International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 7579–7601

A SUPERVISED LEARNING APPROACH FOR AUTOMATIC KEYPHRASE EXTRACTION

Muhammad Abulaish and Tarique Anwar

Center of Excellence in Information Assurance
King Saud University
P.O. BOX 92144, Riyadh 11653, Saudi Arabia
{ mAbulaish; tAnwar.c }@ksu.edu.sa

Received August 2011; revised December 2011

Abstract. *Keyphrases, synonymously spoken as keywords, represent semantic metadata and play an important role to capture the main theme represented by a large text data collection. Although authors provide a list of about five to ten keywords in scientific publications that are used to map them to respective domains, due to exponential growth of non-scientific documents either on the World Wide Web or in textual databases, an automatic mechanism is sought to identify keyphrases embedded within them. In this paper, we propose the design of a light-weight machine learning approach to identify feasible keyphrases in text documents. The proposed method mines various lexical and semantic features from texts to learn a classification model. The efficacy of the proposed system is established through experimentation on datasets from three different domains.*
Keywords: Information extraction, Keyphrase extraction, Feature extraction, Machine learning, Text mining

1. **Introduction.** With the exponentially growing World Wide Web (WWW) and increasing trends in completely digitizing modern world, there is an overwhelming growth in textual data and we are getting engulfed into a new problem coined as *information overload*. Although existing search engines and search techniques help to extract information through pattern matching using keywords, they are still in their infancy to provide conceptualization of a large text collection. The dominance of textual data arising from different walks of life (e.g., scientific publications, complaint logs, online news, and discussion forums) and the associated challenges due to their unstructured or semi-structured nature are provoking substantial number of research efforts in the area of text mining and natural language processing (NLP). A major issue associated with handling textual data is to identify key snippets that can be stored in a structured format and analyzed using data mining techniques for various decision making systems (e.g., content-based document classification) and other NLP tasks including document summarization and conceptualization [5], tag-cloud generation, user profiling and so on. This problem has been dealt by researchers [7, 26, 37] through modeling it as a two-class supervised or unsupervised classification problem in which each individual sentence of a document is checked for its suitability to be a part of the document summary. The difficulty in it lies in integrating the semantic information of different sentences. We cannot generate sentences that could precisely describe the information of multiple sentences due to linguistic limitations. As a result, it diverted the research focus towards summarization using individual phrases rather than using sentences, where the role of linguistic rules is minimized. These phrases are better called as *keyphrases*. A *keyphrase* synonymously spoken as *keyword* is a word or a phrase of multiple words, a set of which captures the main topics of a large text

data collection, or in other words, they act as semantic metadata to represent the collection. Nowadays, large text documents or web pages like research articles, commercial web pages, etc. are often tagged using the keywords assigned by their authors. Apart from summarization, there exist a large number of other application areas in which keyphrases are found to be fruitful. Usually keyphrases are assigned manually to text documents by their authors, which is feasible for a small set documents, but it becomes a tedious, time-consuming, and expensive task for a huge collection. Therefore, even of the availability of highly subjective manual keyphrases, automatic methods for keyphrase identification in text documents has importance and it has a wide range of applications in the present scenario.

Automatic keyphrase extraction can be defined as a process for automatic selection of important and topical phrases from within the body of a text document. There is another related term *automatic keyphrase generation*, which is rather a generalized case of automatic keyphrase extraction [36]. In this case, the generated phrases do not necessarily appear in the body of a given document; rather they describe the theme in semantic phrases. Generally, on an average, only about 70%-80% of author-assigned keyphrases appear somewhere in the body of the corresponding document [35]. In task-5 of SemEval-2010, the textual dataset comprised of author-assigned keyphrases out of which only 81% actually appeared in the text [17]. Thus, an ideal keyphrase extraction algorithm could (in principle) generate phrases that match up to this limited number of author-assigned keyphrases, whereas an ideal keyphrase generation algorithm could generate phrases with 100% accuracy, i.e., author-assigned keyphrases that do not occur in the body can be derived. Although we are proceeding gradually towards the keyphrase generation process (by using Wikipedia, `WordNet` and other knowledge bases), we are not yet fully successful to it. Keyphrase extraction technique is widely applied for mining approaches related to text data and efforts are also being made to incorporate similar techniques in other kind of data, such as audio or video streams named as *keyframe extraction* [20].

In this paper, we have pointed out some fundamental and critical aspects for the task of keyphrase extraction and proposed a light-weight and efficient machine learning approach for this task, feasible for real-time environments. Instead of applying full or partial parsing on text documents, which is an inefficient process for lengthy and complex sentences, the proposed method applies n-gram technique to generate candidate phrases and refines them using a set of heuristic rules. For each candidate phrase, various feature values are mined and used to build a binary classification model, which is used to establish the keyphraseness of the candidate phrases extracted from new text documents. The novelty of the proposed method lies in its enrich set of features and their formulation in such a way to produce an effective and accurate keyphrase extraction system. We have conducted experiments on datasets from three different domains to establish the efficacy of the proposed method.

The remaining part of the paper is structured as follows. The following section presents different application areas for automatic keyphrase extraction method. Section 3 presents a brief review of the existing keyphrase extraction methods, which is followed by the functional detail of the proposed method in Section 4. The experimental setup and evaluation of the proposed system is presented in Section 5. Finally, the last section concludes the paper with possible future enhancements.

2. **Application Areas.**

Document keyphrases are widely being used in various information retrieval (IR) and natural language processing (NLP) tasks. Some of the application areas are described in the following paragraphs to justify the vivid importance of the keyphrase extraction systems.

- The act of collecting similar documents into bins is known as *document clustering*, where similarity is decided by some functions on documents, e.g., legal documents, medical documents, and financial documents. Due to existing limitations of the Web search engines in categorizing retrieved documents for a given user query, clustering methods can be used on the client-side or as middle-tier proxies to automatically group the retrieved documents into a list of meaningful categories that are named by some descriptors. Keyphrases can better represent these descriptors, thus enabling the document classification and clustering process [13].
- As discussed earlier, the prime application area of keyphrases lie in document *summarization* due to existence of huge documents collection available either on the Web or textual databases. Titles, keywords, table-of-contents and abstracts might all be considered as different forms of summary; however, a document summary conventionally refers to an abstract-like summary, where the subject is expressed intelligibly. Although Web search engines present list of documents along with summaries (text portions matching query terms), e.g., Google's query-focused snippets, several other attempts have been also made to exploit keyphrases for document summarization [3].
- In large text data, it becomes difficult to jump into the section of text the reader is interested on. Text books are very often supplemented with an extra section called index, which consists of important topics described in the book and their locations. Hence, an alphabetically categorized list of keyphrases, extracted from a collection of documents or from parts of a single long document can be used as to create an index. *Document indexing* is an important application of automatic keyphrase generation in digital libraries [18, 39].
- A thesaurus is a collection of terms from a specific domain of knowledge, which is formally organized in such a way that priori relationships between concepts are made explicit. Originally intended for indexing and retrieving documents, thesauri are increasingly being seen as knowledge bases and used beyond the domain of library. Keyphrases can well be exploited for the task of *thesaurus creation* [18].
- Document management is a serious problem the digital world is facing and it is rising vertically with the growth of internet and corporate intranets. However, researchers believe that to reduce the size of the problem space, metadata can work as a good identifier for management. Query-focused summaries retrieved by Web search engines as well as by software tools in the form of metadata provide crucial information to users for initial relevance judgment so that they can quickly swim into documents demanding further inspection. Keyphrases of a webpage or a simple text document can well serve as *metadata* for indexing and retrieving them efficiently [12, 40].
- In *relevance feedback*, users give additional input on documents (by marking documents in the result set as relevant or not), and this input is used to re-weight the terms in a query to proceed further. On the other hand, in *query refinement*, users give additional input on query words or phrases, possibly suggesting additional query terms. The goal is to refine ambiguous queries intelligently and with ease, for which keyphrases are being used in a fruitful way [33].
- Web resources are very often assigned *tags* or labels by the end-users for organizing and sharing information that is of interest to them based on their content where each label or tag corresponds to a topic in a given document. The organic system of tags assigned by all users of a given web platform is called *folksonomy*. Due to existence of synonymy and polysemy of human languages and the varying degrees of user-assigned expression abilities, tags assigned by different taggers are found to be inconsistent [11] creating an obstacle in digital libraries. This is generally tackled by using suggestion

tools that automatically compute tags for new documents [14, 25, 34]. The process of keyphrase extraction has been proved to be very useful in computing these tags in an automatic way [24].

- With the tremendous growth in e-businesses and competitive environments, commercial website managers spend a lot of efforts on analyzing user trends and try to identify each individual user interests by using *weblogs* maintained on their corporate servers. These weblogs contain a lot of data including client machine internet address, the file browsed and the time details. The painstaking job of analyzing huge weblogs can greatly be relieved by just going through the keyphrases embedded within weblog records.
- On webpages, topical terms are very often *highlighted* using rich text formats by their fonts or colors or else to get a feel for the content of a collection. This provides sensible entry point and facilitates document skimming by visually emphasizing important and relevant phrases, e.g., Google highlights topical terms of the text entered in search query in retrieved webpages by special colors. Identification of these topical terms is generally done by a keyphrase extraction algorithm, which is solely responsible for its soundness.
- Ontology is a knowledge-management structure, which represents domain knowledge in a structured and machine-interpretable form [9]. The proposed keyphrase extraction method is significantly applicable to learn ontological concepts from text documents.

