A Web Content Mining Approach for Tag Cloud Generation

Muhammad Abulaish
Center of Excellence in Information Assurance
King Saud University, Riyadh, Saudi Arabia
mAbulaish@ksu.edu.sa

Tarique Anwar
Center of Excellence in Information Assurance
King Saud University, Riyadh, Saudi Arabia
tAnwar.c@ksu.edu.sa

ABSTRACT

Tag cloud, also known as word cloud, are very useful for quickly perceiving the most prominent terms embedded within a text collection to determine their relative prominence. The effectiveness of tag clouds to conceptualize a text corpus is directly proportional to the quality of the keyphrases extracted from the corpus. Although, authors provide a list of about five to ten keywords in scientific publications that are used to map them into their respective domain, due to exponential growth in non-scientific documents on the World Wide Web, an automatic mechanism is sought to identify keyphrases embedded within them for tag cloud generation. In this paper, we propose a web content mining technique to extract keyphrases from web documents for tag cloud generation. Instead of using partial or full parsing, the proposed method applies *n*-gram technique followed by various heuristics-based refinements to identify a set of lexical and semantic features from text documents. We propose a rich set of domain-independent features to model candidate keyphrases very effectively for establishing their keyphraseness using classification models. We also propose a font-determination function to determine the relative font-size of keyphrases for tag cloud generation. The efficacy of the proposed method is established through experimentation. The proposed method outperforms the popular keyphrase extraction system KEA.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning – *concept learning, knowledge acquisition.*

General Terms

Algorithms

Keywords

Information Extraction; Web Content Mining, Keyphrase Extraction; Tag Cloud Generation, Feature Extraction.

1. INTRODUCTION

Due to exponential growth in textual data on the World Wide Web (WWW) a substantial amount of research efforts have been directed towards applying web content mining along with natural language processing to identify key snippets from texts and use them to generate tag cloud to conceptualize the underlying text corpora. Sinclair and Cardew-Hall [1] conducted an experiment to compare the usefulness of tag cloud and traditional search interface in terms of finding relevant information, and found that where the information-seeking task required specific information, participants preferred the search interface and conversely, where the information-seeking task was more general, participants preferred the tag cloud. Thus, we may infer that tag cloud could be a useful tool for quickly perceiving the most prominent terms embedded within a text collection to determine their relative prominence and thereby to get an overall perception about the entity represented by the collection. Since a document may contain a large number of words, considering all of them for tag cloud generation would not be an effective solution, rather an automatic technique to identify only representative keyphrases is sought.

Keyphrases that generally contain one or more words (a.k.a. keywords) provide a brief summary of a document's contents. Due to existence of large documents collection in the form of digital libraries, text databases, and textual contents on the WWW the value of such summary information increases. Besides tag cloud generation, keyphrases are useful for various applications such as *document retrieval* [3], *document summarization* [2,10], *thesaurus construction* [8], and *document categorization and clustering* [4,7]. Keyphrases are also very useful for digital libraries and Web search engines. In digital libraries, the keyphrases of a scientific paper can help users to get a rough sense of the paper [5], whereas in Web search the keyphrases of a web page can serve as metadata for indexing and retrieving web pages for user supplied queries [6]. In addition, keyphrases can help users to comprehend the content of a collection without navigating through pile of documents. Keyphrases are also helpful to expand user queries, facilitate document skimming by visually emphasizing important phrases; and offer a powerful mean of measuring document similarity that can be exploited to group them into different categories.

Keyphrases are usually chosen manually, which is a labor-intensive task. In scientific publications, generally authors assign keyphrases to documents they have written, whereas professional indexers often choose phrases from a *predefined controlled vocabulary* relevant to the domain at hand. Since, many documents do not have manually assigned keyphrases, the development of a tool to automatically assign keyphrases to the documents would have a potential use. Moreover, the identification and extraction of keyphrases may be useful for inferring new facts and indexing text corpora for efficient query processing over text documents.

In this paper, we have proposed a web content mining system based on supervised machine learning approach to identify feasible keyphrases from text documents for tag cloud generation. Instead of applying full or partial parsing on text documents, which is an inefficient process for longer sentences, the proposed method

iiWAS'11, 5–7 December, 2011, Ho Chi Minh City, Veitnam.

applies *n*-gram technique to generate candidate keyphrases and refines them using a set of heuristic rules. For each candidate phrase, various lexical and semantic features are identified from text documents and used to build binary classification models for classifying candidate keyphrases of a new document as a keyphrase or as a non-keyphrase. Identified keyphrases are ranked according to their probabilistic values generated by the classification models which facilitates the users to select top few keyphrases (if the whole list is very large) for tag cloud generation. We have also defined a font-size determination function to map the weights of the selected keyphrases to their respective font size for displaying tag cloud. The novelty of the proposed method lies in its keyphrase extraction process and font-size determination function. Some of the distinguishing features of the proposed method can be summarized as follows:

- Instead of using parsing techniques, which is an inefficient process for longer sentences and generally misleads in presence of noise, the proposed method uses *n*-gram technique to identify candidate keyphrases.
- The proposed method does not use any domain-dependant feature and consequently it can be applied on texts pertaining to any domain for keyphrase identification.
- The proposed method outperforms the well-known keyphrases extraction algorithm – KEA.
- The proposed font-size generation function accepts *min* and *max* font sizes from the user and generates a relative distribution of font sizes for the selected keyphrases over this range.

