comput.*, vol. 240, pp. 308–325, Aug. 2014.

[12] S. Lahiri, R. Mihalcea, and P.-H. Lai, ''Keyword extraction from emails,'' *Natural Lang. Eng.*, vol. 23, no. 2, pp. 295–317, Mar. 2017.

[13] C. Y. Lin, ''Rouge: A package for automatic evaluation of summaries,'' in *Proc. Text Summarization Branches Out*, Jul. 2004, pp. 74–81.

[14] M. Litvak and M. Last, ''Graph-based keyword extraction for single-document summarization,'' in *Proc. Workshop Multi-Source Multilingual Inf. Extraction Summarization (Coling)*, Aug. 2008, pp. 17–24.

[15] M. Garg and M. Kumar, ''The structure of word co-occurrence network for microblogs,'' *Phys. A, Stat. Mech. Appl.*, vol. 512, pp. 698–720, Dec. 2018.

[16] J. Piskorski, N. Stefanovitch, G. Jacquet, and A. Podavini, ''Exploring linguistically-lightweight keyword extraction techniques for indexing news articles in a multilingual set-up,'' in *Proc. EACL Hackashop News Media Content Anal. Automated Rep. Gener.*, Apr. 2021, pp. 35–44.

[17] M. Nadim, D. Akopian, and A. Matamoros, ''San Antonio research partnership portal: Evaluating keyword extraction tools to automate matchmaking for community research partnership,'' *Electron. Imag.*, vol. 35, pp. 1–6, Jan. 2023.

[18] K. Dave, S. Lawrence, and D. M. Pennock, ''Mining the peanut gallery: Opinion extraction and semantic classification of product reviews,'' in *Proc. 12th Int. Conf. World Wide Web (WWW)*, May 2003, pp. 519–528.

[19] A. Hulth and B. Megyesi, ''A study on automatically extracted keywords in text categorization,'' in *Proc. 21st Int. Conf. Comput. Linguistics, 44th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2006, pp. 537–544.

[20] K. S. Hasan and V. Ng, ''Automatic keyphrase extraction: A survey of the state of the art,'' in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Jun. 2014, pp. 1262–1273.

[21] S. Siddiqi and A. Sharan, ''Keyword and keyphrase extraction techniques: A literature review,'' *Int. J. Comput. Appl.*, vol. 109, no. 2, pp. 1–6, 2015.

[22] S. Beliga, Meštrović, and S. Martini-Ipi, ''An overview of graph-based keyword extraction methods and approaches,'' *J. Inf. Org. Sci.*, vol. 39, no. 1, pp. 1–20, 2015.

[23] S. K. Bharti and K. S. Babu, ''Automatic keyword extraction for text summarization: A survey,'' 2017, *arXiv:1704.03242*.

[24] Z. Nasar, S. W. Jaffry, and M. K. Malik, ''Textual keyword extraction and summarization: State-of-the-art,'' *Inf. Process. Manage.*, vol. 56, no. 6, Nov. 2019, Art. no. 102088.

[25] Z. A. Merrouni, B. Frikh, and B. Ouhbi, ''Automatic keyphrase extraction: A survey and trends,'' *J. Intell. Inf. Syst.*, vol. 54, no. 2, pp. 391–424, Apr. 2020.

[26] Z. A. Merrouni, B. Frikh, and B. Ouhbi, ''Automatic keyphrase extraction: An overview of the state of the art,'' in *Proc. 4th IEEE Int. Colloq. Inf. Sci. Technol. (CiSt)*, Oct. 2016, pp. 306–313.

[27] E. Papagiannopoulou and G. Tsoumakas, ''A review of keyphrase extraction,'' *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 2, p. e1339, Mar. 2020.

[28] N. Firoozeh, A. Nazarenko, F. Alizon, and B. Daille, ''Keyword extraction: Issues and methods,'' *Natural Lang. Eng.*, vol. 26, no. 3, pp. 259–291, May 2020.

