A Keyphrase-Based Tag Cloud Generation Framework to Conceptualize Textual Data

Muhammad Abulaish[1,2] and Tarique Anwar[1]

[1]*Center of Excellence in Information Assurance*
King Saud University, Riyadh, Saudi Arabia
E-mail IDs: abulaish@ieee.org, tarique.ar@gmail.com

[2]*Department of Computer Science, Jamia Millia Islamia (A Central University)*
Jamia Nagar, New Delhi – 110025, India

Abstract

Due to increasing accumulation of textual data on the World Wide Web, tag cloud has become an effective tool to quickly perceive the most prominent terms embedded within them. Tag clouds help to grasp the main theme of a corpus without exploring the pile of documents. However, the effectiveness of tag clouds to conceptualize text corpora is directly proportional to the quality of the tags extracted from them. In this paper, we propose a keyphrase-based tag cloud generation framework to conceptualize textual data. In contrast to existing tag cloud generation systems that use single words as tags and their frequency counts to determine the font size of the tags, the proposed framework identifies feasible keyphrases and uses them as tags. The font-size of a keyphrase is determined as a function of its relevance weight. Instead of using partial or full parsing, which is inefficient for lengthy sentences and inaccurate for the sentences that do not follow proper grammatical structure, the proposed method applies n-gram techniques followed by various heuristics-based refinements to identify candidate phrases from text documents. A rich set of lexical and semantic features are identified to characterize the candidate phrases and determine their keyphraseness and relevance weights. We also propose a font-size determination function, which utilizes the relevance weights of the keyphrases to determine their relative font size for tag cloud visualization. The efficacy of the proposed framework is established through experimentation and its comparison with the existing state-of-the-art tag cloud generation methods.

Keywords

Text mining; Web content mining, Keyphrase extraction; Tag cloud generation, Feature extraction.

INTRODUCTION

The overwhelming growth of textual data on the World Wide Web (WWW), both in scope as well as in depth, is leading to an unrestricted growth of the Cyberspace. Numerous websites are being created everyday and a number of pages are being added to them in a frequent manner. Moreover, the emergence of social media is vehemently proliferating user-generated content in the form of blogs, social networking sites, discussion boards and forums. It is greatly affecting the pace in which data are being accumulated on the Web. A trend of the blogosphere noticed by Bansal & Koudas (2007a) and Platakis et al. (2009) is that 175,000 new blogs including 1.6 million posts are being added to the WWW every day, and these figures are rapidly rising with time. The vast accumulation of textual data on the WWW is reflected by the large number of hits for a given user query. For example, the number of hits by Google for the query term "*tag cloud generation*" is about 95300, in which top 10 results were so closely related to the query that it became hard to distinguish their singularity. This huge accumulation of textual data on the Web has attracted researchers from variety of fields including data mining, information retrieval, natural language processing, and visualization. These researchers are interested in mining information components to conceptualize huge repository of text documents. However, the prime focus of all these tasks remained as summarizing textual data using various techniques ranging from keyphrase extraction (Abulaish & Anwar, 2012) to question-answering (Light et al., 2001) to sentiment analysis and opinion mining (Pang & Lee, 2008).

Comprising one or more words, keyphrases (a.k.a. keywords) of a document provide a brief summary of its content. Moreover, a quick random navigation through the keyphrases of a document leads to an overall grasp of the information contained in the document and makes the users free from the tedious task of manually exploring the complete content. Due to existence of large document collection in the form of digital libraries, text databases and textual data on the WWW, the value of such summary information has increased significantly. In addition to tag cloud generation, keyphrases are useful to various applications, such as *document indexing and retrieval* (Jones & Staveley, 1999), *document summarization* (Zha, 2002), (Kupiec et al., 1995), *thesaurus construction* (Kosovac et al., 2000), and *document categorization and clustering* (Han et al., 2007; Jhones & Mahoui, 2000). 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 (Gutwin et al., 1999), whereas in web search the keyphrases can serve as metadata to index and retrieve webpages (Li et al., 2004). In addition, keyphrases can help users to comprehend the content of a collection without navigating through the pile of documents. Keyphrases are also helpful to expand user queries, facilitate document skimming by visually emphasizing important phrases, and they offer a powerful means of measuring document similarity that can be exploited to group documents into various categories.

While a naïve user explores and navigates through the Web, a number of problems are faced to locate the exactly needed information. To overcome all hindrances in finding the required information, a new concept of *tag cloud* aroused, especially related to the Web 2.0 applications. A typical *tag cloud* is the visualization of a set of most prominent and popular tags associated with the underlying data collection in the form of a cloud (Lohmann et al., 2009). The prominent tags are usually identified as the most frequent tags, and the prominence of a tag is represented in terms of its font size to attract the attention of viewers. In case of non-textual data, these tags are the terms explicitly assigned by users that stand for their brief description as a set of terms. For example, *Flickr* provides tag clouds of the most popular tags entered by users during last 24 hours or over the last week. On the other hand, textual data tags are automatically identified as the most frequent terms (or entities) in texts (e.g., *Blogoscope*). Papapetrou *et al.* (2010) have described the advantages of tag clouds as being a very simple, intuitive and visually appealing means to serve for two important purposes. First, tag clouds aid user to get a quick overview and hints about what is contained in the underlying data, and second, they allow user to navigate through the available information by selecting an interesting term in the cloud and viewing its source documents. It has also been claimed that tag clouds conceptually resemble histograms (Kaser & Lemire, 2007). Histograms are bar charts representing frequency distribution of data items, where height of each bar corresponds to their frequency values. Similarly, a tag cloud is viable to represent frequencies of items by means of their visualization. Moreover, the added advantage is the presence of hyperlinks that are usually associated with tags. These hyperlinks are generally used to explore and identify the ties between tags.

In a broad sense, an automatic tag cloud generation technique for textual data consists of two major sub-tasks, namely *tag identification* and *tag visualization*. However, in case of non-textual data the whole process is reduced only to tag visualization because tags are usually contributed collaboratively in a folksonomy environment. Although, the purpose of tag cloud is to emphasize prominent terms from the underlying collection to represent its theme, we observed two major shortcomings that have still remained un-noticed in state-of-the-art cloud generation techniques. First, the prominence of a tag is limited to its frequency count that considers a number of frequent non-noun words having little significance to represent the context. Second shortcoming is related to the size of individual tags. State-of-the-art techniques consider only single words as tags that generally fail to represent a clear theme because of their separation from the complete phrase. For example, the term "*tag cloud generation*" is considered as three individual words, and individually all of them focalize on different themes.

In this paper, we introduce the concept of using *keyphrases* for tag cloud generation to overcome the above-mentioned shortcomings. We propose a framework as an extension of our earlier work (Abulaish & Anwar, 2011) to conceptualize text documents using tag clouds, where tags are identified as keyphrases. Depending on the availability or unavailability of training documents with annotated keyphrases, the proposed framework uses either supervised machine learning approach or feature aggregation scheme to identify keyphrases and their significance weights in text documents. Both approaches go through the common steps of *candidate phrase extraction* using *n*-gram technique and *feature extraction* to transform a candidate phrase into a numeric vector, each component representing a salient property. The next step of *feasible keyphrase identification* is followed in a supervised manner if proper training dataset is available; otherwise, feasible keyphrases are identified through aggregation of feature values. Finally, the identified feasible keyphrases along with their significance weights are used to generate tag cloud to conceptualize the underlying text documents.