3. **Related Work.** From the discussions in Section 2, we can realize the importance of keyphrases as primary entities for several `NLP` tasks and Web intelligence. Researchers starting simply with `TF-IDF` [30] in its initial phase have discovered various techniques best suited to keyphrase extraction from documents (text files or webpages) in subsequent phases. Frank et al. [10] categorized automatic keyphrase generation into two fundamental approaches – *i*) *keypharse assignment*, where set of keyphrases are limited to a pre-defined vocabulary, and *ii*) *keyphrase extraction*, which collects most indicative phrases embedded within input documents on the basis of their content properties derived through information retrieval techniques. A major limitation for research in the area of keyphrase assignment is the requirement of well-defined and rich thesaurus to be used as predefined vocabulary, the terms of which are assigned as keyphrases for test documents. The state-of-the-art scientific world is having rich thesauruses only in some particular domains; however, none exists to act as a single entity to assign keyphrases for every domain. To create one, is an exhaustive task in its own leading to the research area of thesaurus creation [29]. In contrast, keyphrase extraction is free from the requirement of any external entity, which has led researchers to follow this approach up to large extent. Keyphrase extraction methods usually work in two stages – *i*) a candidate phrase identification stage, which identifies possible phrases in text documents, and *ii*) a selection stage, which selects only few from the collection as keyphrases. The technique behind ranking can either be supervised using a corpus for training or unsupervised. The latter stage is comparatively more extensive than the former one for which researchers have several conflicting views leading to two major categories based on their approaches. Hulth [15] points out that the performance of state-of-the-art keyword extraction algorithm is much lower than those of other NLP-tasks, such as tagging and parsing, and there is plenty of room for improvements in it. Kim and Kan [16] found candidate selection and feature engineering as prime decisive factors for the performance of an algorithm.

`KEA` developed by Frank et al. [10] uses naive Bayes learning method to induce a probabilistic model from the training corpus with exemplar keyphrases. After hunting for

numerous features, they settled on just two lexical features and later added another one. Despite their small sized feature set, because their significance and simplicity, KEA is still comparable to most of the keyphrase extraction systems being developed today. Later, an improved version KEA++ was designed by Medelyan and Witten [22], which enhances KEA by introducing the concept of a domain specific knowledge base using *node degree* for semantic information about the phrases. It exploits the fact that more a phrase is connected to others by a relation via thesaurus terms, the more relevant it becomes for the corresponding context. Turney in his work [36] came up with two algorithms, C4.5 decision tree induction algorithm employing six lexical features and three linguistic features, and GenEx comprising of the *Genitor* genetic algorithm [38] and the *Extractor* keyphrase extraction algorithm [35] that outperforms the first one. Once parameters are tuned through training, Genitor is discarded and GenEx is left with Extractor (GenEx minus Genitor) only. Hulth [15] tried to rule out effective features as few as possible to be comparable to the state-of-the-art techniques and after a series of experiments, she was able to come up with the feature set shown in Figure 1. She added a key linguistic feature to the baseline features of KEA showing remarkable performance improvements. In LAKE system [3], D'Avanzo et al. scored candidates with only two baseline features of KEA. However, the variation lies in the consideration of heads of candidate phrases instead of the phrase itself for calculating feature values. For determining the head for a candidate *principle of headedness* [2] was followed, according to which head of a verb phrase is the main verb in it and in a noun phrase it is the last noun before any post-modifiers. In [27], Nguyen and Kan tried to work especially for scientific publications. Their system focused mainly on the structure of a scientific publication (i.e., with abstract followed by an introduction, related work). They modeled the distribution of a keyphrase among different logical sections as a vector of feature's frequency values for 14 generic section headers and created a *Maximum Entropy* (ME) based classifier using four features to infer the generic section header from their list of 14 headers. Apart from structural features their feature set consists of baselines of KEA, a binary value for acronym and linguistic features as Parts-Of-Speech (POS) sequence and sequence of morphological suffixes. The feature set of Medelyan et al. [23] consists of one addition to that of KEA++, *document-specific keyphraseness*, a knowledge-based feature that represents the likelihood of a phrase being a link in Wikipedia corpus. Litvak and Last [19] explored some graph-based features for their supervised learning approach. After performing stop-word removal and stemming on the candidates as preprocessing tasks, the document is represented in the form of a directed graph, with remaining words as nodes and a directed edge from immediately preceding word to the next, taking into account the sentence terminating punctuation marks. Thereafter, feature values were extracted. According to Medeleyan et al. [23], most important phrases are those that appear both in beginning as well as at the end, and in Maui they introduced one additional lexical feature and two knowledge-based Wikipedia features. *Inverse Wikipedia linkage* is one among them that harnesses information of incoming links to the most likely Wikipedia article for a given candidate phrase to spotlight those phrases that are referred by other commonly used concepts. In [43], Zhang et al. performed this work for multi-word extraction using augmented mutual information. SZTERGAK, a recent work by Berend and Farkas [6] discovered some interesting and worthy features to be exploited to this task shown in Figure 1. Another system committed to scientific articles is HUMB that uses GROBID (GeneRation Of BIbilographic Data) with some basic assumptions. Although most of the works on this problem followed corpus-based approach, single document based approaches were also targeted and exist in literature. In our earlier work [1], we have shown the importance of a keyphrase extraction technique to handle the problem of tag cloud generation which is greatly in limelight these days,

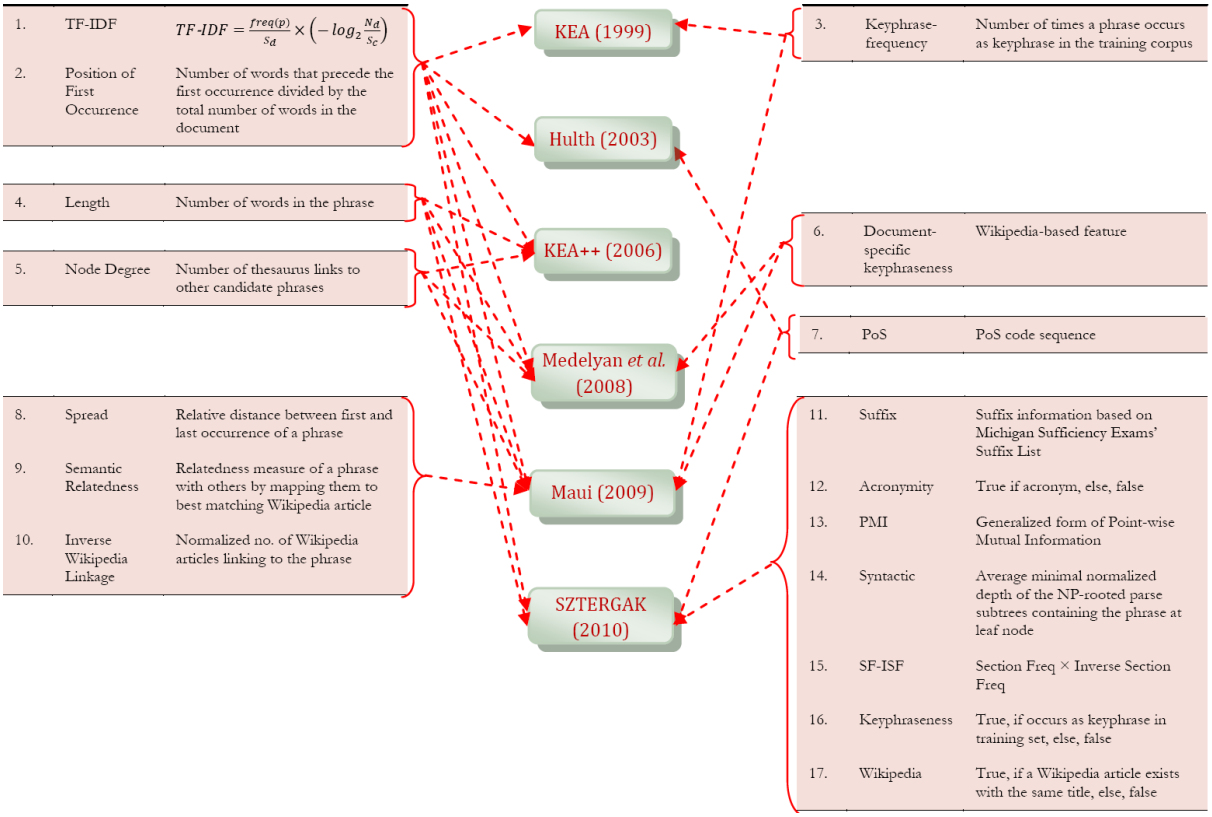


FIGURE 1. Some prominent features for keyphrase extraction

and proposed a method to apply supervised machine learning techniques for keyphrase identification.

It is observed from the previous discussions that exploration of features started from very small sized `KEA` to the long listed `SZTERGAK` and `HUMB`. In all these sets the two baseline features of `KEA` are found to be the most critical and these are solely accountable for more than half of the performance, which can be seen after analyzing the results obtained through `KEA` in the experimental section of this paper. Domain-specific features can also be taken into account when applying keyphrase extraction algorithm to a particular domain. It highlights the inherent properties of the domain of documents from which keyphrases are to be extracted. For example, for keyphrase extraction from medical documents features can be engineered using MeSH (NLM's controlled vocabulary thesaurus) database [31].

3.1. **Role of external knowledge bases in candidate phrase identification.** There are three general categories of techniques for identifying candidate phrases as elucidated by Hulth [15] – *i*) *n-gram filtration* [10, 32], the simplest of all, *ii*) *noun phrase chunking*, based on the fact that nouns act as appropriate content descriptors [27], and *iii*) *POS tag patterning* that arrogates the *n*-gram technique as considering just the arbitrary possibilities rather than semantic relevance, and NP chunking as caring for semantics up to some extent but can be improved further by employing morphological knowledge of constituting words in texts. All of these approaches seem to be very much convincible in their own. However, the matter of concern here is the smartness in judging the candidacy of a phrase. We often face the circumstances where a phrase occurs in texts multiple times, but in one of its several different inflected forms. For example, *artificial intelligence* and *artificially intelligent* are two forms of the same phrase but structurally different for computational purposes, with the common stem as *artifici intellig*. In the parlance of searching and

information retrieval, these variations are generally neutralized by stripping suffices and leaving behind the lone stem common to all using *stemmer*. Google, for instance, uses stemming to search for a string on the Web. Case-folding is also the concern for neutralizing differently capitalized phrases. Apart from neutralizing to syntactic stem, very often it demands mechanisms beyond this. For example, *information science*, *information theory* and *informatics* are semantically similar terms but due to their syntactic difference they will be treated as different candidates by the algorithm. Therefore, some mechanisms are sought to neutralize this kinds of similar terms and identify their relationship strength to exploit in classification task. Medelyan and Witten [22] dealt with this problem using a thesaurus in `KEA++` by identifying the thesaurus terms related to document's contents.