The remaining part of the paper is structured as follows. The following section presents a brief review of the existing methods for keyphrase extraction from text documents. In section 3, we present the functional detail of the proposed method. The performance evaluation of the proposed system is presented in section 4. Finally, section 5 concludes the paper with possible enhancements of the proposed method.

2. RELATED WORK

In this section, we present an overview of the recent research efforts that have been directed towards the problem of keyphrase extraction and its applications to various domains including tag cloud generation. In most of the cases, keyphrase extraction is viewed as a supervised machine learning task where the training examples are text documents with manually tagged keyphrases. Turney [12] proposed a keyphrase extractor, called GenEx, which design is based on a set of parameterized heuristic rules that are fine-tuned to the training corpus using a genetic algorithm. On comparison of GenEx with the standard machine learning technique called *bagging* which uses a bag of decision trees for keyphrase extraction, it is found that GexEx performs better than the bagging procedure. Turney [11] also proposed an approach; called "Extractor", which uses a set of nine features including *phrase position*, *phrase frequency*, *phrase length*, etc. to score a candidate phrase. The feature set is calculated after stemming the phrases using Lovins stemmer.

Frank *et al.* [13] proposed a keyphrase extraction system, KEA[1], which uses naïve Bayes learning method to induce a probabilistic model from the training corpus. A classification model is learned on a set of training documents with exemplar keyphrases for the purpose of identifying keyphrases from new documents. Besides other features, KEA uses two important features such as position of the first appearance of phrases and their *tf-idf*. Here *tf* corresponds to the frequency of a phrase into the document and *idf* is estimated as the ratio of the number of documents containing the particular phrase and the total number of documents in the training corpus. In both systems, KEA and Extractor, candidate keyphrases are identified by splitting text documents into small chunks of maximum three words based on the presence of phrase boundaries like *punctuations*, *brackets*, etc. The phrases beginning or ending with a stop-word are filtered out from the candidate list. Frank *et al.* [13] compared the performance of the KEA with GenEx and found that performance of KEA is comparable to it. Moreover, they claim that training naïve Bayes learning technique is quicker than training GenEx, which employs a special purpose genetic algorithm for training. Due to simplicity and decent accuracy, KEA is more popular and used in different applications. For example, the Keyphind system proposed by Gutwin *et al.* [9] to support keyphrase-based search uses KEA in background to identify keyphrases in text documents.

Medelyan and Witten [16] proposed KEA++, an improved version of KEA, in which they defined four features including two already defined in KEA – *tf-idf* score, position of the first occurrence of a phrase, length of a phrase in words, and node degree. The node degree represents the number of thesaurus links that connect the term to other candidate phrases. Like KEA, a supervised learning method using naïve Bayes classifier is used to learn the KEA++ model. The reported precision, recall, and f-measure values are 28.3%, 26.1% and 25.2% respectively.

Medelyan *et al.* [17] presented another method with a slightly rich set of features for topic indexing with Wikipedia. Firstly, for the purpose of candidate selection, they devised a formula to score Wikipedia articles corresponding to each *n*-gram from the collection and then selected the highest scoring article to represent as index term for the corresponding *n*-gram. The collection of index terms is regarded as the set of candidate phrases. Finally, they defined five features, out of which first three are the same as those defined in KEA++, the fourth one uses Wikipedia but same as the last feature in KEA++, and the remaining one, a novel one using Wikipedia. In this work too, they used naïve Bayes classifier for classification. On evaluation, the average f-measure value defined in terms of consistency is found to be 30.5%.

Medelyan *et al.* [18] also presented an algorithm "Maui" in which they used nine features including five lexical features and four Wikipedia based features. On experimentation, they found that when considering all these nine features for classification the bagged decision tree yields better result than naïve Bayes. But, at the same time, when considering the first three lexical features only, naïve Bayes classifier performs better than bagged decision tree classifier.

A number of research efforts have also been diverted towards exploiting keyphrases or key terms to generate tag clouds. In [25], Kuo *et al.* proposed a tool PubCloud to summarize results returned by search engines in the form of tag clouds and to allow the navigation from tag clouds to the results. Terms to act as tags in the cloud are extracted from the contents by applying simple information retrieval techniques including stop-word removal and relative phrase frequency values. This tool is developed for use with PubMed database, which stores biomedical literatures and it is well tested on it. Song *et al.* [26] proposed a graph-based clustering technique to predict tags for recommendation by learning from existing tagged documents. In CourseCloud [24], Koutrika *et al.* used clouds to summarize results of keyword searches over structured data in the form of a course database and to guide users

[1] http://www.nzdl.org/Kea/

to refine their searches. In [23], Schrammel *et al.* did a lot of analysis for proper arrangement of tags in the cloud to make it effective for search tasks and also to attract the attention of a user towards the most relevant terms. They found semantically clustered tag clouds to be the most suitable arrangement. Kaser and Lemire [22] worked for the effective visualization of clouds using algorithmic approaches to improve the display of tag clouds.