The rest of the paper is organized as follows. The following section gives an overview of the existing methods related to tag cloud generation and keyphrase extraction. Thereafter, we present the architectural design of the proposed framework. The functioning detail of each module is also discussed in this section. Thereafter, we present our experimental details and evaluation results, followed by a brief discussion on the need of another tag cloud generator. Finally, we conclude the paper with possible future enhancement of the proposed framework.

RELATED WORK

In this section, we present a brief review of the existing state-of-the-art tag cloud generation methods. In addition, we also provide a review of the research directed towards keyphrase extraction from text documents and its application to various domains including tag cloud generation.

The origin of tag clouds dates back to 1976 in the experimentation of Stanley Milgram (Milgram & Jodelet, 1976), in which a collective "*mental map*" of Paris was created using different font size to show how often each place was mentioned as a landmark in the city. It gained wide-scale popularity only after the launch of Flicker in 2004, and after that most of the Web 2.0 sites started using it. Today, it is widely used in commercial websites for both user-assigned as well as automatically generated tags. These include Flickr[1], Del.icio.us[2], Technorati[3], etc., among those assigned collaboratively by users, whereas Blogscope (Bansal & Koudas, 2007b, Grapevine (Angel et al., 2007), PubCloud (Kuo et al., 2007) are among those in which tags are automatically extracted from the underlying textual data. Several websites have also started providing the functionality for users to create their own tag clouds from different textual sources (e.g., Wordle, TagCrowd, Tag Cloud Generator, and Tree Cloud).

Using tag clouds in websites for thematic presentation is greatly influenced by the collaboratively tagged non-textual data sources, such as images and videos to give a brief description in the form of text. As the availability of tags does not remain an issue in this case, not much work has been done to determine the tags for a data source, whereas a considerable number of efforts have been made to improve tag cloud layouts. In (Seifert et al., 2008), Seifert *et al.* designed an algorithm to overcome the fundamental issues of traditional layouts, which generates the layout with reduced whitespace to make it compact and sharp, and features an arbitrary convex polygons to bound the terms. Tree Cloud (Gambette & Véronis, 2009) visualizes textual data sources in a tree-structured tag cloud. It simply considers all individual words in the text as tags after filtering stop-words. The semantic proximity between the words is reflected in their placement in the tree. The popular tag cloud generator, Wordle (Viégas et al., 2009), has come up with remarkably distinctive layouts with a rich set of fonts and colors by utilizing the typography and composition to balance various aesthetic criteria. The basic tag identification and its font size determination process are same as that of the Tree Cloud and other state-of-the-art techniques. Tag Maps (Jaffe et al., 2006) generates summaries of photo collections based on geographic as well as contextual data associated with the photographic media and produces the layout in an entirely different way based on real geographical space. PubCloud (Kuo et al., 2007) acts as a mediator to query the PubMed database of biomedical literature and generates a tag cloud to summarize the search results. For a given query, it extracts the words from the retrieved abstracts and goes through stop-word filtering and stemming to compute their frequencies based on which the font size of tags are determined in the cloud. As another aspect, font color, varying from bright red to dark grey, is determined by its *recency* (based on average publication date). In (Song et al., 2008), Song *et al.* proposed a graph-based clustering technique to predict tags for recommendation by learning from existing tagged documents. Some other well-known cloud generators are TopicRank (Berlocher et al., 2008), in which positioning of tags is based on their semantic closeness; Page History Explorer (Jatowt et al., 2008) to display clouds to visually explore the contents of a web page over time; and Data clouds (Koutrika et al., 2009), which uses terms of complex objects retrieved for a query from a database to display as a cloud.

Almost all existing works have completely ignored to enrich the tags which form the basic components in the cloud. Although, there is considerable number of research to extract keyphrases from text documents, to the best of our knowledge, no attempt has been made to consider keyphrases as tags while generating a tag cloud. We provide a brief review of the state-of-the-art techniques for keyphrase extraction in the following paragraph.

In most of the cases, keyphrase extraction is viewed as a supervised machine learning task which requires annotated text documents containing manually tagged keyphrases for training purpose. Turney (1999) proposed a keyphrase extractor, GenEx, whose 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 (2000) also proposed "*Extractor*", which uses a set of nine features including *phrase position*, *phrase frequency*, *phrase length*, etc. to generate a score for a candidate phrase.

Frank et al. (1999) proposed KEA[4], which uses naïve Bayes learning method to induce a probabilistic model from training corpus. A classification model using two basic features is learned on a set of annotated training documents for the purpose of identifying keyphrases in new documents. The authors claim that training on naïve Bayes is quicker than training GenEx, which employs a special purpose genetic algorithm. Due to its simplicity and decent accuracy, KEA has become a standard keyphrase extraction tool in various applications. For example, the Keyphind system proposed by Gutwin *et al.* (1999) to support keyphrase-based search uses KEA in background to identify keyphrases in text documents. KEA++, proposed by Medelyan and Witten (2006), is an extended version of KEA in which they improved the feature set by introducing more features.

For topic indexing, in (Medelyan et al., 2008), Medelyan *et al.* added some Wikipedia-based features. Candidates are selected as the high-scoring Wikipedia articles mapped to extracted *n*-grams from the underlying collection based on a designed formula. Rest of the machine learning procedure remained same as their earlier work with an improved set of features. Maui, another system proposed by Medelyan et al. (2009), uses nine features including five lexical features and four Wikipedia-based features. On experimentation, they found that depending on the consideration of different subsets of the proposed nine features either decision tree yields better result than naïve Bayes or vice-versa. For example, on consideration of all nine features decision tree yields better result than naïve Bayes, whereas on consideration of only first three lexical features naïve Bayes classifier performs better than bagged decision tree classifier.

PROPOSED TAG CLOUD GENERATION FRAMEWORK

In this section, we present the design of the proposed keyphrase-based tag cloud generation framework. As shown in Figure 1, it primarily consists of five different components– *document pre-processor, candidate phrase extractor, feature extractor, keyphrase extractor,* and *cloud generator*. The proposed framework implements a supervised machine learning approach for the domains with available training dataset as well as a feature aggregation approach for the domains not having a proper training dataset, which happens in most of the cases. These two variations are only in the part of its fourth component, *keyphrase extractor*, and all the remaining subtasks are performed with no difference. Functioning detail of each module of the proposed framework is presented in the following sub-sections.

Document Pre-processor

This section presents the document pre-processing mechanism, which is basically confined to transform the input texts into machine readable format. It accepts text documents as input irrespective of its format. *HTML parser* is used to filter out the tags from web documents, whereas *pdftotext* is used to collect the textual content from *pdf* documents. Since, we are concerned only with textual content, all the images and their labels are filtered out during the text gathering process, which comes under the task of *cleaning*, as presented in figure 1. The cleaning process is followed by the process of *tokenization*, which is a process to decompose texts into record-size chunks. The boundaries of these chunks are determined heuristically using the appearance of various punctuation marks like comma, semicolon, full stop, inverted comma, opening or closing brackets, exclamation mark, question mark, and other special characters. Also, a full stop is cleverly differentiated from a simple period or dot which generally appears in association with acronyms or numeric decimal values.