In [24], Medelyan et al. found Wikipedia as a best knowledge base to boost up the candidate phrase identification strategy. Their similarity-based disambiguation technique ranks all n-grams with respect to a *keyphraseness* value calculated by dividing the no. of Wikipedia articles where the n-gram appears as a link by its total no. of appearances. Those having this value greater than a threshold are mapped to matching titles in Wikipedia resolving ambiguities using a similarity measure which are then collected as candidate phrases. Another aspect is to capture the relationship type between related terms like synonyms, antonyms and holonyms. `WordNet` is a useful tool for determining the semantic relatedness between different terms. In [4], Banerjee and Pederson, introduced *extended gloss overlaps* to determine the relatedness of concepts by expanding glosses of concepts with those directly linked by a `WordNet` relation and valued by the number of matching (overlapping) words between definitions (glosses) of two different terms. In [28], Patwardhan and Pederson used co-occurrence information in `WordNet` definitions to generate gloss vectors corresponding to each concept and measured semantic relatedness by the cosine of angles between respective gloss vectors. Therefore, applying this kind of techniques in keyphrase extraction can drastically boost up the task of candidate phrase identification. Zesch et al. [42] introduced Wikitionary for this measure by applying a concept vector based approach, whereas Wubben and Bosch harnessed Wikipedia and `ConceptNet` [41]. Apart from the above knowledge bases, some domain specific knowledge bases can also be employed for domain specific term conflation, e.g., MeSH database can be used for exploring relationships among medical terminologies [31], `agrovoc` thesaurus can be used for terminologies of all subject fields in agriculture, forestry, fisheries, food and related domains [22], HEP for physics terminologies and so on. At the same, exploiting domain-specific knowledge bases restricts it to be deployed in an environment with the same domain. When exploiting domain-independent knowledge bases, the efficiency of the system must be analyzed at an earlier stage to settle down its applicability limitations.

4. **Proposed Keyphrase Extraction Method.** In this section, we present functional detail of the proposed light-weight keyphrase extraction method which follows an algorithmic approach to identify keyphrases in text documents. In line with most of the existing keyphrase extraction systems, it identifies keyphrases using the following four ordered sub-tasks – *document pre-processing*, *candidate phrase identification*, *feature extraction*, and finally, *model learning and keyphrase identification*. Figure 2 depicts the dependencies among these sub-tasks. Further details about each sub-task is presented in the following sub-sections.

4.1. **Document pre-processing.** Before the algorithm is applied, documents need to be transformed into machine-readable format for which some pre-processing steps are followed. To accomplish this, each document goes through *tokenization* followed by *n-gram*

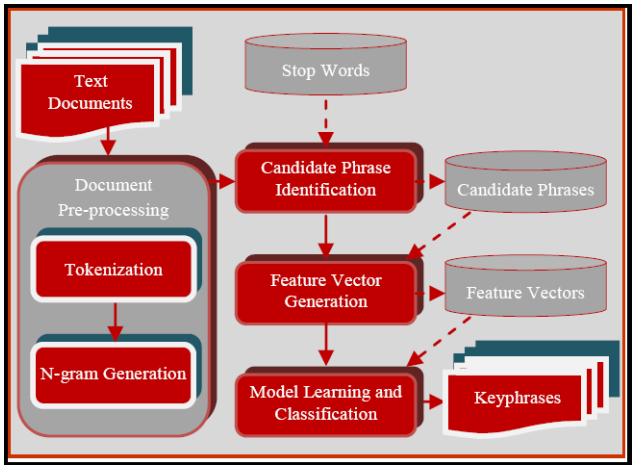


FIGURE 2. Functional details of the proposed approach

generation. Tokenization is a process to fragmentize texts into small-sized chunks. The tokenizer is implemented to work on different file formats including portable document format (pdf) by using `pdftotext` and Web documents by using `HTML parser`. As we are concerned only with textual contents, all the images and their labels are filtered out during the text gathering process. Thereafter, the collected text is tokenized into record-size chunks, boundaries of which are decided heuristically on the basis of the presence of various punctuation marks. On analysis, we observed that in rare cases the keyphrases of any document consist of more than three words. Therefore, ignoring the rare cases in which a keyphrase may contain more than three words, we have restricted the maximum value of n to 3 for generating n-grams (1-, 2- and 3-grams) to achieve a light-weight keyphrase extraction system.

4.2. **Candidate phrase identification.** In this phase, the record-size chunks generated by document processor are analyzed to identify potential candidate phrases that serve as the topical terms to act as a substitute for the whole document text, and feasible keyphrases from this set are selected as the most promising ones in later stages. It is implemented as a two-step process – *phrase processing* to normalize for the noise in it followed by *filtering* to filter out those incapable to compete further. The phrase processing task consists of removing apostrophes, cleaning numerals associated with a phrase at the boundaries, stemming, case folding, etc., whereas the filtering task consists of discarding n-grams containing special characters or symbols in it, 1-grams with less than four letters in it, n-grams containing numerals in between the boundaries of a word, etc.

4.3. **Feature vector generation.** In this step, the set of candidate phrases are transformed to equivalent feature vectors comprising eight feature values. Discovering prominent features for a classification task is a thought provoking task, as it requires to conceive the salient properties of a keyphrase. The state-of-the-art systems are using various kinds of features that can be categorized as *lexical features* computed from the lexical structure of the candidate, *linguistic features* based on the morphological structure, and *knowledge-based features* computed using external knowledge-bases to deduce some semantic information. Lexical features are the most common of all to capture foremost attention and plays the most creditworthy role because of their light-weight computational tasks, domain independency and efficacious results. In our approach, we have explored some features limited to lexical features because of their mentioned importance and come up with their

exact formulations that could produce adequate results for real time environments. This section presents the formulation of different features in subsequent parts.

TF-IDF (W_{tfidf}). It is the most promising among all features, which combines the frequency of a phrase in a particular document with its occurrence in the whole corpus. This score is high for rare phrases that appear frequently in a document and therefore are more likely to be significant. We have used a different version of it than the standard one formulated in [30], as shown in Equation (1), where $f(p_i)$ represents the frequency of phrase p_i, $|D|$ is the length of document D, and the function $countDoc(.)$ returns the number of documents containing the phrase supplied as an argument.

$$W_{tfidf}(p_i) = \frac{f(p_i)}{|D|} \times \left(-\log_2 \frac{\max\{countDoc(p_j)\}}{countDoc(p_i)} \right) \tag{1}$$

Positional Weight (W_{pos}). Phrases occurring either at the beginning or at the end of a document are generally considered as important. So, a positional weight (higher, if occurrence is at either end and lower for comparatively inwards) is assigned to a phrase to reflect its positional importance and calculated using Equation (2), where $occ_f(p_i)$ is the position of the first occurrence of p_i in D.

$$W_{pos}(p_i) = \begin{cases} \left| 1 - \frac{occ_f(p_i)}{\left(\frac{|D|}{2}\right)} \right| & \text{if } occ_f(p_i) \neq \frac{|D|}{2} \\ \left| 1 - \frac{occ_f(p_i)}{\left(\frac{|D|}{2}\right)} \right| + \frac{1}{\frac{|D|}{2}} & \text{otherwise} \end{cases} \tag{2}$$

Cumulative Weight (W_{cum}). Sometimes it is found that a multi-word phrase, even of being important, is not so common to appear in its complete form in texts. We observed that in such cases the constituting words of the phrase possess a high frequency. So, we have tried to capture this information by using Equation (3), where $f(w_j)$ represents the frequency count of the word w_j and $l(p_i)$ is the length of phrase p_i representing number of words in it.

$$W_{cum}(p_i) = \log_2 \left(1 + \frac{\sum_{j=1}^{l(p_i)} f(w_j)}{\max\left\{\sum_{j=1}^{l(p_k)} f(w_j)\right\}} \right) \tag{3}$$

Length (W_{len}). Generally, small-sized phrases are boosted up by some other features due to their tendency to be more frequent. To overcome this biasness, we have considered to assign higher weight to lengthy phrases so that its value for the complete phrase could be higher than the constituting words. This is modeled using Equation (4) for each phrase p_i with length $l(p_i)$.

$$W_{len}(p_i) = \frac{l(p_i)}{\max\{l(p_j)\}} \tag{4}$$

Relatedness (W_{rel}). An important questionable issue in keyphrase extraction task is the integrity of a complete phrase to conclude whether the complete phrase would have more priority than the constituting words that are among other candidates in the set or vice versa. To capture the associativity measure among constituting words for a phrase p_i occurring in D, we have devised this feature and modeled using Equations (5) and (6).

$$W_{rel}(p_i)\begin{cases} = \log_2\left(1+\frac{f(p_i)}{|D|}\right) & \text{if } p_i \text{ is a single word} \\ = \log_2\left(1+\frac{prob(w_1,w_2)}{prob(w_1)\times prob(w_2)}\right) & \text{if } p_i \text{ is a double word, and } w_1, w_2 \in p_i \\ = \log_2\left(1+\frac{prob(w_1,w_2)}{prob(w_1)\times prob(w_2)}\right) & \\ \quad + \log_2\left(1+\frac{prob(w_2,w_3)}{prob(w_2)\times prob(w_3)}\right) & \\ \quad + \log_2\left(1+\frac{prob(w_1,w_2,w_3)}{prob(w_1)\times prob(w_2)\times prob(w_3)}\right) & \text{if } p_i \text{ is a triple word, and } w_1, w_2, w_3 \in p_i \end{cases} \tag{5}$$

$$\begin{aligned} prob(w_j) &= \frac{f(w_j)}{\max\{f(w_r)\}} \\ prob(w_j, w_k) &= \frac{f(w_j, w_k)}{\max\{f(w_r, w_s)\}} \\ prob(w_j, w_k, w_l) &= \frac{f(w_j, w_k, w_l)}{\max\{f(w_r, w_s, w_t)\}} \end{aligned} \tag{6}$$