Compared to the previous works, our approach uses a rich set of domain-independent features for automatic keyphrase extraction. In addition to *tf-idf*, phrase position, etc., we introduce few novel statistical features, e.g., semantic relatedness of individual words of a candidate phrase, positional weight, and so on. We also present a novel application of the extracted keyphrases and their learned weight to generate tag cloud to conceptualize underlying text documents collection.

3. PROPOSED WEB CONTENT MINING METHOD

In this section, we present the functional detail of the proposed web content mining method for keyphrase extraction and tag cloud generation, which follows an algorithmic approach to identify keyphrases in text documents. The proposed method identifies keyphrases and generates tag cloud using following five ordered subtasks – *document pre-processing*, *candidate phrase identification*, *feature extraction*, *model learning and keyphrase identification*, and *tag cloud generation*. Further detail about each of these subtasks is presented in the following sub-sections.

3.1 Document Pre-processing

This section presents the document pre-processing process, which is a first step towards identifying keyphrases in text documents. This process takes text documents as input and generates *n*-grams after proper tokenization of text documents. The tokenization and *n*-gram generation processes are explained in the following sub-sections.

3.1.1 Tokenization

Tokenization is a process to decompose texts into small size chunks. The tokenizer is implemented to work on different file formats including portable document format (pdf). Since we are concerned only with textual contents, all the images and their labels are excluded while converting pdf documents into text documents. Similarly, for web documents, the HTML tags are filtered out before further processing. Thereafter, the tokenizer divide the text into record-size chunks which boundaries are decided using the appearance of punctuations like comma, semicolon, full stop, inverted comma, opening or closing brackets, exclamation mark, question mark and so on. The chunks so generated are then used to generate *n*-grams.

3.1.2 N-gram Generation

An *n*-gram can be defined as a sequence of *n* consecutive words from text documents. Depending on the size of window, it can be a 1-gram containing single word if the window-size is one, 2-gram containing two consecutive words if the window-size is two, and so on. On analysis, we observed that in rare cases the manually assigned keywords (as in case of scientific papers) to a document consist of more than three words. In most of the cases it is a double word phrase, and in other cases it is either a single word or a triple word phrase. Hence, in our method the value of *n* is constrained to 3, i.e., we generate all possible 1-, 2-, and 3-grams from the chunks output by the tokenizer. While generating *n*-grams, the position of first occurrence, the position of last occurrence, and the frequency of each *n*-gram is captured and stored with it in a structured format. Some of the 1-, 2-, and 3-grams along with their first occurrence position (*fop*), last occurrence position (*lop*), and frequency counts (*freq*), generated from the following sentence is shown in table 1.

"populations in developing countries are growing so quickly that the land and water are unable to sustain them"

Table 1. N-grams along with their first and last occurrence positions and frequency counts

N-grams	fop	lop	freq
1-grams			
Populations	0	0	1
In	1	1	1
Developing	2	2	1
Countries	3	3	1
Are	4	13	2
2-grams			
populations in	0	0	1
in developing	1	1	1
developing countries	2	2	1
countries are	3	3	1
3-grams			
populations in developing	0	0	1
in developing countries	1	1	1
developing countries are	2	2	1

Table 2. Candidate phrases obtained after applying filtering rules on the *n*-grams of table 1

N-grams	fop	lop	freq
1-grams			
Populations	0	0	1
Developing	2	2	1
Countries	3	3	1
2-grams			
developing countries	2	2	1
3-grams			
populations in developing	0	0	1

3.2 Candidate Phrase Identification

In this phase, all possible *n*-grams (hereafter, we say phrases) are analyzed and cleaned to determine the set of candidate phrases. This is implemented as a two-step process – i) *phrase processing*, and ii) *phrase filtering*. During processing phase, in addition to removal of apostrophes, the words in a phrase starting or ending with numeric values are processed to drop the numeric constituents, whereas during filtering phase a set of heuristic rules is applied to filter-out phrases that are obviously not a candidate for keyphrase. Some of the heuristic rules used by the candidate phrase identification module are as follows:

- Filter out a single-word phrase which is a member of the set of stop-words.
- Filter out a multi-word phrase starting or ending with a stop-word. In order to reduce redundancy, we have opted to drop whole phrase instead of removing the stop-words from them (i.e., phrase cleaning). For example, if we clean the 2-gram "*populations in*" of table 1 by removing the stop-word "*in*", we will get the phrase "*populations*", which is already an 1-gram in that table. So, we have opted for dropping instead of cleaning.

- Drop the phrases containing special characters like /, \\, %, etc. as they can never be a keyphrase. Like previous rule, we have opted phrase removal instead of phrase cleaning to reduce redundancy.
- Drop the phrases consisting only characters other than English letters (e.g., 93%), as they are meaningless individually.
- Filter out a phrase starting or ending with non-acronym words containing three or less letters as such words are generally not used as a keyphrase.

All the phrases retained after applying the above-mentioned filtering rules are compiled as a list of candidate phrases. After applying the above-mentioned rules on the phrases, presented in table 1, the list of candidate phrases is shown in table 2. For each candidate phrase a list of feature set, as explained in the following sub-section, is generated which is then used to establish its keyphraseness using classification systems.

3.3 Feature Extraction

The input to this component is the list of candidate phrases generated during candidate phrase identification process. In order to characterize a candidate phrase, we have identified a set of nine lexical and statistical features. Here, we present a brief detail about all the features identified to characterize candidate phrases.