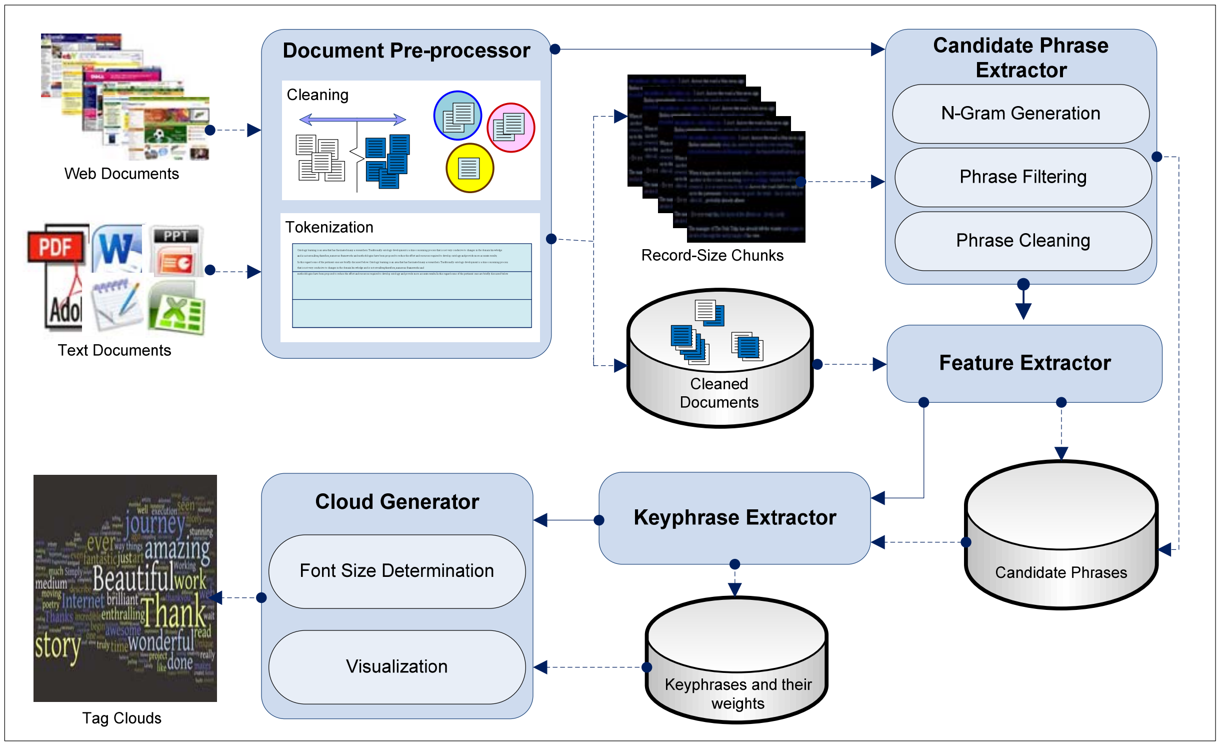

Figure 1: Architecture of the proposed keyphrase-based tag cloud generation framework

Candidate Phrase Extractor

This component is concerned with the task of candidate phrase extraction and is implemented as a two step process, *n-gram generation,* and *phrase filtering and cleaning*. Each of these sub-tasks is described in the following paragraphs.

N-gram Generation

An *n*-gram is defined as a sequence of *n* consecutive words from a block of text. 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 to a document or article contains 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. Therefore, in our method the value of *n* is restricted to 3, i.e., we generate only 1-, 2-, and 3-grams from the chunks generated by the *tokenizer*. This decision about the size of a phrase is based on an experiment provided in the *experimental evaluation* section. 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 in a structured format.

Phrase Filtering and Cleaning

In this phase, all *n*-grams (hereafter, we say phrases) are analyzed, insignificant phrases are filtered out, and the retained phrases are cleaned to determine the set of candidate phrases. To filter out irrelevant phrases, a set of heuristic rules is designed. These heuristic rules are briefly explained in the following paragraphs:

- 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*" by removing the stop-word "*in*", we get the phrase "*populations*", which could already be in the list of a 1-gram. Therefore, we have opted for dropping instead of cleaning.

- Filter out a phrase containing special characters, such as /, \\, %, etc. as they can never be a keyphrase. Like previous rule, we have opted phrase removal instead of phrase cleaning to reduce redundancy.
- Filter out a phrase containing no 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.

Phrases retained after applying the above-mentioned filtering rules are treated with the *phrase cleaning* process. During this process, in addition to removal of apostrophes, the words in a phrase starting or ending with numeric values are processed to drop the numeric constituents. Also the phrases are stemmed and case-folded. Stemming is defined as to remove the suffix from a word to reduce it to its base form. For this task, we have used the Porter stemmer (Porter, 1980), which implements heuristic rules to remove or transform English suffixes. Case-folding is defined as to convert the complete phrase into lower-case letters. Stemming and case-folding are applied to neutralize the numerous variations of a phrase and to consider them same. The set of cleaned phrases are finally considered as candidate phrases and further processed by the *feature extractor* module to identify their lexical and semantic features.

Feature Extractor

The feature extractor module takes the list of candidate phrases as input and transforms them to their feature vectors for characterizing them in terms of numeric values. We have identified a set of nine features for the supervised machine approach, out of which only sevens are applicable for the cases in which there are no annotated training documents and we have to apply feature aggregation process to identify keyphrases. A brief description along with the mathematical formulation of each identified feature is given in the following paragraphs.

Relative phrase frequency (W_{relf}): In line with the existing approaches, we assume that the importance of a phrase is directly proportional to its frequency count. Therefore, we count the number of times a phrase appears in a document and then normalize the count by dividing it with the maximum frequency of any candidate phrase and use the normalized value as a feature termed as *relative phrase frequency*. The relative phrase frequency of a phrase p_i is calculated using equation 1 in which $f(p_i)$ represents the frequency count of the phrase p_i.

$$W_{relf}(p_i) = \frac{f(p_i)}{max\,\{f(p_j)\}} \tag{1}$$

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 words of the phrase possess a high frequency. Therefore, we capture this information by using equation 2, 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 the number of words in it.

$$W_{cum}(p_i) = \log_2\left(1 + \frac{\sum_{j=1}^{l(p_i)} f(w_j)}{\max\left\{\sum_{k=1}^{l(p_i)} f(w_k)\right\}}\right) \tag{2}$$

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_i is calculated using equation 3, where $posf(p_i)$ represents the position of the first occurrence of p_i in document D.

$$W_{pos}(p_i) = \begin{cases} \left|1 - \frac{posf(p_i)}{\left(\frac{|D|}{2}\right)}\right|, & \text{if } posf(p_i) \neq \frac{|D|}{2} \\ \left|1 - \frac{posf(p_i)}{\left(\frac{|D|}{2}\right)}\right| + \frac{1}{\left(\frac{|D|}{2}\right)}, & \text{if } posf(p_i) = \frac{|D|}{2} \end{cases} \tag{3}$$

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 consider assigning higher weight to lengthy phrases so that its value for the complete phrase could be higher than the constituting words. This feature is modeled using equation 4.