Capitalization (W_{cap}). Any phrase appearing in a document as its first letter in uppercase is considered as an important one. This feature reflects the casing aspect of a phrase using Equation (7), where $countUpper(p_i)$ returns the number of words in p_i with its first letter in upper case. $l(p_i)$ and $f(p_i)$ represent the length and frequency count of phrase p_i respectively.

$$W_{cap}(p_i) = \frac{\sum \frac{countUpper(p_i)}{l(p_i)}}{f(p_i)} \tag{7}$$

Lifespan (W_{lsp}). It determines the extent of a phrase in a document. The more the gap between the first and last occurrence of a multiple times occurring phrase is, higher will be its score for this feature. It is defined using Equation (8), where $occ_f(p_i)$ and $occ_l(p_i)$ return the position of the first and last occurrence of p_i respectively.

$$W_{lsp}(p_i) = \frac{occ_l(p_i) - occ_f(p_i)}{|D|} \tag{8}$$

Keyphraseness (W_{key}). It is also an important feature and Frank et al. [10] had later included it in `KEA`. It quantifies how often a candidate phrase appears as a keyword in the training corpus. If a candidate phrase already exists in the set of keywords of the training set, it is very likely for this phrase to be a keyphrase in the document itself, for which a higher score in comparison to those not appearing as a keyword in the training set is assigned. It is defined by Equation (9), where $countPerfect(G_i, K)$ gives the number of perfect matches of G_i (the set of 1, 2 and 3-grams of phrase p_i) and K represents the set of keywords in training set.

$$W_{key}(p_i) = \frac{\sum countPerfect(G_i, K)}{\max\{\sum countPerfect(G_j, K)\}} \tag{9}$$

4.4. **Model learning and classification.** In line with existing supervised learning algorithms for keyphrase extraction, our proposed system also works in two phases – *model learning* and *classification*. The first phase, also called training phase, uses the feature vectors of training documents to learn a classification model, which is later used to identify keyphrases in new text documents. The second phase is centered on classification of keyphrases from test documents using the learned model. We have performed experiments

with various classifiers for model learning and classification, but finally settled on naive Bayes and decision tree (C4.5) classifiers due to their best performance.

Turney [36] used C4.5 decision tree for his first experiment and genetic algorithm in GenEx for training and classification, whereas Frank et al. [10] ascertained Bayesian classification approach for machine learning to perform best for his system KEA and therefore used naive Bayes Classifier. Their experiments show that KEA and GenEx have statistically equivalent levels of performance. Most of the later systems like KEA++ [22] (an improved version of KEA), LAKE (Learning Algorithm for Keyphrase Extraction) [3], work of Medelyan et al. [23], and Nguyen and Kan [27] also used naive Bayes Classifier. However, Medelyan et al. found an interesting fact in Maui [24] through their experiments. They found that the selection of a particular classifier to yield the best results depends very much on the available feature set. Their empirical results show that with the baseline features of KEA, naive Bayes produces the best results, but in case of the whole feature set in Maui bagged decision tree is found as the best classifier. Also in HUMB [21], they used bagged decision tree for machine learning as it gave optimum results in comparison to other classifiers. Further, for better performance they applied a post-ranking process to reflect the cohesion among the ranked phrases.

5. **Experimental Results.** In this section, we discuss our experimental setup and evaluation results of the proposed keyphrase extraction method. As our objective is highly focused for a light-weight system with acceptable performance, we have compared its efficacy and correctness with KEA [10] and KEA++ [22], that are still widely used in most of the standard information retrieval systems due to their real-time feasibility. The classification task of the problem of keyphrase extraction is highly imbalanced in nature, i.e., there is a huge difference between the number of positive and the negative class instances. And, even simply declaring all the instances as negative will result to almost 95%-99% accuracy, due to which system evaluation in terms of accuracy does not make a good sense for this problem. Therefore, we have also used standard information retrieval performance measures *precision*, *recall*, and f_1-*measure* for evaluation of our proposed system. From classification results we calculate the true positive TP (number of correct keyphrases the system identifies as correct), the false positive FP (number of incorrect keyphrases the system falsely identifies as correct), and the false negatives FN (number of correct phrases the system fails to identify as correct). These parameters are used to calculate the value of *precision*, *recall* and f_1-*measure* using Equations (10)-(12) respectively. The *macro-average* values for a complete dataset are computed by adding up all the individual TF, FP and FN values and then calculating the *precision*, *recall* and f_1-*measure* values from them.

$$precision(\pi) = \frac{TP}{TP + FP} \tag{10}$$

$$recall(\rho) = \frac{TP}{TP + FN} \tag{11}$$

$$F_1\text{-}measure(F_1) = 2 \times \frac{\pi \times \rho}{\pi + \rho} \tag{12}$$

5.1. **Analysis of TF-IDF.** As far as the field of *Information Retrieval* (IR)is concerned, the immense contribution of TF-IDF [30] can never be overlooked, as is evident from its significant role in various NLP tasks, Web search strategies, etc. In the same way, it has found its place in the task of keyphrase extraction too. It can be observed in Figure 1 that since from the inception of the keyphrase extraction problem the TF-IDF measure is included as a feature in every subsequent system. It is very much a pragmatic measure to integrate the two different elementary concepts – *local presence* and *global presence*. In

TABLE 1. Ranking result on ten different variations of TF-IDF

	Top-3($\pi/\rho/F_1$)	**Top-5**($\pi/\rho/F_1$)	**Top-7**($\pi/\rho/F_1$)	**Top-9**($\pi/\rho/F_1$)
TF-IDF1	13.75/07.90/10.03	11.63/11.11/11.36	09.91/13.26/11.34	09.30/16.01/11.77
TF-IDF2	13.75/07.90/10.03	11.63/11.11/11.36	09.91/13.26/11.34	09.30/16.01/11.77
TF-IDF3	14.58/08.36/**10.63**	11.88/11.35/**11.61**	09.82/13.14/11.24	08.68/14.93/10.98
TF-IDF4	14.17/08.12/10.32	11.63/11.11/11.36	10.27/13.74/**11.75**	09.38/16.13/**11.86**
TF-IDF5	14.17/08.12/10.32	11.63/11.11/11.36	10.27/13.74/**11.75**	09.38/16.13/**11.86**
TF-IDF6	14.38/08.24/10.48	11.25/10.75/10.99	09.38/12.54/10.73	08.33/14.34/10.54
TF-IDF7	13.96/08.01/10.18	11.25/10.75/10.99	09.10/12.19/10.42	08.13/13.98/10.28
TF-IDF8	13.96/08.01/10.18	11.25/10.75/10.99	09.10/12.19/10.42	08.13/13.98/10.28
TF-IDF9	12.71/07.29/09.27	09.75/09.32/09.53	07.86/10.51/08.99	06.94/11.95/08.78
TF-IDF10	13.75/07.90/10.03	11.63/11.11/11.36	09.91/13.26/11.34	09.30/16.01/11.77

TF-IDF1 = $\frac{f(p_i)}{|D|} \times \left(\log_2 \frac{|C|}{countD(p_i)}\right)$

TF-IDF2 = $\frac{f(p_i)}{\max\{f(p_j)\}} \times \left(\log_2 \frac{|C|}{countD(p_i)}\right)$

TF-IDF3 = $\frac{f(p_i)}{f(p_i)+1} \times \left(\log_2 \frac{|C|}{countD(p_i)}\right)$

TF-IDF4 = $\frac{f(p_i)}{|D|} \times \left(\log_2 \frac{\max\{countD(p_j)\}}{countD(p_i)}\right)$

TF-IDF5 = $\frac{f(p_i)}{\max\{f(p_j)\}} \times \left(\log_2 \frac{\max\{countD(p_j)\}}{countD(p_i)}\right)$

TF-IDF6 = $\frac{f(p_i)}{f(p_i)+1} \times \left(\log_2 \frac{\max\{countD(p_j)\}}{countD(p_i)}\right)$

TF-IDF7 = $\frac{f(p_i)}{|D|} \times \left(\log_2 \frac{countD(p_i)+1}{countD(p_i)}\right)$

TF-IDF8 = $\frac{f(p_i)}{\max\{f(p_j)\}} \times \left(\log_2 \frac{countD(p_i)+1}{countD(p_i)}\right)$

TF-IDF9 = $\frac{f(p_i)}{f(p_i)+1} \times \left(\log_2 \frac{countD(p_i)+1}{countD(p_i)}\right)$

TF-IDF10 = $\left(\frac{1}{2} + \frac{f(p_i)}{2 \times \max\{f(p_j)\}}\right) \times \left(\log_2 \frac{|C|}{countD(p_i)}\right)$

FIGURE 3. Ten different variations of TF-IDF

fact, nearly half of the credit for any information retrieval system goes to this measure and researchers are still looking for further improvements to enrich its formulation, if it can be made possible. On analysis, we found that several variations of `TF-IDF` are being used for related tasks. We reached on to its ten different formulations, as given in Figure 3, and in order to determine their exact experimental significance, we tested all of them individually to rank keyphrases based on this single value. This experiment was carried out on a dataset of 160 `CSTR` abstracts and results are presented in Table 1 in terms of the standard `IR` metrices. In Table 1, π, ρ and F_1 are used to represent *precision*, *recall* and f_1-*measure* values.

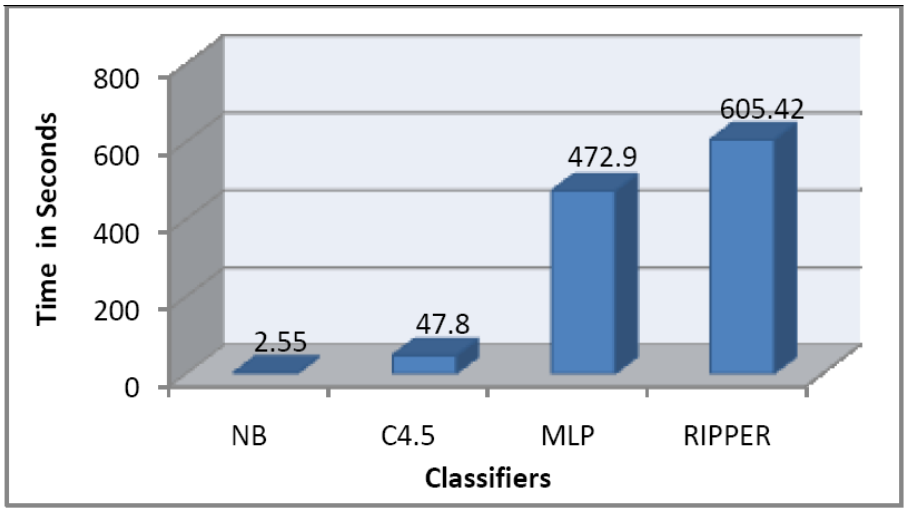
- **Observation 1**: It can be observed from Table 1 that the performance measure values for `TF-IDF1` and `TF-IDF2`, which have only difference in the denominator of the `TF` part, are exactly same for all cut-off values. Similar behaviour is found with pairs `(TF-IDF4, TF-IDF5)` and `(TF-IDF7, TF-IDF8)`. On the basis of these pair-wise similarities, we can conclude that in the TF part of the `TF-IDF` formulation, both the denominators, and the size of document and the highest frequency of any phrase in that document have exactly an equal effect on the whole value.