The feature extraction process takes the list of candidate phrases as input and outputs feature vectors for them. Since, most of the features are defined as a function of phrase frequency, we have applied stemming to remove the suffix from a phrase and consider all variations to increase its frequency count. For stemming, we have used the Porter stemmer [14], which uses heuristic rules to remove or transform English suffixes. Besides stemming, it also applies case-folding – a property by which it converts the whole phrase into the lower-case letters. In this way, while calculating features values, two different phrase with same stem and in whatever case (upper-case or lower-case) they are, considered as the same phrase.

Relative Phrase Frequency (F_{rel}): We assume that the importance of a phrase is directly proportional to its frequency count. So, we count the number of times a phrase occurs in a document and then normalize the count by dividing it with the maximum frequency of any candidate phrase and use the value as a feature termed as *relative phrase frequency*. The relative phrase frequency of a phrase p is calculated using equation 1 in which f represents the frequency count of p and $max\{f_i\}$ is the maximum value among the frequencies of all the phrases.

$$F_{rel}(p) = \frac{f}{max\,\{f_i\}} \tag{1}$$

Cumulative Weight (W_{cum}): If a phrase frequency is not very high, but frequency of the individual words of the phrase is very high, we consider this phrase as an important one. So, based on the frequency of the individual words of a phrase p, we calculate its cumulative weight using equation 2 in which $tf(t_i)$ represents the frequency count of term t_i and $length(p)$ is the number of words in phrase p.

$$W_{cum}(p) = log_2\left(1 + \sum_{i=1}^{length(p)} tf(t_i)\right) \tag{2}$$

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 (more weight if occurrence is at either end and lower weight for middle) is assigned to a phrase to reflect its positional importance. The positional weight of a phrase p is calculated using equation 3 where, pos is the position of first occurrence of the phrase and s is the size of the document in terms of words.

$$W_{pos} = \begin{cases} \left|1 - \dfrac{pos}{\left(\frac{s}{2}\right)}\right|, if\ pos \neq \dfrac{s}{2} \\ \left|1 - \dfrac{pos}{\left(\frac{s}{2}\right)}\right| + \dfrac{1}{\left(\frac{s}{2}\right)}, if\ pos = \dfrac{s}{2} \end{cases} \tag{3}$$

Length (L): A phrase with a larger length in terms of words is considered to be more important than a phrase with smaller length provided their frequency count is same. So, we have taken the length, L, of a phrase as a feature which is calculated by counting the number of words in it.

Relatedness: This feature is used to reflect the relatedness among the individual consecutive words of a multiword phrase. For a single-word phrase, the value of relatedness feature, $R(w)$, is calculated using equation 4 in which $f(w)$ represents the frequency of w and s is the size of documents. For a phrase containing two words w_1 and w_2 the relatedness feature, $R(w_1, w_2)$ is calculated using equation 5, where $P(w_1)$, $P(w_2)$ and $P(w_1,w_2)$ are defined by equations 7 and 8, and for a phrase containing three words w_1, w_2, and w_3, it is calculated using equation 6, where $P(w_1)$, $P(w_2)$, $P(w_3)$, $P(w_1,w_2)$ and $P(w_1,w_2,w_3)$ are defined by equations 7, 8 and 9. In equations 7, 8 and 9, $f(w_1)$, $f(w_1,w_2)$ and $f(w_1,w_2,w_3)$ are the frequencies of co-occurrences of the words inside the parentheses.

$$R(w) = log_2\left(1 + \frac{f(w)}{s}\right) \tag{4}$$

$$R(w_1, w_2) = log_2\left(1 + \frac{P(w_1, w_2)}{P(w_1)P(w_2)}\right) \tag{5}$$

$$\begin{aligned} R(w_1, w_2, w_3) = log_2\left(1 + \frac{P(w_1, w_2)}{P(w_1)P(w_2)}\right) \\ + \left(1 + \frac{P(w_2, w_3)}{P(w_2)P(w_3)}\right) \\ + log_2\left(1 + \frac{P(w_1, w_2, w_3)}{P(w_1)P(w_2)P(w_3)}\right) \end{aligned} \tag{6}$$

$$P(w_1) = \frac{f(w_1)}{s} \tag{7}$$

$$P(w_1, w_2) = \frac{f(w_1, w_3)}{s} \tag{8}$$

$$P(w_1, w_2, w_3) = \frac{f(w_1, w_3, w_3)}{s} \tag{9}$$

Capitalization: Any phrase occurring in a text document as its first letter in uppercase is considered as an important word. So, we have devised a formula to assign a capitalization weight, *Cap*, to each phrase with this property, which is calculated by equation 10, where *N(U$_w$)* is the number of words with its first letter in upper case, *L* is the length of the phrase and *f* is the frequency of the phrase in the document.

$$W_{cap} = \frac{\sum \frac{N(U_w)}{L}}{f} \tag{10}$$

TF-IDF: It combines the frequency of a phrase in a particular document with its inverse document frequency. This score is high for rare phrases that appear frequently in a document and therefore are more likely to be significant [15]. It is calculated using equation 11, where N_d is the number of documents in the corpus in which the phrase exists and *C* is the total number of documents in the corpus.