[29] L. Ajallouda, F. Z. Fagroud, A. Zellou, and E. H. Benlahmar, ''Automatic keyphrases extraction: An overview of deep learning approaches,'' *Bull. Electr. Eng. Informat.*, vol. 12, no. 1, pp. 303–313, Feb. 2023.

[30] M. Garg, ''A survey on different dimensions for graphical keyword extraction techniques: Issues and challenges,'' *Artif. Intell. Rev.*, vol. 54, no. 6, pp. 4731–4770, Aug. 2021.

[31] C. Sun, L. Hu, S. Li, T. Li, H. Li, and L. Chi, ''A review of unsupervised keyphrase extraction methods using within-collection resources,'' *Symmetry*, vol. 12, no. 11, p. 1864, Nov. 2020.

[32] M. Nadim, D. Akopian, and A. Matamoros, ''Community research partnership: A case study of San Antonio research partnership portal,'' *Electron. Imag.*, vol. 34, no. 3, pp. 1–6, Jan. 2022.

[33] M. T. Artese and I. Gagliardi, ''What is this painting about? Experiments on unsupervised keyphrases extraction algorithms,'' *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 364, Jun. 2018, Art. no. 012050.

[34] O. Shapira, R. Pasunuru, I. Dagan, and Y. Amsterdamer, ''Multi-document keyphrase extraction: Dataset, baselines and review,'' 2021, *arXiv:2110.01073*.

[35] T. B. Sarwar, N. M. Noor, and M. S. U. Miah, ''Evaluating keyphrase extraction algorithms for finding similar news articles using lexical similarity calculation and semantic relatedness measurement by word embedding,'' *PeerJ Comput. Sci.*, vol. 8, p. e1024, Jul. 2022.

[36] X. Jiang, Y. Hu, and H. Li, ''A ranking approach to keyphrase extraction,'' in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2009, pp. 756–757.

[37] C. Buckley and E. M. Voorhees, ''Retrieval evaluation with incomplete information,'' in *Proc. 27th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2004, pp. 25–32.

[38] A. Kumar, A. Sharma, S. Sharma, and S. Kashyap, ''Performance analysis of keyword extraction algorithms assessing extractive text summarization,'' in *Proc. Int. Conf. Comput., Commun. Electron. (Comptelix)*, Jul. 2017, pp. 408–414.

[39] K. S. Hasan and V. Ng, ''Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art,'' in *Proc. Coling, Posters*, Aug. 2010, pp. 365–373.

[40] H. Zha, ''Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering,'' in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2002, pp. 113–120.

[41] X. Wan, J. Yang, and J. Xiao, ''Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction,'' in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*, Jun. 2007, pp. 552–559.

[42] A. Hulth, ''Improved automatic keyword extraction given more linguistic knowledge,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2003, pp. 216–223.

[43] S. D. Gollapalli and C. Caragea, ''Extracting keyphrases from research papers using citation networks,'' in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2014, vol. 28, no. 1, pp. 1629–1635.

[44] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, ''Yake! Collection-independent automatic keyword extractor,'' in *Proc. 40th Eur. Conf. IR Res. (ECIR)*. Grenoble, France: Springer, Mar. 2018, pp. 806–810.

[45] T. D. Nguyen and M. Y. Kan, ''Keyphrase extraction in scientific publications,'' in *Proc. Asian Digit. Libraries Looking Back Years Forging New Frontiers, 10th Int. Conf. Asian Digit. Libraries (ICADL)*, Hanoi, Vietnam. Berlin, Germany: Springer, Dec. 2007, pp. 317–326.

[46] S. N. Kim, O. Medelyan, M. Y. Kan, T. Baldwin, and L. P. Pingar, ''SemEval-2010 task 5: Automatic keyphrase extraction from scientific,'' in *Proc. 5th Int. Workshop Semantic Eval.* Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 21–26.

[47] G. O. Aquino and L. C. Lanzarini, ''Keyword identification in Spanish documents using neural networks,'' *J. Comput. Sci. Technol.*, vol. 15, no. 2, pp. 55–60, 2015.

[48] O. Medelyan, E. Frank, and I. H. Witten, ''Human-competitive tagging using automatic keyphrase extraction,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 1318–1327.