- **Observation 2**: Although, there is a major variation in the TF part of TF-IDF10, the performance measure values for TF-IDF10 are exactly same as that of the TF-IDF1 and TF-IDF2 for all cut-off values. This proves that they have an equal effect.
- **Observation 3**: The best results are obtained using TF-IDF3 for top-3 and top-5 phrases, whereas TF-IDF4 and TF-IDF5 excel for top-7 and top-9 phrases.

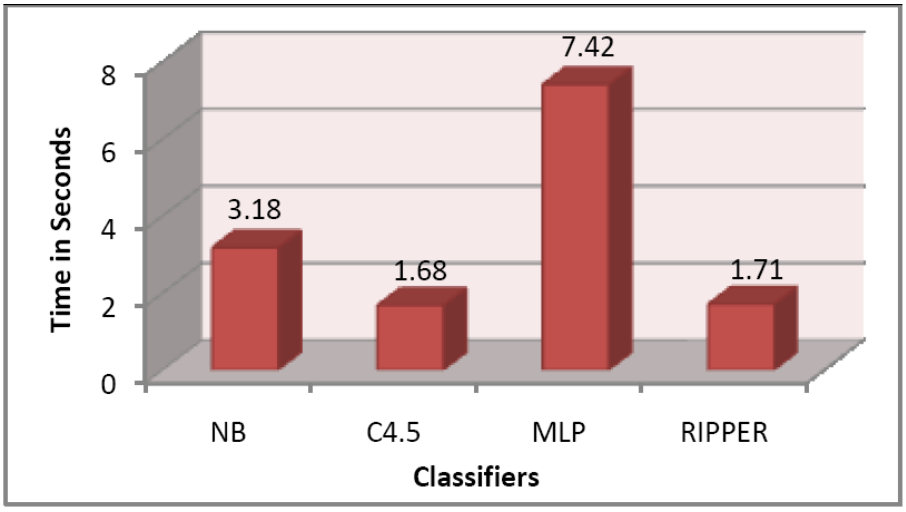
5.2. **Experimentation with Wikipedia documents.** In this experiment, we used a Wikipedia crawler developed by one of our team member to crawl Wikipedia pages related to a topic of interest supplied as an argument. We crawled a total of 500 pages for the term, *information retrieval*, out of which 490 documents with rich contents were considered for training and another set of 10 documents for testing. During the crawling process, in addition to HTML tags noisy texts along with images were filtered out. We assumed the anchor text of each of the hyperlinks on the pages as a text block to convey some important information for which it is emphasized to go further inside that topic. All these anchor texts from each page were collected and analyzed to collect the manual keyphrases from them. From the collected set, some of them were dropped, some others not present among them were added, while some were even cleaned and processed to make a complete sense. In this way, we generated our dataset for experimentation and comparison of our system with KEA.

Once we settled with the feature set, we started experimenting with some prominent classifiers best suited for the classification task. For this, we considered four different classifiers to test with, *naive Bayes* (a simple probabilistic classifier based on Bayes theorem), C4.5 *decision tree* (an extension of the basic ID3 algorithm), *multilayer perceptron* – MLP (a feed forward artificial neural network model with one input layer, one output layer and one or more hidden layers), and *Repeated Incremental Pruning to Produce Error Reduction* – RIPPER (a propositional rule learner). For determining real-time applicability of the approach using these classifiers, time consumption by the classifiers is an important concern. Figure 4 shows their time consumptions during our experiment. Naive Bayes, being the simplest of all consumes 2.55 seconds, the shortest time duration of all to train the model, whereas RIPPER takes the longest. The major demerit of naive Bayes is its longer testing time requirement. While all other classifiers have comparatively lesser testing time than their training time, naive Bayes shows an opposite symptom and its testing time is much higher than that of the others. Since training needs to be done only once for building the model, longer training time is not a big issue. However, keyphrase extraction is to be done repetitively depending on the application for which it must be taken care of. Focusing on this strategy, C4.5 seems to be the best classifier for deployment in real-time systems. We found naive Bayes and C4.5 decision trees as the computationally most efficient algorithms of all to produce satisfactory results.

The other and foremost concern is accuracy of the approach with these classifiers. As discussed earlier, we have used the standard information retrieval performance measures to evaluate our results. For each of the four classifiers, the performance measure values in terms of *precision*, *recall* and f_1-*measure* are shown in Table 2. It can be observed from Table 2 that multilayer perceptron produces the highest precision value, but at the same time its recall value is lowest resulting in lowest f_1-measure value to 31.8%. Recall value is achieved highest with naive Bayes classifier whose difference with its precision value is the narrowest making its f_1-measure value to 33.8%. In terms of f_1-measure, the best result is achieved with C4.5, with 48.1%, 30.7% and 37.5% as its precision, recall and f_1-measure respectively. Hence, as mentioned in Maui [24], for our feature set C4.5 performs the best and the performance of naive Bayes (next highest f_1-measue value) is also considerable. The advantage with C4.5 is that in addition to producing good results,



(a) Training time



(b) Testing time

FIGURE 4. Comparison of time requirements

TABLE 2. Comparison of four different classifiers in terms of IR metrices

Classifier	**Precision** (π)	**Recall** (ρ)	**F-measure** (F_1)
Naive Bayes	0.354	**0.322**	0.338
`C4.5`	0.481	0.307	**0.375**
Multilayer Perceptron	**0.558**	0.222	0.318
`RIPPER`	0.416	0.274	0.330

its time requirement for testing is lowest. Figure 5 presents ROC curves of all the four classifiers, where we can see the relationships between their true positive and false positive rates and visualize their comparative accuracy in terms of these rates.

Getting naive Bayes and `C4.5` as the most suitable classifier for our feature set, we carried out further experiments using only these two classifiers. The individual results for each of the ten test documents are presented in Table 3. The macro-average values

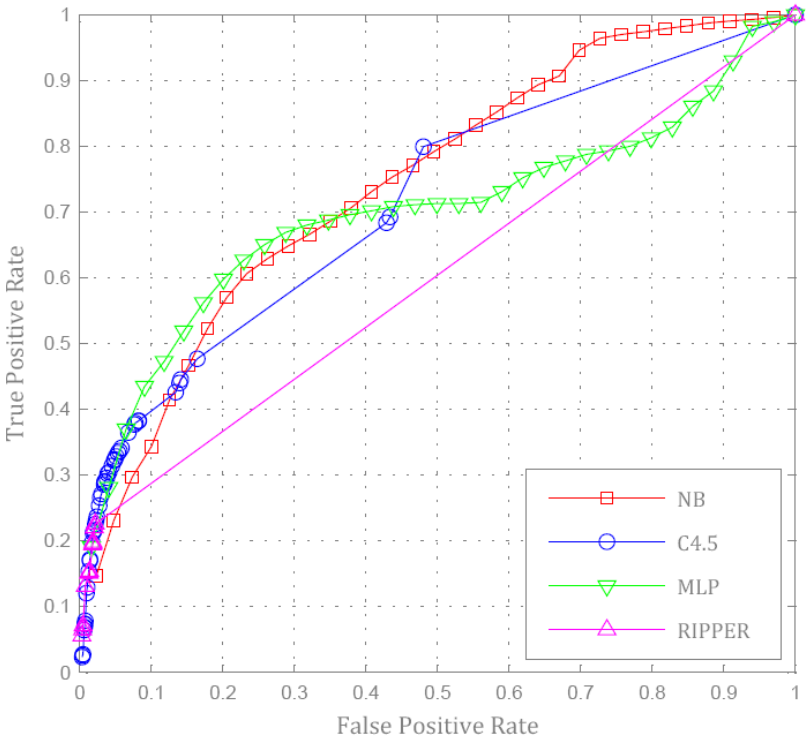


FIGURE 5. ROC curves of experimented classifiers

TABLE 3. Comparison of proposed system with KEA on Wikipedia crawled documents

Document no.	**Proposed**$_{\texttt{NB}}$ ($\pi/\rho/F_1$)	**Proposed**$_{\texttt{C4.5}}$ ($\pi/\rho/F_1$)	KEA ($\pi/\rho/F_1$)
Doc-1	0.50/0.39/0.44	0.47/0.29/0.36	0.42/0.32/0.36
Doc-2	0.24/0.37/0.29	0.47/0.57/0.52	0.26/0.40/0.31
Doc-3	0.33/0.30/0.31	0.39/0.22/0.28	0.26/0.23/0.25
Doc-4	0.40/0.36/0.38	0.48/0.22/0.30	0.22/0.21/0.21
Doc-5	0.23/0.40/0.29	0.53/0.52/0.52	0.18/0.31/0.23
Doc-6	0.30/0.37/0.33	0.48/0.37/0.42	0.24/0.29/0.26
Doc-7	0.50/0.30/0.37	0.56/0.30/0.39	0.52/0.31/0.38
Doc-8	0.50/0.25/0.33	0.48/0.25/0.33	0.32/0.16/0.21
Doc-9	0.34/0.36/0.35	0.41/0.36/0.38	0.28/0.29/0.28
Doc-10	0.30/0.27/0.28	0.52/0.22/0.31	0.28/0.26/0.27
Macro-Average	0.35/**0.32**/0.34	**0.48**/0.31/**0.37**	0.29/0.26/0.28

present summary of overall results on this dataset, where we can see that our approach using both the naive Bayes and C4.5 classifiers outperform KEA establishing its efficacy.