$$TF\text{-}IDF = \frac{f}{s} \times \left(- log_2 \frac{N_d}{C} \right) \tag{11}$$

Lifespan: It determines the extent of a phrase in a document. A phrase occurring either at the beginning or at the end position will get more score as compared to those that occur far from the beginning and end. It is defined by equation 12 in which P_f and P_l represent the positions of the first and last occurrences of the phrase P respectively.

$$LS = \frac{P_l - P_f}{s} \tag{12}$$

Keyphraseness: 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 has a good chance of being a keyphrase in the document. So, we give more score to this phrase as compared to those not existing in the keyword set of training documents. It is defined by equation 13, where *N$_{PM}$(G,kw)* is the number of perfect matches in 1, 2 & 3-grams of candidate phrases with the keywords, and N_m is the number of matches occurred.

$$K = \frac{\sum N_{PM}\,(G, kw)}{N_m} \tag{13}$$

3.4 Model Learning and Keyphrase Identification

In the previous section, we have discussed various features identified for each candidate phrase to learn classification systems. In this section, we discuss the classification models used to identify keyphrases from text documents. From training dataset, we have extracted the previously mentioned features using *n*-gram and statistical techniques and then trained four different classification models namely *naïve Bayes*, *Decision Tree* (*C4.5*), *Multi-Layer Perceptron (MLP)*, and *RIPPER* to establish the effectiveness of the proposed feature set to discriminate keyphrases from non-keyphrases. From classification results, we calculate true positives TP (number of actual keyphrases classified as keyphrase), false negatives FN (number of actual keyphrases classified as non-keyphrase), false positives FP (number of non-keyphrases classified as keyphrase), and true negatives TN (number of non-keyphrases classified as non-keyphrase). By using these values, we calculate the standard performance measure "accuracy" as defined in equation 14. We have applied 10-fold cross validation for evaluating the performance of the classifiers, i.e., the dataset is divided into 10 smaller subsets, out of which 9 subsets are used for training and one subset is used for testing. This process is repeated 10 times. The accuracy values of the different classifiers are shown in table 3. It can be observed from the results presented in table 3 that the accuracy of the C4.5 and MLP is highest, and the accuracy of the RIPPER is the next highest, but in terms of true positive values MLP is lowest and C4.5 is highest. Although, the accuracy of the naïve Bayes classifier is lowest, the difference is not very significant, but on the other hand it achieved highest true positive value. Hence, the C4.5 and Naïve Bayes are the obvious choice to use for the classification purpose.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FN + FP + TN}} \tag{14}$$

Table 3. Classification accuracy values for proposed feature set

Classifier	TP	FN	FP	TN	Accuracy (%)
Naïve Bayes	17	625	24	78454	99.180
C4.5	8	8	33	79071	99.948
MLP	0	0	41	79079	99.948
RIPPER	6	7	35	79072	99.947

3.5 Tag Cloud Generation

Searching for relevant information from vast collections of digital data is a common activity and the problem of retrieving this critical information was found long ago [20]. However, the rapid growth of internet has enlarged its domain from digital libraries to the Web itself [21], and still a lot of efforts are being made to retrieve a precise as well as informative collection. One way to present a voluminous textual data retrieved for a search query is by using *tag clouds* [24, 25]. A tag cloud can be said as a collection of main topical terms mined from the voluminous contents and presented pictorially as a cloud of terms emphasizing them according to their relevance. It enables the reader to identify context of the retrieved text data and quickly determine if it is of interest or not. The most general way to generate tags for the content is to simply rank the single words in terms of their frequency after filtering out the common stop-words. However, in our approach we have applied a machine learning scheme to mine topical terms intelligently, which are presented in the form of a cloud of tags with varying sizes highlighting the most relevant ones. The complete process is summarized as below:

- For each candidate phrase, a set of nine feature values are calculated from text contents.
- The trained classification model is applied to determine the keyphraseness of the candidate phrases.
- Positively classified keyphrases are ranked primarily according to their probability of being positive and secondarily according to their *tf-idf* values to break the ties.

- Top n (user given parameter) phrases p_i, $i = 1, 2, \ldots, n$ are declared as tags and for each p_i the corresponding *tf-idf* value is considered as weight $\omega(p_i)$
- For each tag p_i, the weight $\omega(p_i)$ is used to calculate its font size using equation 15 in which F_{max} and F_{min} represent the maximum and minimum font size respectively supplied by the user.
- Finally, tags are displayed with their sizes as calculated, positioning the larger ones towards the cloud center and moving outwards gradually with comparatively smaller sizes.