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

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 associative measure among constituting words of a phrase p_i occurring in a document D, we have devised this feature and modeled using equations 5 to 11.

$$rel(p_i) = \log_2\left(1 + \frac{f(p_i)}{|D|}\right), \qquad if\ p_i is\ a\ single\ word \tag{5}$$

$$rel(p_i) = \log_2\left(1 + \frac{prob(w_1, w_2)}{prob(w_1) \times prob(w_2)}\right), \qquad if\ p_i is\ a\ double\ word, and\ w_1, w_2 \in p_i \tag{6}$$

$$rel(p_i) = \log_2\left(1 + \frac{prob(w_1,w_2)}{prob(w_1)\times prob(w_2)}\right) + \log_2\left(1 + \frac{prob(w_2,w_3)}{prob(w_2)\times prob(w_3)}\right) + \log_2\left(1 + \frac{prob(w_1,w_2,w_3)}{prob(w_1)\times prob(w_2)\times prob(w_3)}\right), if\ p_i is\ a\ triple\ word, and\ w_1, w_2, w_3 \in p_i \tag{7}$$

$$prob(w_j) = \frac{f(w_j)}{\max\{f(w_r)\}} \tag{8}$$

$$prob(w_j, w_k) = \frac{f(w_j, w_k)}{\max\{f(w_r, w_s)\}} \tag{9}$$

$$prob(w_j, w_k, w_l) = \frac{f(w_j, w_k, w_l)}{\max\{f(w_r, w_s, w_t)\}} \tag{10}$$

$$W_{rel}(p_i) = \frac{rel(p_i)}{max\{rel(p_j)\}} \tag{11}$$

Capitalization (W_{cap}): Any phrase appearing 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 to each phrase with this property, which is calculated by using equation 12, where $countUpper(p_i)$ is the number of words in p_i with its first letter in upper case, $l(p_i)$ and $f(p_i)$ denote the length and frequency, respectively of the phrase p_i.

$$W_{cap}(p_i) = \frac{\sum \frac{countUpper(p_i)}{l(p_i)}}{f(p_i)} \tag{12}$$

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 (equation 13) than the standard one formulated in (Salton & McGill, 1983). In equation 13, $f(p_i)$ represents the frequency of phrase p_i, $|D|$ is the length of document D, and *countDoc()* returns the number of documents containing the phrase supplied as an argument.

$$W_{tfidf}(p_i) = \frac{f(p_i)}{|D|} \times \left(-\log_2 \frac{\max\{countDoc(p_i)\}}{countDoc(p_i)}\right) \tag{13}$$

Lifespan (W_{lsp}): 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. The lifespan feature is defined using equation 14 in which $posl$ and $posf$ represent the positions of the first and last occurrences, respectively of the phrase p_i.

$$W_{lsp} = \frac{posl(p_i) - posf(p_i)}{|D|} \tag{14}$$

Keyphraseness (W_{key}): It is also an important feature and Frank *et al.* (Frank et al., 1999) 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. The keyphraseness feature is defined in equation 15, where *countPerfect(G_i,K)* gives the number of perfect matches in G_i and K, where G_i represents the set of 1-, 2- and 3-grams of phrase p_i, and K represents the set of keywords in training set.

$$W_{key}(p_i) = \frac{\sum countPerfect(G_i, K)}{\max\{\sum countPerfect(G_i, K)\}} \tag{15}$$

All the above-mentioned feature values lie in the scale of [0, 1]. Among them, W_{tfidf} and W_{key} are specific to corpus-based approach and can only be applied when we have a corpus for training purpose in which keyphrases are tagged. In case the tag cloud is to be generated only for a single document as happens most of the time, these two features are excluded from the set and the feature aggregation approach, described later, is applied.

Keyphrase Extractor

In this section, we provide the procedural detail to identify feasible keyphrases and their relevance weights that can be used to generate tag cloud. The task of tag cloud generation is usually applied to a single document or a block of text. In this scenario, we follow the feature aggregation approach to identify feasible keyphrases in single document. Due to growing popularity of tag cloud, it is also being used in online as well offline applications to conceptualize a huge repository of textual data. The PubCloud proposed by Kuo *et al.* (2007) is one example to deal with such cases. To this end, we have designed a comparatively more efficient and effective technique using supervised machine leaning approach. Both of the approaches are described in the following sub-sections.

Feature Aggregation Approach

This approach is followed when we are given with only a single document to identify keyphrases and generate tag cloud. In this case, the keyphrases are to be extracted from within the document, without having any earlier knowledge related to this domain. Therefore, we exclude *tf-idf* and *keyphraseness* features and consider only seven remaining features to identify keyphrases. For each phrase p_i, the feature values are calculated to represent it as a feature vector F = $<f_1, f_2, f_3, f_4, f_5, f_6, f_7>$, where each f_i represents the value of the corresponding feature. The aggregated score of the phrase p_i is calculated using equation 16, in which $\Lambda = \left\{\lambda_1, \lambda_2, \ldots, \lambda_{|F|_{C_2}}\right\}$ is a set of constants used to assign different weights to different feature combinations. The values of these constants are determined experimentally. Since all feature values are between 0 and 1, the aggregated score also lies in this range. Finally, the candidate phrases are ranked on the basis of their aggregated scores to place the most promising keyphrases on the top. Depending on the application, top ranked phrases are considered as keyphrases and their aggregated scores are used to determine the font-size for visualizing tag cloud.

$$W(p_i) = \frac{\sum_{i<j}(f_i \times f_j) \times \lambda_k}{\sum \lambda_k} \tag{16}$$

Supervised Machine Learning Approach

Currently, tag clouds are also gaining attention for domain-specific applications. PubCloud (Kuo et al., 2007) is a most suitable example, which generates tag cloud to conceptualize biomedical text corpus. In this section, we describe our supervised machine learning approach for keyphrase extraction that can be employed to generate tag clouds for domain-specific applications like PubCloud. Before the system is deployed for its real-time use, a model needs to be learned using a standard training dataset in which keyphrases are tagged. This is called *model learning* phase or *training* phase. The training dataset comprises the text files along with gold standard keyphrases. All nine feature values are computed for each candidate phrase and the class attribute value is assigned either POS if the phrase belongs to the set of gold standard keyphrases, or NEG otherwise. Thereafter, the set of feature vectors are provided to a classifier, which generates a trained classification model for the training dataset. It is performed only once during the training phase of the system for a particular application domain. In case the domain is susceptible to vary with time, the concept of incremental learning can be used to incorporate the changes in the trained model efficiently. Once a classification model for a particular application domain is trained, it can be used to identify keyphrases from unseen documents. For model learning and classification, we have used the naïve Bayes' machine learning classifier due to its simplicity and decent accuracy. In addition to classify a candidate phrase as *POS* or *NEG*, it provides their probabilistic values. The probabilistic values of the phrases are considered as their overall weights to rank them for identifying the most promising keyphrases. Moreover, these weights are used to determine the font-size of the corresponding keyphrases to visualize tag cloud.