Although we found our approach to be showing good results for a real-time environment, it is very important to figure out its gist and know about its core components. As the main credit of the algorithm goes to feature extraction task, we performed experiments to find out contributions of each feature towards the overall performance. Excluding each of them one by one, we conducted eight rounds of experiments to find out F_1-measure for each combination and computed its difference from the performance value obtained with the complete set of features. These differences rule out their importance in the task.

TABLE 4. Contribution of individual features

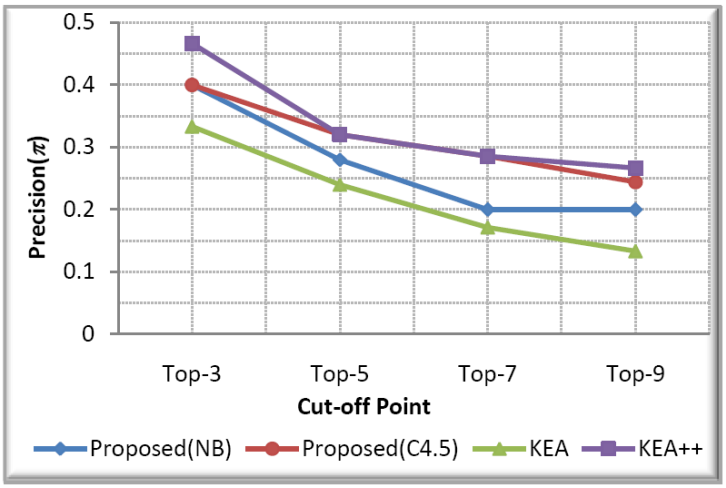
Features	**Proposed**$_{\texttt{NB}}$ (F_1)	**Difference**	**Proposed**$_{\texttt{C4.5}}$ (F_1)	**Difference**
All features	0.338		0.375	
$-W_{tfidf}$	0.315	0.023	0.361	0.014
$-W_{pos}$	0.309	0.029	0.354	0.021
$-W_{key}$	0.294	0.044	0.303	0.072
$-W_{len}$	0.320	0.018	0.355	0.020
$-W_{fre}$	0.330	0.008	0.366	0.009
$-W_{cum}$	0.321	0.017	0.347	0.028
$-W_{rel}$	0.313	0.025	0.329	0.046
$-W_{cap}$	0.338	0.000	0.372	0.003
$-W_{lsp}$	0.336	0.002	0.367	0.008

Table 4 presents the experimental values, where we can see that for the system trained on `C4.5`, the difference is highest when *keyphraseness* feature is excluded. This reflects the pronounced impact of this feature. The next noticeable features are *relatedness* and *cumulative weight* showing a fall of 4.6% and 2.8% respectively in f_1-measure values when they are excluded from the feature set. Going through all these values, we can judge the importance of each of them. During our analysis we noticed that a major flaw in `KEA` is its tendency to rank the small-sized phrases higher than those with comparatively larger in size, which breaks up large sized important contextual phrases to single words. We have worked to sort out this flaw by our W_{rel} and W_{cum} measures, and the values in Table 4 prove their importance.

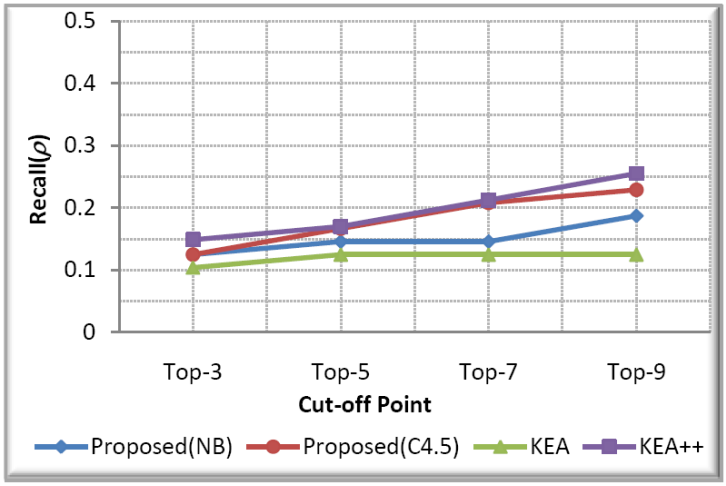
5.3. **Comparative analysis on documents from agriculture domain.** As discussed in Section 3, `KEA` and `KEA++` are the widely used standard techniques, whereas `Maui` employs Wikipedia heavily to compute some of its feature values. `KEA++` also uses external thesaurus links which limits its scope to the domain of the thesaurus used. In this section, we present a comparative evaluation of the proposed system with the existing systems `KEA`[1] and `KEA++`[2], trained on a set of 20 agricultural documents and tested on 5 other documents from the same domain, using `agrovoc` thesaurus for thesaurus links in `KEA++`. This dataset is collected from the webpages of `KEA` itself and the results of each of the 5 individual documents are presented in Table 5. The last row of this table presents *macro-averaged* values for each category. Macro-averaged precision and recall values are computed by adding up all true positives, false positives and false negatives and then applying metric formulas on them. Macro-averaged f_1-measure values is calculated as the harmonic mean of the macro-averaged precision and recall values. Keyphrases for a document, in general, are declared very few in numbers, however it also depends on the application and the strategy. In our experiment, we have extracted the top 3, 5, 7 and 9 most promising keyphrases separately and compared each set with those of `KEA` and `KEA++` in terms of precision (π), recall (ρ) and f_1-measure (F_1). From the macro-average row of Table 5 it can be observed that in all the four cases of top three to nine keyphrases, `KEA++` produces the best values followed by our proposed system and the worst results are shown by `KEA`. It proves that our feature set is far more rich and sound than that of `KEA`, but at the same time not as much as `KEA++`. Its very clear that the only lack of our proposed

[1] *Version 3.0* is a java implementation of the original `KEA` by Frank et al. in 1999

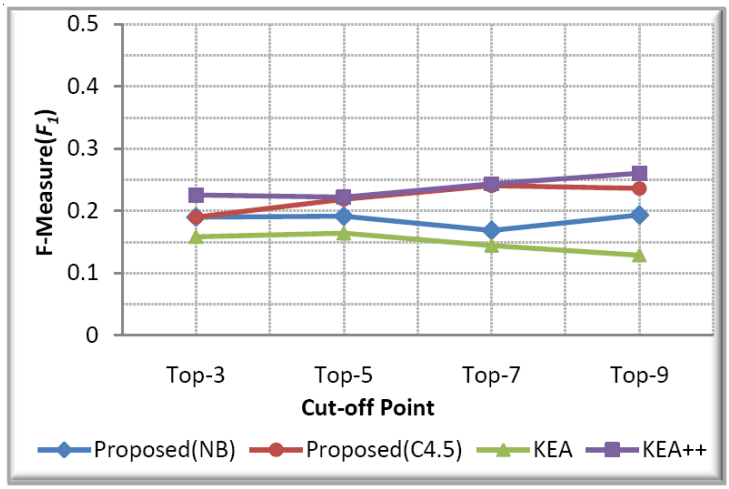
[2] *Version 5.0* is the java implementation of `KEA++` by Medeleyan and Witten in 2006



(a) Precision



(b) Recall



(c) F_1 measure

FIGURE 6. Comparative performance curves on agricultural documents

TABLE 5. Comparison of the proposed system with KEA and KEA++ on documents from agriculture domain

Cut-off Point	**Proposed**$_{\texttt{NB}}$ ($\pi/\rho/F_1$)	**Proposed**$_{\texttt{C4.5}}$ ($\pi/\rho/F_1$)	KEA ($\pi/\rho/F_1$)	KEA++ ($\pi/\rho/F_1$)
Document 1				
Top-3	0.33/0.11/0.17	0.67/0.22/0.33	0.33/0.11/0.17	0.33/0.11/0.17
Top-5	0.40/0.22/0.29	0.42/0.22/0.29	0.20/0.11/0.14	0.20/0.11/0.14
Top-7	0.29/0.22/0.25	0.29/0.22/0.25	0.14/0.11/0.13	0.14/0.11/0.13
Top-9	0.22/0.22/0.22	0.22/0.22/0.22	0.11/0.11/0.11	0.11/0.11/0.11
Document 2				
Top-3	0.67/0.20/0.31	0.33/0.10/0.15	0.33/0.10/0.15	0.33/0.10/0.15
Top-5	0.40/0.20/0.27	0.40/0.20/0.27	0.20/0.10/0.13	0.40/0.20/0.27
Top-7	0.29/0.20/0.24	0.29/0.20/0.24	0.14/0.10/0.12	0.29/0.20/0.24
Top-9	0.22/0.20/0.21	0.22/0.20/0.21	0.11/0.10/0.11	0.22/0.20/0.21
Document 3				
Top-3	0.33/0.20/0.25	0.33/0.20/0.25	0.67/0.40/0.50	1.00/0.75/0.86
Top-5	0.20/0.20/0.20	0.20/0.20/0.20	0.40/0.40/0.40	0.60/0.75/0.67
Top-7	0.14/0.20/0.17	0.29/0.40/0.33	0.29/0.40/0.33	0.43/0.75/0.55
Top-9	0.11/0.20/0.14	0.33/0.60/0.43	0.22/0.40/0.29	0.33/0.75/0.46
Document 4				
Top-3	0.33/0.14/0.20	0.33/0.14/0.20	0.33/0.14/0.20	0.33/0.14/0.20
Top-5	0.20/0.14/0.17	0.40/0.29/0.33	0.40/0.29/0.33	0.20/0.14/0.17
Top-7	0.14/0.14/0.14	0.29/0.29/0.29	0.29/0.29/0.29	0.14/0.14/0.14
Top-9	0.22/0.29/0.25	0.22/0.29/0.25	0.22/0.29/0.25	0.11/0.14/0.13
Document 5				
Top-3	0.33/0.06/0.10	0.40/0.13/0.19	0.00/0.00/0.00	0.33/0.06/0.10
Top-5	0.20/0.06/0.09	0.32/0.17/0.22	0.00/0.00/0.00	0.20/0.06/0.09
Top-7	0.14/0.06/0.08	0.29/0.21/0.24	0.00/0.00/0.00	0.43/0.18/0.25
Top-9	0.22/0.12/0.15	0.24/0.23/0.24	0.00/0.00/0.00	0.56/0.29/0.38
Macro-Average				
Top-3	0.40/0.13/0.19	0.40/0.13/0.19	0.33/0.10/0.16	0.47/0.15/0.23
Top-5	0.28/0.15/0.19	0.32/0.17/0.22	0.24/0.13/0.16	0.32/0.17/0.22
Top-7	0.20/0.15/0.17	0.29/0.21/0.24	0.17/0.13/0.14	0.29/0.21/0.24
Top-9	0.20/0.19/0.19	0.24/0.23/0.24	0.13/0.13/0.13	0.27/0.26/0.26

system is the application of domain knowledge to judge the relevance of domain-specific terms, whereas the other one uses thesaurus links for this. These links works very well when the domain of the application matches with its domain-specific thesaurus, but in case it needs to be applied to a multi-domain environment or an environment with domain other than the available thesaurus, KEA++ will come up with adverse effects to produce biased results, whereas our approach being free from this limitation shows consistent results irrespective of the application environment which is proved by results presented in the following section. For our approach, we have presented the results with two most