[49] M. Krapivin, A. Autaeu, and M. Marchese, ''Large dataset for keyphrases extraction,'' Universita Degli Studi Di Trento, Italy, Tech. Rep. DISI-09-055, , 2009.

[50] A. R. Aronson, O. Bodenreider, H. F. Chang, S. M. Humphrey, J. G. Mork, S. J. Nelson, T. C. Rindflesch, and W. J. Wilbur, ''The NLM indexing initiative,'' in *Proc. AMIA Symp.* Bethesda, MD, USA: American Medical Informatics Association, 2000, p. 17.

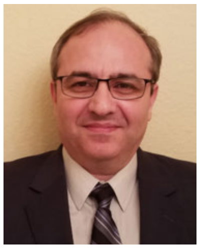
[51] A. T. Schutz, ''Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods,'' M.S. thesis, Digit. Enterprise Res. Inst., National Univ. Ireland, Galway, Ireland, 2008.

[52] I. Augenstein, M. Das, S. Riedel, L. Vikraman, and A. McCallum, ''SemEval 2017 task 10: ScienceIE–extracting keyphrases and relations from scientific publications,'' 2017, *arXiv:1704.02853*.

[53] L. Marujo, M. Viveiros, and J. P. da Silva Neto, "Keyphrase cloud generation of broadcast news," 2013, *arXiv:1306.4606*.
[54] L. Marujo, A. Gershman, J. Carbonell, R. Frederking, and J. P. Neto, "Supervised topical key phrase extraction of news stories using crowdsourcing, light filtering and co-reference normalization," 2013, *arXiv:1306.4886*.
[55] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *Proc. Int. Joint Conf. Natural Lang. Process. (IJCNLP)*, Oct. 2013, pp. 543–551.
[56] O. Medelyan and I. H. Witten, "Domain independent automatic keyphrase indexing with small training sets," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 59, no. 7, pp. 1026–1040, 2008.
[57] O. Medelyan, I. H. Witten, and D. Milne, "Topic indexing with Wikipedia," in *Proc. AAAI WikiAI Workshop*, vol. 1, Jul. 2008, pp. 19–24.
[58] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988.
[59] S. R. El-Beltagy and A. Rafea, "KP-miner: A keyphrase extraction system for English and Arabic documents," *Inf. Syst.*, vol. 34, no. 1, pp. 132–144, Mar. 2009.
[60] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "YAKE! Keyword extraction from single documents using multiple local features," *Inf. Sci.*, vol. 509, pp. 257–289, Jan. 2020.
[61] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," in *Text Mining: Applications and Theory*. Chichester, U.K.: Wiley, 2010, pp. 1–20.
[62] S. Brin, "The PageRank citation ranking: Bringing order to the web," in *Proc. ASIS*, vol. 98, , 1998, pp. 161–172.
[63] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Jul. 2004, pp. 404–411).
[64] X. Wan and J. Xiao, "CollabRank: Towards a collaborative approach to single-document keyphrase extraction," in *Proc. 22nd Int. Conf. Comput. Linguistics (Coling)*, Aug. 2008, pp. 969–976.
[65] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, Jul. 2017, pp. 1105–1115.
[66] K. Sasirekha and P. Baby, "Agglomerative hierarchical clustering algorithm—A," *Int. J. Sci. Res. Publications*, vol. 83, no. 3, p. 83, 2013.
[67] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2010, pp. 366–376).
[68] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[69] F. Boudin, "Unsupervised keyphrase extraction with multipartite graphs," 2018, *arXiv:1803.08721*.
[70] H. Kılıç Ünlü and A. Çetin, "Keyword extraction as sequence labeling with classification algorithms," *Neural Comput. Appl.*, vol. 35, no. 4, pp. 3413–3422, Feb. 2023.
[71] Y. Zhang, M. Tuo, Q. Yin, L. Qi, X. Wang, and T. Liu, "Keywords extraction with deep neural network model," *Neurocomputing*, vol. 383, pp. 113–121, Mar. 2020.
[72] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," *To Appear*, vol. 7, no. 1, pp. 411–420, 2017.
[73] M. Grootendorst, "KeyBERT: Minimal keyword extraction with BERT," Zenodo, Tech. Rep., 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4461265
[74] O. Kabasakal and A. Mutlu, "On the effect of word positions in graph-based keyword extraction," *J. Naval Sci. Eng.*, vol. 17, no. 2, pp. 217–239, 2021.
[75] T. Zesch and I. Gurevych, "Approximate matching for evaluating keyphrase extraction," in *Proc. Int. Conf. RANLP*, Sep. 2009, pp. 484–489.
[76] F. Boudin, H. Mougard, and D. Cram, "How document pre-processing affects keyphrase extraction performance," 2016, *arXiv:1610.07809*.
[77] R. Wang, W. Liu, and C. Mcdonald, "How preprocessing affects unsupervised keyphrase extraction," in *Proc. 15th Int. Conf. Comput. Linguistics Intell. Text Process. (CICLing)*, Kathmandu, Nepal. Berlin, Germany: Springer, Apr. 2014, pp. 163–176.
[78] W. Ni, T. Liu, and Q. Zeng, "Extracting keyphrase set with high diversity and coverage using structural SVM," in *Proc. 14th Asia–Pacific Web Conf. Web Technol. Appl. (APWeb)*, Kunming, China. Berlin, Germany: Springer, Apr. 2012, pp. 122–133.
[79] *ACL Anthology*. Accessed: Apr. 5, 2023. [Online]. Available: https://aclanthology.org/
[80] *Knowledge Base Acceleration Track*. Accessed: Apr. 5, 2023. [Online]. Available: https://trec.nist.gov/data/kba.html
[81] *Datasets of Automatic Keyphrase Extraction*. Accessed: Apr. 5, 2023. [Online]. Available: https://github.com/LIAAD/KeywordExtractor-Datasets
[82] *Keyword Extraction Datasets*. Accessed: Apr. 5, 2023. [Online]. Available: https://github.com/zelandiya/keyword-extraction-datasets
[83] *Benchmark Datasets for Keyphrase Extraction*. Accessed: Apr. 5, 2023. [Online]. Available: https://github.com/boudinfl/ake-datasets
[84] D. Inupakutika, M. Nadim, G. R. Gunnam, S. Kaghyan, D. Akopian, P. Chalela, and A. G. Ramirez, "Integration of NLP and speech-to-text applications with chatbots," *Electron. Imag.*, vol. 33, no. 3, pp. 35-1–35-6, Jun. 2021.