Tag Cloud Generator

Searching for relevant information from vast collections of digital data is a common activity and the problem of retrieving this critical information was noticed long ago (Maron & Kuhns, 1960). However, the rapid growth of internet has enlarged its domain from digital libraries to the Web itself (Aula et al., 2005), 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* (Koutrika et al., 2009; Kuo et al., 2007). A tag cloud can be said as a collection of main topical terms mined from the voluminous texts 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 whether it is of interest or not. Sometimes, it provides an additional provision as the hyperlink in tags that allow a viewer to navigate and explore the underlying theme. 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 this framework, we implement a keyphrase identification approach to identify feasible keyphrases embedded within text documents and use them as tags. The component *Keyphrase Extractor* produces the output as ranked phrases along with their overall weight, *W(p_i)*, which characterize their prominence and keyphraseness. For visualization as a tag cloud, there are two major factors that emphasize a tag – *font size* and *placement*. Wordle (Kuo et al., 2007) uses a random placement scheme in which the font size is determined by using the frequency of tags or words. In most of the existing techniques the font size of a tag is determined using its frequency, but they differ largely in terms of placements of tags. Some of them place the tags horizontally in the same order as they appear in the actual text document. However, according to our perception, the objective of the visualization scheme should be only to highlight and emphasize the most prominent tags. For this purpose, the tag cloud generator module uses the weight of identified keyphrases $W(p_i)$ to determine their font size $F(p_i)$ for tag cloud visualization. Equation 17 presents the font size generation function, where F_{max} and F_{min} represent the maximum and minimum font size values supplied by the user. The tags are ordered in terms of decreasing font size. The topmost tag is placed at the center of the visualization window or at the cloud center, and the following tags are subsequently placed around it moving farther gradually with decreasing font size.

$$F(p_i) = (F_{max} - F_{min}) \times \left(\frac{W(p_i) - \min\{W(p_j)\}}{\max\{W(p_j)\} - \min\{W(p_j)\}} \right) + F_{min} \tag{17}$$

EXPERIMENTAL EVALUATION

In this section, we present evaluation results of the proposed keyphrase-based tag cloud generation framework to establish its efficacy. A substantial contribution in this framework is the keyphrase extraction method, which identifies feasible keyphrases to be considered as tags in the cloud. Hence, our focus is very much remained centered to enrich the tags by identifying prominent keyphrases. Therefore, in the first sub-section we evaluate our feature aggregation based keyphrase extraction method, followed by the evaluation of the supervised keyphrase extraction method, and finally the tag cloud generation in last subsection.

Why *n*-grams restricted up to 3-grams?

In previous works, the *n*-gram technique has remained most popular due to its light-weight and simplicity (Frank et al., 1999). Another popular and worthy technique is the use of Parts-Of-Speech (POS) tags to identify noun phrases and use them as candidate phrases (Hulth, 2003). Due to complex morphological structures of English sentences and the intricacies of parsing them for POS tag identification, it becomes a computationally expensive task to identify candidate phrases in this manner. Considering the *n*-gram technique for candidate phrase identification poses another type of challenge as what should be the ideal value of *n*. In order to determine an appropriate value for *n*, we have performed an experiment on the 2009 IEEE Taxonomy Version 1.01, which consists of technical terms that are organized hierarchically into 24 major categories and 3021 internal categories at different levels making a total of 3045 terms. These terms are categorized into six bins according to the number of words in them. Figure 2 visualizes the statistics of these bins. The horizontal axis represents the bins and the vertical axis represents the frequency count. From figure 2(a), we conclude that the top most hierarchical terms are up to a maximum of 5 words and most of them are double-word terms. Another observation is that there are considerable numbers of terms containing four and five words. But, when we look at the inner hierarchical terms in figure 2(b), we find that majority of terms contain two words; one word and three word terms are almost equal in number and a very little portion of terms contains more than three words. To reduce computational overheads, in our experiments, we welcomed the tradeoff to restrict the *n*-grams to a maximum of 3-grams at the cost of ignoring the terms which exceed this limit. However, in real time applications it may be fixed according to the domain and nature of its terms.

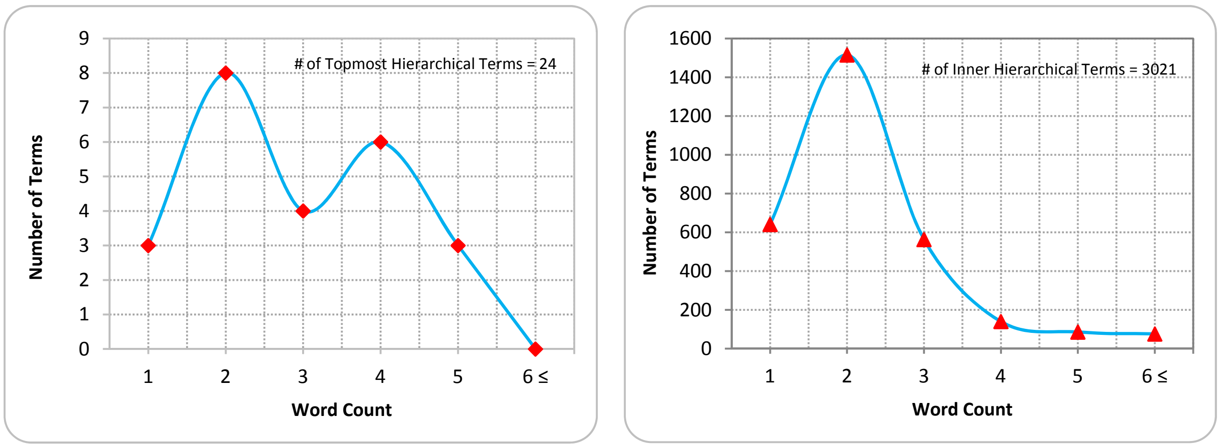

Figure 2: Trends of terms' length in 2009 IEEE Taxonomy V-1.01 (a) Terms at the root-level of the taxonomy hierarchy (b) Terms at the inner-levels of the taxonomy hierarchy

Experimentation with Keyphrase Extraction Method

In this experiment, we present a comparison result of our keyphrase extraction method with KEA proposed by Frank *et al.* (1999). For evaluation of the experimental results, we 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 (number of correctly identified keyphrases), the false positive FP (number of incorrectly identified keyphrases), and false negatives FN (number of actual keyphrases ignored) and by using these values we calculate the values of Precision, Recall and F-measure using equations 18, 19 and 20, respectively.

$$Precision\ (\pi) = \frac{TP}{TP + FP} \tag{18}$$

$$Recall\ (\rho) = \frac{TP}{TP + FN} \tag{19}$$

$$F\text{-}measure\ (F_1) = 2 \times \frac{\pi \times \rho}{\pi + \rho} \tag{20}$$

The experiment is conducted on a standard agricultural dataset downloaded from KEA website, along with their gold standard keyphrases. The training and test datasets consists of 20 and 5 large-sized text documents, respectively. Experiment for the feature aggregation approach is performed on the five test documents, whereas for the supervised approach the training set is used for learning naïve Bays classification model and the test set is used for its evaluation. The extracted keyphrases are arranged in decreasing order of their class probability values generated by the classification system, where the most promising keyphrases occupy top places. We have calculated the values of *Precision*, *Recall* and *F-measure* on 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. Since the number of author-assigned keywords does not always remain the same for all documents, we 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 fixed as 3, 5, 7, and 9 for each document to compute values of evaluation metrics. Table 1 presents the comparison results of our both feature aggregation and supervised machine learning method with KEA. It can be seen in table 1 that for all cut-off values the proposed supervised learning method outperforms both feature aggregation method and the KEA, whereas the performance of the feature aggregation method is almost comparable to KEA. Thus, we find that the proposed machine learning approach is very suitable for domain-specific applications.