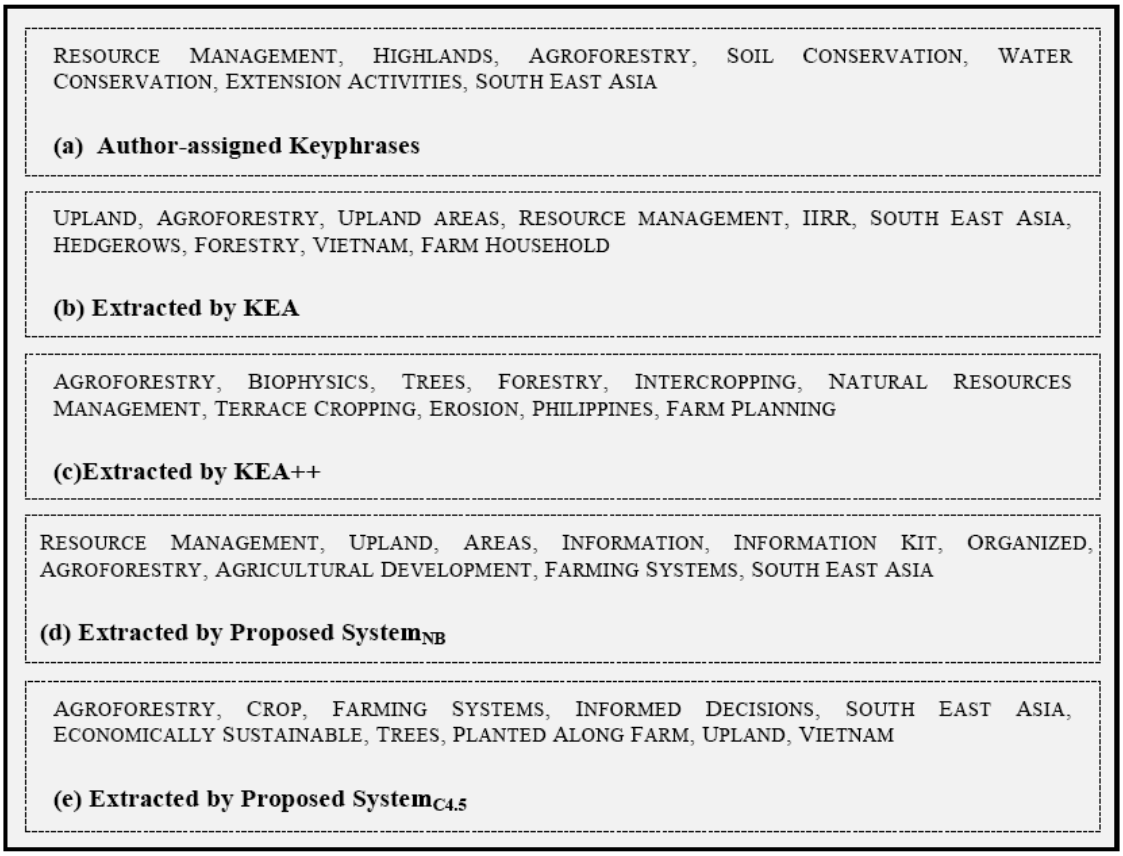
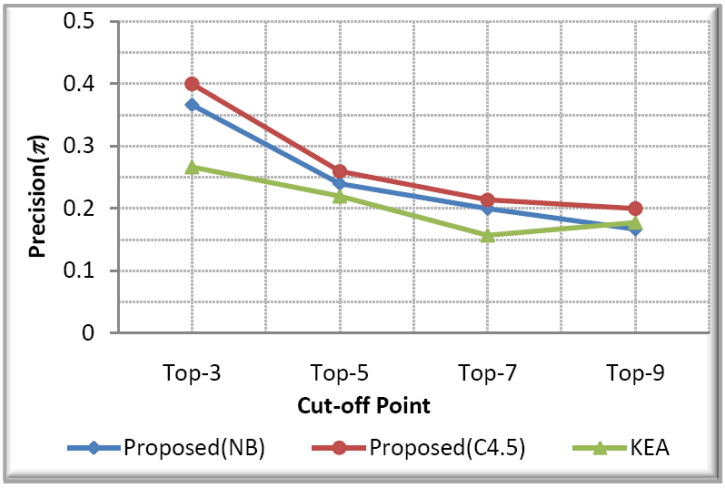


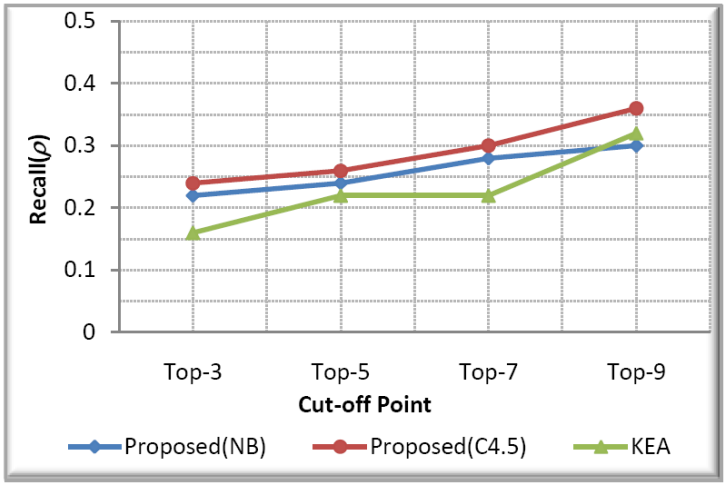
FIGURE 7. Extracted keyphrases from a documents

suitable classifiers, naive Bayes and `C4.5`. Our macro-averaged f_1-measures for top 3, 5, 7 and 9 phrases are 19%, 19%, 17% and 19% respectively with naive Bayes, whereas 19%, 22%, 24% and 24% respectively for `C4.5`. For top 5 phrases our approach with both the classifiers is at par with `KEA++` and for top 7 phrases our approach using `C4.5` is at par with `KEA++`, which shows that our results are very much acceptable even without the use of any domain knowledge. Their comparative performance curves in are shown in Figure 6. Figure 7 shows some keyphrases extracted from one of the five documents by all the four approaches along with those assigned by the author. Analyzing keyphrases extracted by our system, we found that although all of them do not match with the author assigned keywords (it also happens with `KEA` and `KEA++`), on their own they convey almost the same basic theme.

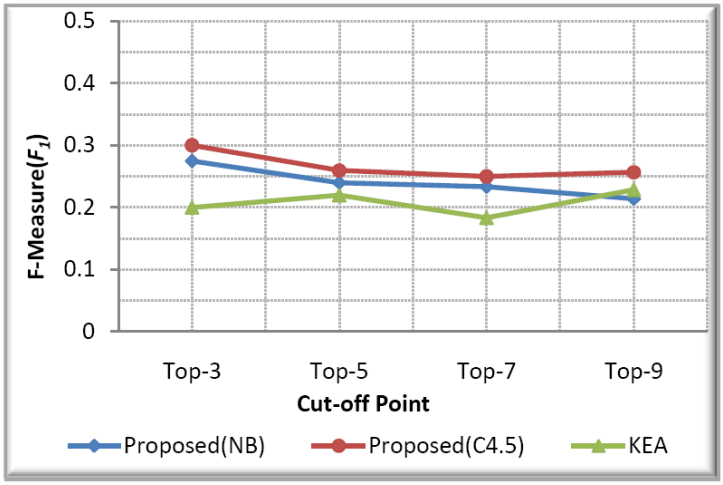
5.4. **Experimentation with documents from scientific domain.** In our third experimentation, we have considered scientific papers from the domain of medical sciences, which generally have a rich structure. Although we do not employ any feature that uses the structural information of text documents, the obtained results are found to be very satisfactory and encouraging. To generate a dataset, we collected 200 publically available research papers in the domain of medical sciences. The pdf files were converted to plain text files using `pdftotext` converter, from which the text contents and corresponding author-assigned keywords assumed as gold standard were collected separately. Out of these 200, we selected 190 documents for training and 10 documents for testing. A summarized view of the results obtained from this dataset is presented in Table 6 and pictorially shown in Figure 8. It can be observed that for all cut-off points our proposed system using `C4.5` surpasses the others.