Table 4. Top 20 keyphrases (tags), their weights and font-sizes

Tags (p_i)	***W(P$_i$)***	***F(P$_i$)***	**Tags (p_i)**	***W(P$_i$)***	***F(P$_i$)***
Resource management	0.00098	13	Farming systems	0.00465	42
Upland areas	0.00099	13	Systems	0.00000	5
Southeast Asia	0.00069	10	Forest	0.00103	13
Information	0.00000	5	Trees	0.00190	20
Information kit	0.00026	7	Soil conservation	0.00059	10
Organized	0.00000	5	Conservation	0.00138	16
Agroforestry	0.00541	48	Participants	0.00000	5
Agricultural	0.00000	5	Plant	0.00146	17
Development	0.00000	5	Farm household	0.00151	17
Farming	0.00000	5	Upland	0.00017	6

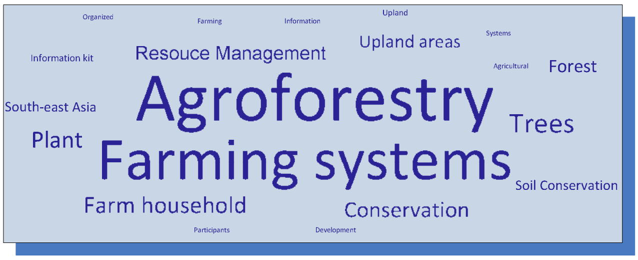

Figure 1. A sample tag cloud visualization of the tags shown in table 4

We have performed our experiment on a set of large-sized agricultural documents consisting of 20 documents for learning the model and 5 other documents for testing our results. After extracting the most promising keyphrases from test documents they are weighted by their *tf-idf* values and font sizes are calculated using equation 15. Table 4 shows top 20 keyphrases considered as tags in the cloud along with their *tf-idf* values and calculated font sizes using maximum and minimum font size as 48 and 5, respectively. Figure 1 shows the generated tag cloud. On observing this cloud, we can see that the terms with larger sizes are brought into notice earlier than those that are comparatively smaller. This effect makes us realize the relevance of these highlighted terms in the context, by going through which we can easily chose to move inside the document for more details or switch on to other as per our requirement. Moreover, the size of this cloud can be enlarged further to accommodate more terms in it to have a broader review of the content. It can be set proportional to the depth of review we need to have.

$$F(p_i) = (F_{max} - F_{min}) \times \left(\frac{\omega(p_i) - \min\{\omega(p_j)\}}{\max\{\omega(p_j)\} - \min\{\omega(p_j)\}} \right) + F_{min} \qquad (15)$$

4. PERFORMANCE EVALUATION

In this section, we present the comparative performance evaluation results of the proposed keyphrase extraction method. We have compared the efficacy and correctness of the proposed method with KEA proposed by Frank *et al.* [13]. For evaluation of the experimental results, we have used standard information retrieval metrics – *precision*, *recall*, and *f-measure*. Precision (π) is defined as the ratio of true positives among all retrieved instances; recall (ρ) is defined as the ratio of true positives among all positive instances; f-measure, also called f_1-measure (F_1) is the harmonic mean of recall and precision. From the classification results, we calculate the true positives (TP), the false positive (FP), and false negatives (FN) and by using these values we calculate the values of precision, recall and f_1-measure using equations 16, 17, and 18 respectively.

$$precision\ (\pi) = \frac{TP}{TP + FP} \qquad (16)$$

$$recall\ (\rho) = \frac{TP}{TP + FN} \qquad (17)$$

$$f\text{-}measure\ (F_1) = 2 \times \frac{\pi \times \rho}{\pi + \rho} \qquad (18)$$

For evaluation, we have used an agricultural dataset which has been also used by KEA for training and testing. On training set, we have applied our method to identify candidate keyphrases and generate feature vectors for each of them. A feature vector consists of one value for each of the features discussed earlier in this paper. Once, the feature vector is generated for all candidate phrases, we use the naïve Bayes classifier, implemented as a part of WEKA[2] (Waikato Environment for Knowledge Analysis) machine learning software to learn the classification model. We have applied 10-fold cross-validation to test the learned model.

The trained classification model is applied on the KEA test documents to extract keyphrases from them. The extracted keyphrases are arranged in decreasing order of their class probability values generated by the classification model in which the topmost phrases are the most promising keyphrases for the document. We have calculated the values of precision, recall and f_1-measure on the test documents and compared the results with KEA. For each document in this set, the list of author-assigned keywords is used to calculate the values of true positives, false positives, and false negatives. An extracted keyphrase which is a member of the list of author-assigned keywords is considered as true positive, and the one which is not a member is considered as false positive. Similarly, a phrase in the list of author-assigned keywords which is not recognized by the system as a keyphrase contributes for false negatives. Since the number of author-

[2] http://www.cs.waikato.ac.nz/ml/weka/

assigned keywords (generally available for scientific publications) are not same for all documents we have defined different cut-offs to consider that many phrases from the top of the list of extracted keyphrases. On analysis, we found that generally a document contains less than 10 author-supplied keywords. Hence, the cut-off values are taken as 3, 5, 7, and 9 and for each of them we have calculated the values of precision (π), recall (ρ), and f$_1$-measure (F$_1$).