Figures 3-5 present the trends of the macro-averaged *Precision*, *Recall* and *F-measure* values, respectively for all three approaches. The lines of the proposed feature aggregation approach and the KEA are clearly following almost same trend, and that the line corresponding to the proposed supervised approach is always at top of them. Thus, without any doubt, the proposed supervised approach outperforms the other two. Moreover, it can be observed that precision values are monotonically decreasing, whereas the recall is monotonically increasing when the cut-off point is raised.

Table 1. Comparison results of the proposed keyphrase extraction methods with KEA

Cut-off Point	**Proposed Method$_{FA}$**				**Proposed Method$_{Sup}$**				**KEA**			
	TP	**FP**	**FN**	**π/ρ/F$_1$**	**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	1	2	8	0.33/0.11/0.17
Top-5	1	4	8	0.20/0.11/0.14	3	2	6	0.60/0.33/0.43	1	4	8	0.20/0.11/0.14
Top-7	2	5	7	0.29/0.22/0.25	3	4	6	0.43/0.33/0.37	1	6	8	0.14/0.11/0.12
Top-9	2	7	7	0.22/0.22/0.22	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	0.67/0.20/0.31	2	1	8	0.67/0.20/0.31	1	2	9	0.33/0.10/0.15
Top-5	2	3	8	0.40/0.20/0.27	3	2	7	0.60/0.30/0.40	2	3	8	0.40/0.20/0.27
Top-7	2	5	8	0.29/0.20/0.24	3	4	7	0.43/0.30/0.35	2	5	8	0.29/0.20/0.24
Top-9	3	6	7	0.33/0.30/0.32	3	6	7	0.33/0.30/0.32	2	7	8	0.22/0.20/0.21
Document – 3												
Top-3	1	2	3	0.33/0.25/0.29	2	1	2	0.67/0.50/0.57	2	1	2	0.67/0.50/0.57
Top-5	1	4	3	0.20/0.25/0.22	2	3	2	0.40/0.50/0.44	2	3	2	0.40/0.50/0.44
Top-7	2	5	2	0.29/0.50/0.36	2	5	2	0.29/0.50/0.36	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	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	1	2	6	0.33/0.14/0.20
Top-5	2	3	5	0.40/0.29/0.33	2	3	5	0.40/0.29/0.33	1	4	6	0.20/0.14/0.16
Top-7	2	5	5	0.29/0.29/0.29	2	5	5	0.29/0.29/0.29	1	6	6	0.14/0.14/0.14
Top-9	2	7	5	0.22/0.29/0.25	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	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	1	4	16	0.20/0.06/0.09
Top-7	2	5	15	0.29/0.12/0.17	1	6	16	0.14/0.06/0.08	3	4	14	0.43/0.18/0.25
Top-9	2	7	15	0.22/0.12/0.15	2	7	15	0.22/0.12/0.15	5	4	12	0.56/0.29/0.38
Macro-Average												
Top-3	6	9	41	0.40/0.13/0.19	7	8	40	0.47/0.15/0.23	6	9	41	0.40/0.13/0.20
Top-5	7	18	40	0.28/0.15/0.19	11	14	36	0.44/0.23/0.30	7	18	40	0.28/0.15/0.19
Top-7	10	25	37	0.29/0.21/0.24	11	24	36	0.31/0.23/0.26	9	26	38	0.26/0.19/0.22
Top-9	11	34	36	0.24/0.23/0.24	13	32	34	0.29/0.28/0.28	11	34	36	0.24/0.23/0.24

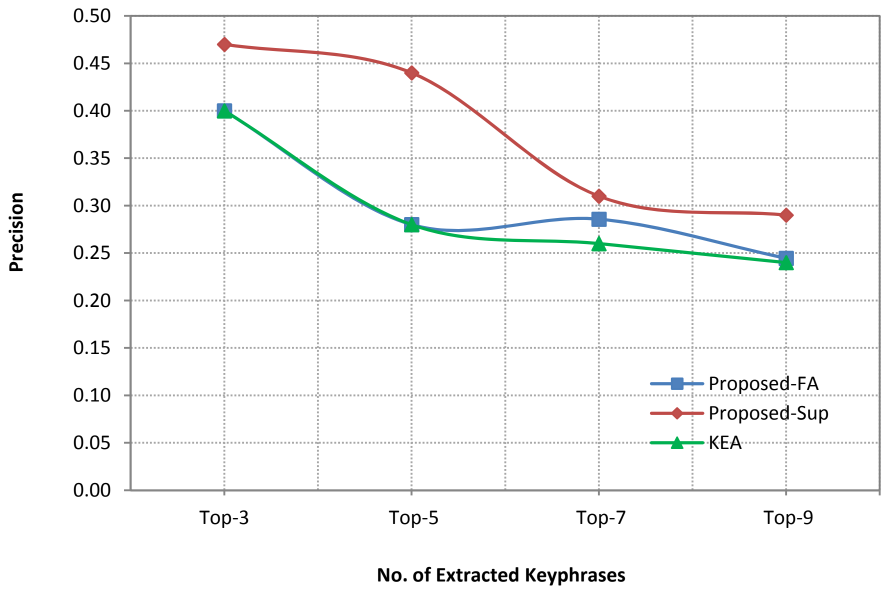

Figure 3. Visualization of the evaluation results of the proposed keyphrase extraction methods and KEA – *precision*

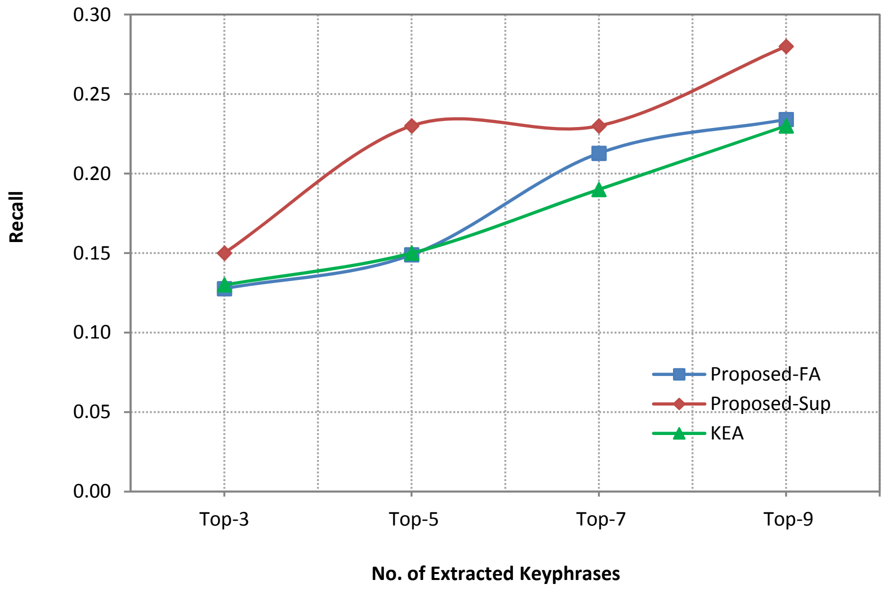

Figure 4. Visualization of the evaluation results of the proposed keyphrase extraction methods and KEA – *recall*