(a) Precision



(b) Recall



(c) F_1 measure

FIGURE 8. Comparative performance curves on scientific papers

TABLE 6. Comparison of proposed system with KEA on Scientific papers

Cut-off Point	TP	FP	FN	π	ρ	F_1
KEA						
Top-3	08	22	42	0.27	0.16	0.20
Top-5	11	29	39	0.22	0.22	0.22
Top-7	11	59	39	0.16	0.22	0.18
Top-9	16	74	34	0.18	0.32	0.23
Proposed System$_{\texttt{NB}}$						
Top-3	11	19	39	0.37	0.22	0.28
Top-5	12	38	38	0.24	0.24	0.24
Top-7	14	56	36	0.20	0.28	0.23
Top-9	15	75	35	0.17	0.30	0.21
Proposed System$_{\texttt{C4.5}}$						
Top-3	12	18	38	0.40	0.24	**0.30**
Top-5	13	37	37	0.26	0.26	**0.26**
Top-7	15	55	35	0.21	0.30	**0.25**
Top-9	18	72	32	0.20	0.36	**0.26**

6. **Conclusion and Future Work.** In this paper, we have proposed a light-weight machine learning approach to mine keyphrases from semi-structured as well as unstructured text documents. Instead of applying full or partial parsing of text documents for PoS tag patterning, which is generally not feasible for complex sentences, our method applies n-gram technique for candidate phrase generation and refines them using a set of heuristic rules. The soundness of a keyphrase extraction algorithm lies very much on the quality of its features used to gather information about candidates and their formulations. We have identified a rich set of features, including few novel and prominent ones sorting out weaknesses of KEA, and also formulated them in such a way to produce the best results. In addition, we have brought forth the concern of efficiency in terms of time complexity for real time systems. The experimental results in Section 5 demonstrate its wide applicability. Unlike KEA++, being free from utilizing any domain knowledge (except its domain-dependency on the training set), our approach is very much feasible to work in real-time environments irrespective of their domain.

An important future prospect of this task could be to capture formal structural relationships (e.g., synonyms, antonyms and holonyms) among different words and derive some relevant information to judge the importance of a phrase. WordNet is a good source for mining these relationships. Moreover, since manually selected keyphrases (assumed as gold standards) vary from person to person and also from time to time for the same person, evaluation has become a challenging issue. For some person the extracted phrases may seem to be quite convincing, whereas for others they may not be acceptable. Improving the evaluation scheme is another important future prospect for this task.

Acknowledgment. The authors would like to thank King Abdulaziz City for Science and Technology (KACST) and King Saud University for their support. This work has been funded by KACST under the NPST project number 11-INF1594-02.

REFERENCES

[1] M. Abulaish and T. Anwar, A web content mining approach for tag cloud generation, *Proc. of the 13th International Conference on Information Integration and Web Based Applications and Services*, Ho Chi Minh City, Vietnam, 2011.

[2] A. Arampatzis, T. van der Weide, C. Koster and P. van Bommel, An evaluation of linguistically motivated indexing schemes, *Proc. of the 22nd BCS-IRSG Colloquium on IR Research*, 2000.

[3] E. D'Avanzo and B. Magnini, A keyphrase-based approach to summarization: The lake system at DUC-2005, *Proc. of the 5th Document Understanding Conference*, 2005.

[4] S. Banerjee and T. Pedersen, Extended gloss overlaps as a measure of semantic relatedness, *Proc. of the 18th International Joint Conference on Artificial Intelligence*, 2003.

[5] N. Begum, M. A. Fatah and F. Ren, Automatic text summerization using support vector machine, *International Journal of Innovative Computing, Information and Control*, vol.5, no.7, pp.1987-1996, 2009.

[6] G. Berend and R. Farkas, SZTERGAK: Feature engineering for keyphrase extraction, *Proc. of the 5th International Workshop on Semantic Evaluation, ACL 2010*, pp.186-189, 2010.

[7] J. M. Conroy and D. P. O'leary, Text summarization via hidden Markov models, *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[8] J. Dougherty, R. Kohavi and M. Sahami, Supervised and unsupervised discretization of continuous features, *Proc. of the12th International Conference on Machine Learning*, 1995.

[9] D. Fensel, I. Horrocks, F. V. Harmelen, D. McGuinness and P. F. Patel-Schneider, OIL: Ontology infrastructure to enable the semantic web, *IEEE Intelligent Systems*, vol.16, no.2, pp.38-45, 2001.

[10] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin and C. G. Nevill-Manning, Domain-specific keyphrase extraction, *Proc. of the International Joint Conference on Artificial Intelligence*, pp.668-673, 1999.

[11] S. A. Golder and B. A. Huberman, Usage patterns of collaborative tagging systems, *Journal of Information Science*, vol.32, no.2, pp.198-208, 2006.

[12] C. Gutwin, G. W. Paynter, I. H. Witten, C. Nevill-Manning and E. Frank, Improving browsing in digital libraries with keyphrase indexes, *Decision Support Systems*, vol.27, no.1-2, pp.81-104, 1999.

[13] J. Han, T. Kim and J. Choi, Web document clustering by using automatic keyphrase extraction, *Proc. of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pp.56-59, 2007.

[14] P. Heymann, D. Ramage and H. Garcia-Molina, Social tag prediction, *Proc. of International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.531-538, 2008.

[15] A. Hulth, Improved automatic keyword extraction given more linguistic knowledge, *Proc. of the Conference on Emprical Methods in Natural Language Processing*, Sapporo, Japan, pp.216-223, 2003.

[16] S. N. Kim and M. Y. Kan, Re-examining automatic keyphrase extraction approaches in scientific articles, *Proc. of the ACL/IJCNLP Multiword Expressions Workshop*, 2009.

[17] S. N. Kim, O. Medelyan, M. Y. Kan and T. Baldwin, SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles, *Proc. of the SemEval-2010 Workshop at ACL-2010*, Uppasala, Sweden, 2010.

[18] B. Kosovac, D. J. Vanier and T. M. Froese, Use of keyphrase extraction software for creation of an AEC/FM thesaurus, *Electronic Journal of Information Technology in Construction*, pp.25-36, 2000.

[19] M. Litvak and M. Last, Graph-based keyword extraction for single-document summarization, *Proc. of the Workshop on Multi-Source Multilingual Information Extraction and Summarization*, pp.17-24, Manchester, UK, 2008.

[20] T. Liu, H. Zheng and F. Qi, A novel video key-frame extraction algorithm based on perceived motion energy model, *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.10, pp.1006-1013, 2003.

[21] P. Lopez and R. Romary, HUMB: Automatic key term extraction from scientific articles in GROBID, *Proc. of the 5th International Workshop on Semantic Evaluation*, 2010.

[22] O. Medelyan and I. H. Witten, Thesaurus based automatic keyphrase indexing, *Proc. of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp.296-297, 2006.

[23] O. Medelyan, I. H. Witten and D. Milne, Topic indexing with Wikipedia, *Proc. of the 1st AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08)*, Chicago, USA, pp.19-24, 2008.

[24] O. Medelyan, E. Frank and I. H. Witten, Human-competitive tagging using automatic keyphrase extraction, *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing, ACL*, Singapore, pp.1318-1327, 2009.
[25] G. Mishne, Autotag: A collaborative approach to automated tag assignment for weblog posts, *Proc. of the 15th International Conference on World Wide Web*, pp.953-954, 2006.
[26] A. Nenkova, Automatic text summarization of newswire: Lessons learned from the document understanding conference, *Proc. of the AAAI*, PA, USA, pp.1436-1441, 2006.
[27] T. D. Nguyen and M. Y. Kan, Keyphrase extraction in scientific publications, *Proc. of the International Conference on Asian Digital Libraries*, pp.317-326, 2007.
[28] S. Patwardhan and T. Pederson, Using wordNet based context vectors to estimate the semantic relatedness of concepts, *Proc. of the EACL 2006 Workshop Making Sense of Sense – Bringing Computational Linguistics and Psycholinguistics Together*, Trento, Italy, pp.1-8, 2006.
[29] P. Rychly and A. Kilgarriff, An efficient algorithm for building a distributional thesaurus (and other sketch engine developments), *Proc. of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Prague, Czech Republic, pp.41-44, 2007.
[30] G. Salton, *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.
[31] K. Sarkar, Automatic keyphrase extraction from medical documents, *Proc. of the 3rd International Conference on Pattern Recognition and Machine Intelligence*, 2009.
[32] M. Song, I. Y. Song and T. Hu, KPSpotter: A flexible information gain-based keyphrase extraction system, *Proc. of the 5th International Workshop on Web Information and Data Management*, 2003.
[33] M. Song, I. Y. Song, R. B. Allen and Z. Obradovic, Keyphrase extraction-based query expansion in digital libraries, *Proc. of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, ACM*, pp.202-209, 2006.
[34] S. Sood, K. Hammond, S. Owsley and L. Birnbaum, TagAssist: Automatic tag suggestion for blog posts, *Proc. of the International Conference on Weblogs and Social Media*, 2007.
[35] P. D. Turney, Extraction of keyphrases from text: Evaluation of four algorithms, *Technical Report ERB-1051. (NRC #41550)*, National Research Council, Institute for Information Technology, 1997.
[36] P. D. Turney, Learning algorithms for keyphrase extraction, *Journal of Information Retrieval*, vol.2, no.4, pp.303-336, 2000.
[37] X. Wan, J. Yang and J. Xiao, Manifold-ranking based topic-focused multi-document summarization, *Proc. of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, pp.2903-2908, 2007.
[38] D. Whitely, The GENITOR algorithm and selective pressure, *Proc. of the 3rd International Conference on Genetic Algorithms*, pp.116-121, 1989.
[39] I. H. Witten, Browsing around a digital library, *Proc. of Australasian Computer Science Conference*, Auckland, New Zealand, pp.1-14, 1999.
[40] W. F. B. Wu and Q. Li, Document keyphrases as subject metadata: Incorporating document key concepts in search results, *Information Retrieval*, vol.11, no.3, pp.229-249, 2008.
[41] S. Wubben and A. van den Bosch, A semantic relatedness metric based on free link structure, *Proc. of the 8th International Conference on Computational Semantics*, Tilburg, Netherland, pp.355-358, 2009.
[42] T. Zesch, C. Muller and I. Gurevich, Using wiktionary for computing semantic relatedness, *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, pp.861-866, 2008.
[43] W. Zhang, T. Yoshida, T. B. Ho and X. Tang, Augmented mutual information for multi-word extraction, *International Journal of Innovative Computing, Information and Control*, vol.5, no.2, pp.543-554, 2009.