Table 5. Comparison result of the proposed method with KEA

Cut-off Point	Proposed Method				KEA			
	TP	FP	FN	π/ρ/F$_1$	TP	FP	FN	π/ρ/F$_1$
Document – 1								
Top-3	1	2	8	0.33/0.11/0.17	1	2	8	0.33/0.11/0.17
Top-5	3	2	6	0.6/0.33/0.43	1	4	8	0.20/0.11/0.14
Top-7	3	4	6	0.43/0.33/0.37	1	6	8	0.14/0.11/0.12
Top-9	3	6	6	0.33/0.33/0.33	1	8	8	0.11/0.11/0.11
Document – 2								
Top-3	2	1	8	0.66/0.20/0.31	1	2	9	0.33/0.10/0.15
Top-5	3	2	7	0.60/0.30/0.40	2	3	8	0.40/0.20/0.27
Top-7	3	4	7	0.43/0.30/0.35	2	5	8	0.29/0.20/0.24
Top-9	3	6	7	0.33/0.30/0.31	2	7	8	0.22/0.20/0.21
Document – 3								
Top-3	2	1	2	0.67/0.50/0.57	2	1	2	0.67/0.50/0.57
Top-5	2	3	2	0.40/0.50/0.44	2	3	2	0.40/0.50/0.44
Top-7	2	5	2	0.29/0.50/0.37	2	5	2	0.29/0.50/0.37
Top-9	2	7	2	0.22/0.50/0.31	2	7	2	0.22/0.50/0.31
Document – 4								
Top-3	1	2	6	0.33/0.14/0.20	1	2	6	0.33/0.14/0.20
Top-5	2	3	5	0.40/0.29/0.34	1	4	6	0.20/0.14/0.16
Top-7	2	5	5	0.29/0.29/0.29	1	6	6	0.14/0.14/0.14
Top-9	3	6	4	0.33/0.49/0.39	1	8	6	0.11/0.14/0.12
Document – 5								
Top-3	1	2	16	0.33/0.06/0.10	1	2	16	0.33/0.06/0.10
Top-5	1	4	16	0.20/0.06/0.09	1	4	16	0.20/0.06/0.09
Top-7	1	6	16	0.14/0.06/0.08	3	4	14	0.43/0.18/0.25
Top-9	2	7	15	0.22/0.11/0.15	5	4	12	0.56/0.29/0.38
Macro-Average								
Top3	7	8	40	0.47/0.15/0.23	6	9	41	0.40/0.13/0.20
Top5	11	14	36	0.44/0.23/0.30	7	18	40	0.28/0.15/0.20
Top7	11	24	36	0.31/0.23/0.26	9	26	38	0.26/0.19/0.22
Top9	13	32	34	0.29/0.28/0.28	11	34	36	0.24/0.23/0.23

The comparison result of our method with KEA is presented in table 5. Figures 2, 3, and 4 present a visual perception of the results shown in table 5. It can be seen in table 5 that for all cut-offs the average precision, recall, and f-measure values of our method is greater than that of the KEA. Moreover, it can be observed from figures 2 and 3 that precision values are monotonically decreasing, whereas the recall is monotonically increasing. In other words, if we increase the value of *n*, where *n* is the number of top-ranked phrases considered as keyphrases, the false positives are increasing since the average number of author-assigned keywords across all the test documents are approx. 5. This is why the value of precision is decreasing with increasing value of *n*. Similarly, for increasing value of *n* the false negatives are decreasing which results in increased recall value. Both of the methods – the proposed one and the KEA, follow the same trend but for all cut-off values our method outperforms KEA.

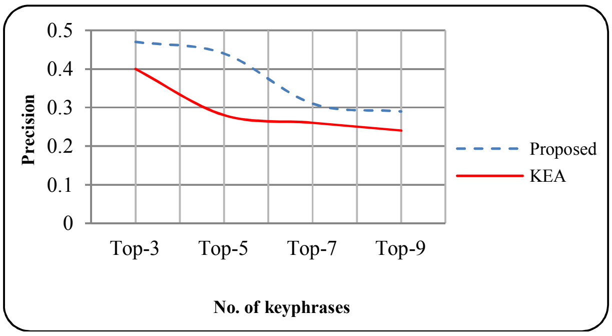

Figure 2. Precision of the proposed method and KEA

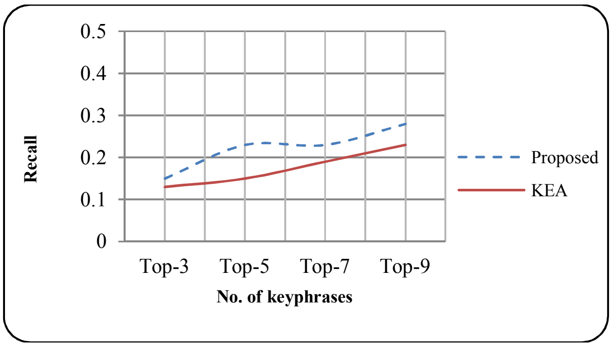

Figure 3. Recall of the proposed method and KEA

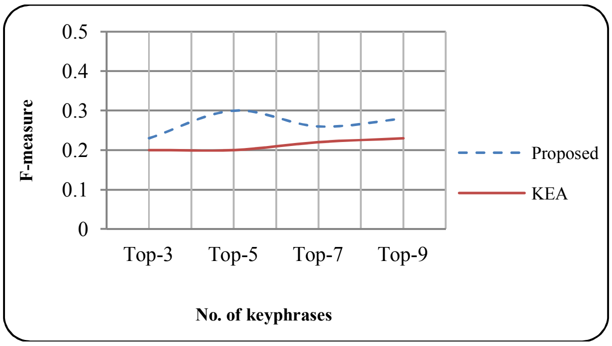

Figure 4. F-measure of the proposed method and KEA

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented a web content mining method based on supervised machine learning approach to mine keyphrases from text documents for tag cloud generation. Instead of applying full or partial parsing on text documents, 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. Thereafter, each phrase is converted into a vector of feature values generated from text documents using NLP and statistical analysis. We have also defined a font-size determination function to map the weights of keyphrases to their respective font sizes for visualization of tag cloud. Since different persons may select different set of keyphrases from same documents evaluation of the quality of extracted keyphrases is a challenging task. Therefore, our future work will be focused on evaluation of the extracted phrases with a more robust measure. Moreover, we are trying to explore some more features that can use the structural relation (synonyms, antonyms, etc.) of phrases in the text to reflect the context or environment of phrase occurrence.