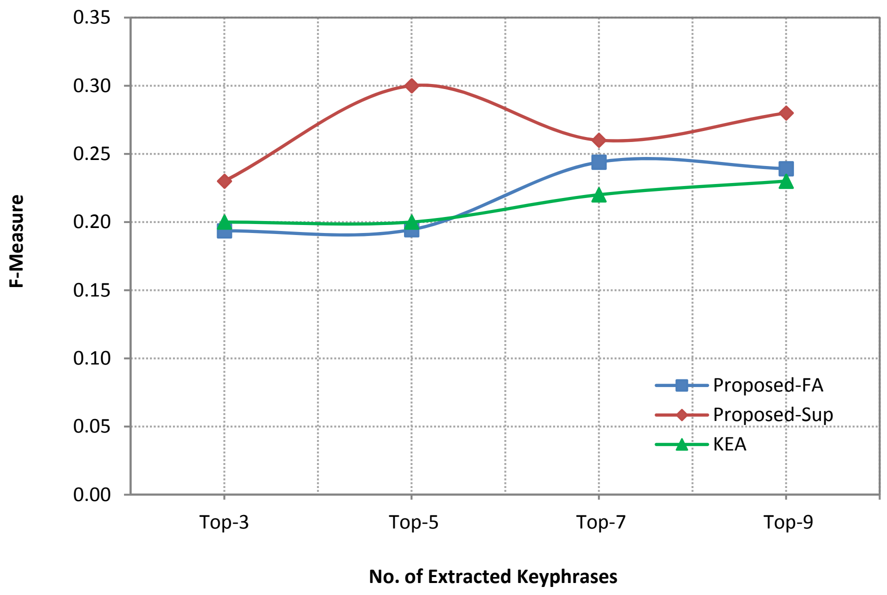

Figure 5. Visualization of the evaluation results of the proposed keyphrase extraction methods and KEA – *F-score*

For supervised learning approach, in addition to naïve Bayes classifier, we have also considered other classification algorithms and compared their accuracy values. The same dataset is treated with the same approach but with different classifiers each time, which includes *naïve Bayes*, *Decision Tree* (*C4.5*), *Multi-Layer Perceptron (MLP)*, and *RIPPER*. From classification results, we have calculated 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). Thereafter, we use the evaluation metric, *accuracy*, defined in equation 21 to compare the results produced by different classifiers. The accuracy values of all classifiers are shown in table 2. It can be observed from the results presented in table 2 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 be used for the classification purpose.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FN + FP + TN}} \tag{21}$$

Table 2. Classification accuracy of four different classifiers using the 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

Experimentation with Tag Cloud Generation

Although, the main contribution of this paper is the identification of enriched tags from texts to generate tag cloud, we continued our experiment to generate clouds using the extracted tags from the same dataset. In this section, we present the final part of our experimentation related to tag cloud generation and its visual presentation. For a document, table 3 shows top 20 keyphrases considered as tags in the cloud along with their probabilistic values and calculated font sizes using maximum and minimum font size as 48 and 5, respectively. Figure 6 shows the generated tag cloud. It can be observed in this cloud 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 the highlighted terms in the context, by going through which we can easily choose to move inside the document for more details or switch on to other term as per the requirement. Moreover, the size of this cloud can be enlarged further to accommodate more terms in it to have a broader overview of the content. It can be set proportional to the depth of review we need to have.

Table 3: 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

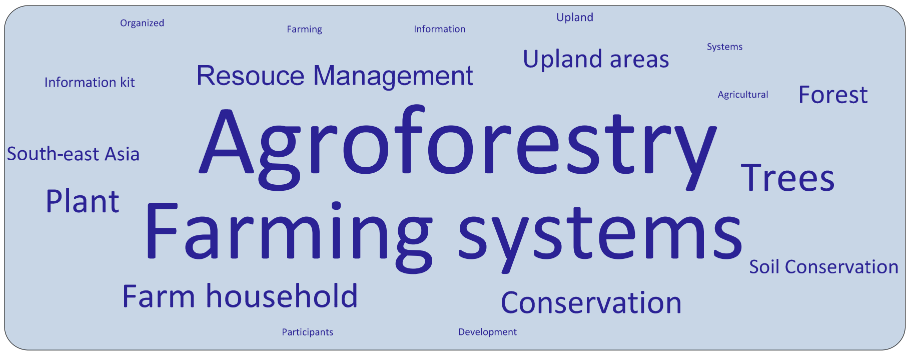

Figure 6: A sample tag cloud visualization of the tags in table 3 extracted by our method

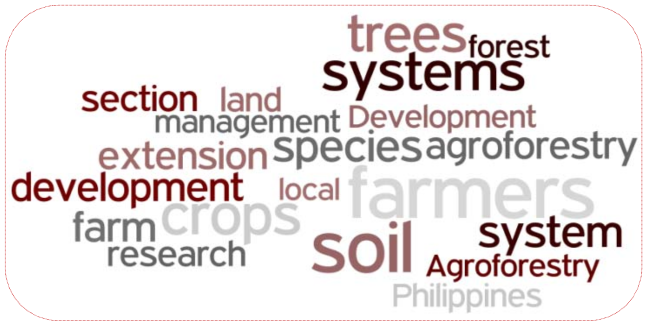

Figure 7: Tag cloud generated by Wordle

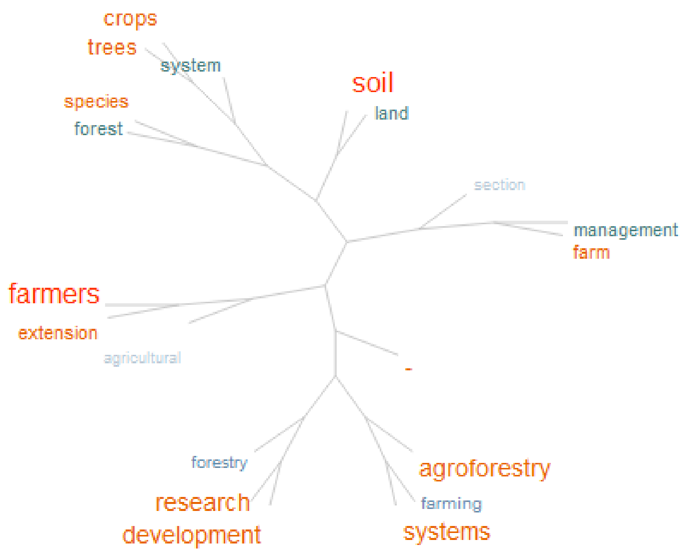

Figure 8: Tag cloud generated by Tree Cloud

Figures 7 and 8 show the tag clouds generated by Wordle and Tree Cloud, respectively for top 20 terms of the same document. In the Tree Cloud, Jaccard coefficient is used as the co-occurrence distance formula to generate the cloud. Among the three clouds shown here, Wordle makes use of the least intelligence, which is very clear from the visualization. It just counts the occurrence of individual words to determine their font size and places them randomly according to the selected layout, font type, and color pattern. The visualization is of course very simple to understand for a naïve user, but at the same time, the theme of underlying document still remains ambiguous just because of the breakdown of multi-word phrases into single words. Moreover, the font sizes are not much appropriate to emphasize the prominence of tags.