MOHAMMAD NADIM received the B.Sc. degree from the Bangladesh University of Engineering and Technology (BUET), and the M.Sc. and Ph.D. degrees in electrical and computer engineering from The University of Texas at San Antonio (UTSA).

He is currently a Postdoctoral Research Associate with Texas A&M University Central Texas, where he is also with the Center for Cybersecurity Innovation (CCI). His research interests include the development of community applications, software development, machine learning, and cyber security. His awards and honors include the Who's Who at UTSA Award, the UTSA Professional Development Award, and the Harvey E. Najim Center for Innovation and Career Advancement Fellowship.



DAVID AKOPIAN (Senior Member, IEEE) received the Ph.D. degree from the Tampere University of Technology, Finland. He is currently a Professor with The University of Texas at San Antonio (UTSA). Prior to joining UTSA, he was a Senior Research Engineer and a Specialist with Nokia Corporation. His current research interests include digital signal processing algorithms for communication and navigation receivers, positioning, dedicated hardware architectures, and platforms for software defined radio and communication technologies for healthcare applications. He is a fellow of U.S. National Academy of Inventors.

ADOLFO MATAMOROS received the Ph.D. degree in civil engineering from the University of Illinois at Urbana–Champaign, USA, in 1999.

He is currently a Faculty Member (Structure) and a Peter T. Flawn Distinguished Professor with The University of Texas at San Antonio (UTSA). His research interests include the design and behavior of reinforced concrete members, fatigue repair in structural steel bridges, earthquake engineering, and community research partnership.