6. REFERENCES

[1] Sinclair J. and Cardew-Hall, M. 2008. The folksonomy tag cloud: when is it useful? *Journal of Information Science*, 34(1), 15–29.

[2] Zha, H. 2002. Generic Summarization and Keyphrase Extraction using Mutual Reinforcement Principle and Sentence Clustering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 113–120.

[3] Jones, S. and Staveley, M. S. 1999. Phrasier: A System for Interactive Document Retrieval using Keyphrases. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 160–167.

[4] Han, J., Kim, T. and Choi, J. 2007. Web Document Clustering by using Automatic keyphrase extraction. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology,* 56–59.

[5] Gutwin, C., Paynter, G., Witten, I. H., Nevill-Manning, C. and Frank, E. 1999. Improving Browsing in Digital Libraries with Keyphrase Indexes. *Decision Support Systems,* 27(1-2), 81–104.

[6] Li, Q., Wu, Y.B., Bot, R. and Chen, X. 2004. Incorporating Document Keyphrases in Search Results. In *Proceedings of the 10th American Conference on Information Systems*, New York.

[7] Jonse, S. and Mahoui, M. 2000. Hierarchical Document Clustering using Automatically Extracted Keyphrase. In *Proceedings of the 3rd International Asian Conference on Digital Libraries*, Seoul, Korea, 113–120.

[8] Kosovac, B., Vanier, D. J. and Froese, T. M. 2000. Use of Keyphrase Extraction Software for Creation of an AEC/FM Thesaurus. *Journal of Information Technology in Construction,* 5, 25–36.

[9] Gutwin, C., Paynter, G. W., Witten, I. H., Nevill-Manning, C. G. and Frank, E. 1999. Improving Browsing in Digital Libraries with Keyphrase Indexes. *Journal of Decision Support Systems,* 27, 81–104.

[10] Kupiec, J., Pedersen, J. and Chen, F. 1995. A Trainable Document Summarizer. In *Proceedings of the SIGIR, ACM Press*, 68–73.

[11] Turney, P. D. 2000. Learning Algorithm for Keyphrase Extraction. *Journal of Information Retrieval,* 2(4), 303–36.

[12] Turney, P. D. 1999. Learning to Extract Keyphrases from Text. *National Research Council, Institute for Information Technology,* Technical Report ERB-1057.

[13] Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C. and Nevill-Manning, C. G. 1999. Domain-specific Keyphrase Extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, San Mateo, CA.

[14] Porter, M. F. 1980. An Algorithm for Suffix Stripping, Program, 14(3), 130–137.

[15] Salton, G., & McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY.

[16] Medelyan, O. and Witten, I. H. 2006. Thesaurus-Based Automatic Keyphrase Indexing, In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, New York, USA, 296–297.

[17] Medelyan, O., Witten, I. H. and Milne, D. 2008. Topic Indexing with Wikipedia. In Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, Chicago, USA. 19–24.

[18] Medelyan, O., Frank, E., and Witten, I. H. 2009. Human-Competitive Tagging using Automatic Keyphrase Extraction. In *Proceedings of the International Conference of Empirical Methods in Natural Language Processing* (EMNLP), Singapore, 1318–1327.

[19] Fensel, D., Horrocks, I., Harmelen, F. van, McGuinness, D. L. and Patel-Schneider, P. 2001. OIL: Ontology Infrastructure to Enable the Semantic Web, *IEEE Intelligent Systems,* 16(2), 38–45.

[20] Maron, M. E. and Kuhns, J. L. 1960. On Relevance, Probabilistic Indexing and Information Retrieval, *Journal of the ACM*, 7(3), 216–244.

[21] Aula, A., Jhaveri, N. and Kaki, M. 2005. Information search and re-access strategies of experienced web users, In *Proceedings of the 14th international conference on World Wide Web* (WWW'05), 583–592.

[22] Kaser, O. and Lemire, D. 2007. TagCloud Drawing: Algorithms for Cloud Visualization, In *Proceedings of the 16th International Conference on World Wide Web* (WWW'07), Canada.

[23] Schrammel, J. Littner, M. and Tscheligi, M. 2009. Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches, In *Proceedings of the 27th International Conference on Human Factors in Computing Systems,* 2037–2040.

[24] Koutrika, G., Zadeh, Z. M. and Garcia-Molina, H. 2009. Data clouds: summarizing keyword search results over structured data, In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* 391–402.

[25] Kuo, B. Y. Hentrich, T., Good, B. M. and Wilkinson, M. D. 2007. Tag clouds for summarizing web search results. In *Proceedings of the 16th International Conference on World Wide Web* (WWW'07), 1203–1204.

[26] Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W. and Giles, C. L. 2008. Realtime automatic tag recommendation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '08), Singapore, 515–522.