The cloud generated by Tree Cloud in which the placement of terms is determined intelligently using their co-occurrence feature is little-bit technical to grasp the theme. More a pair of terms co-occur in the underlying text, more closely they occur in the tree-structured cloud. For example, the term "*farming system*" is a complete term with two individual words, and their co-occurrence in the document is very high. Due to this fact, the two words appear as two leaves in the same branch of the tree. Thus, the tree-structured cloud is conceptually richer and sounds well than the simple and haphazard cloud of Wordle.

WHY YET ANOTHER TAG CLOUD GENERATOR?

In previous section, we have presented the experimental evaluation and comparison results of the keyphrase extraction module, which constitute an important component of the proposed framework. We have also compared our visualization scheme with two existing tag cloud visualization techniques – *Wordle* and *Tree Cloud* that present the theme of underlying texts in different ways. In the following paragraph, we discuss some of the major issues associated with existing tag cloud generation techniques, followed by the contribution of the proposed framework to handle them up to some extent.

Issues:

- Most of the existing techniques consider tag as a single word. When a multi-word phrase is broken into individual words it fails to retain the original information and the individual words can not be used independently to infer it. For example, the term "*tag cloud generation*" is considered as three individual words that individually focalize on different directions with different themes.
- Font size of a tag is determined simply on the basis of their frequency counts in underlying text. In this approach, a common tag, even of being worthless, may have high frequency leading to its display using large font size. For example, the word "*section*" is one such tag in the tag cloud generated by Wordle shown in Figure 7.
- The positioning of tags is another issue. Some consider positioning tags sequentially in alphabetical order (e.g., Flickr), some place them randomly (Kuo et al., 2007), and some other place tags in a clustered layout (Hassan-Montero & Herrero-Solana, 2006).

Our contributions:

- The proposed framework is integrated with a light-weight keyphrase extraction module. Instead of considering individual words as tags, we consider extracted keyphrases, which may contain a single, double or triple word, as tags. The keyphrases are able to convey the exact theme of a text block, because they keep the complete information intact in them.
- Instead of simply using the frequency count of a tag to determine its display font size, the proposed framework considers its identified relevance weight with respect to the underlying text corpus. This highlights the importance of a tag in proportion to its degree of relevance to represent the theme of the underlying text corpus.
- According to the comparative study of tag cloud layouts performed by Lohmann *et al.* (2009), there exist no single best way to arrange weighted terms in a cloud. An optimal layout is dependent strongly on its application purpose. When a user is looking for a specific tag in the cloud, it is the sequential layout to serve best; when one is looking for tags belonging to a certain area or topic, it is the clustered layout to serve best; but, when the purpose is to find the most popular tags, circular layout serves the best. A circular layout places the most weighted tag at the center and the lower weighted tags outwards. Thus, we found that layout is not a major issue, and it can be determined based on the associated purpose.

CONCLUSION AND FUTURE WORK

In this paper, we have presented the design of a keyphrase-based tag cloud generation framework to conceptualize textual data. Instead of using partial or full parsing, the proposed method uses *n*-gram techniques to identify candidate phrases and refines them using a set of heuristic rules. A set of lexical and semantic features is proposed to characterize candidate phrases and determine their keyphraseness. The novelty of the proposed framework lies in its keyphrase extraction method, which uses a supervised machine learning approach for a domain with an annotated set of training corpus, or a simple feature aggregation approach, otherwise. We have also defined a font-size determination function to map the relevance weights of the keyphrases generated during feasibility analysis process to their respective font sizes for tag cloud visualization. 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. Currently, we are enhancing our proposed framework to identify the semantic relations between tags (keyphrases) and visualize them in tag clouds. Exploration of more features to enhance the performance of the proposed keyphrase extraction method is also one of our future works.

REFERENCES

Abulaish, M., & Anwar T., (2011). A Web Content Mining Approach for Tag Cloud Generation, In 13th Int'l. Conf. on IIWAS, (pp. 52-59)

Abulaish, M. & Anwar, T., (2012). A Supervised Learning Approach for Automatic Keyphrase Extraction, Int'l J. of Inn. Comp. Inf. and Ctrl. (IJICIC), 8(11), (to appear)

Angel, A., Koudas, N., Sarkas, N., & Srivastava., D., (2007). What's on the grapevine?, In Proc. of the SIGMOD, (pp. 1047-1050)

Aula, A., Jhaveri, N. & Kaki, M. (2005). Information search and re-access strategies of experienced web users, In Proc. of the 14th Int‘l conference on WWW, (pp. 583–592)

Bansal, N., & Koudas., N., (2007a). Searching the blogosphere. In WebDB.

Bansal, N. & Koudas., N., (2007b). Blogscope: spatio-temporal analysis of the blogosphere. In Proc. of the Int'l Conf. on WWW, (pp. 1269-1270)

Berlocher, I., Lee, K.-I., & Kim, K., (2008). TopicRank: Bringing Insight to Users. In Proc. of the SIGIR, (pp. 703-704)

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C. & Nevill-Manning, C. G. (1999). Domain-specific Keyphrase Extraction, In Proc. of the 16th Int‘l J. Conf. on AI (IJCAI), San Mateo, CA

Gambette, P. & Véronis, J., (2009). Visualising a Text with a Tree Cloud, In Proc. of the 11th IFCS Biennial Conf.

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

Han, J., Kim, T. & Choi, J. (2007). Web Document Clustering by using Automatic keyphrase extraction. In Proc. of the 2007 IEEE/WIC/ACM Int‘l Conf. on WI and IAT, (pp. 56–59)

Hassan-Montero, Y. & Herrero-Solana, V., (2006). Improving Tag-Clouds as Visual Information Retrieval Interfaces. In Proc. of Int‘l Conf. on Multidisc. Inf. Sc. and Tech. (INSTAC), Mérida

Hulth,, A., (2003). Improved Automatic Keyword Extraction Given More Linguistic Knowledge, In Proc. of the Int‘l Conf. on Emp. Meth. in NLP (EMNLP), Sapporo, Japan, (pp. 216-223)

Jaffe, A., Naaman, M., Tassa, T., & Davis, M., (2006). Generating Summaries and Visualization for Large Collections of Geo-Referenced Photographs, In Proc. of the MIR '06, (pp. 89-98)

Jatowt, A., Kawai, Y., & Tanaka. K., (2008). Visualizing Historical Content of Web Pages. In WWW, (pp. 1221-1222)

Jones, S., & Staveley, M.S. (1999). Phrasier: A System for Interactive Document Retrieval using Keyphrases. In Proc. of the 22nd Ann. Int‘l ACM SIGIR Conf., (pp. 160–167)

Jonse, S. & Mahoui, M. (2000). Hierarchical Document Clustering using Automatically Extracted Keyphrase. In Proc. of the 3rd Int‘l As. Conf. on Digital Libraries, Seoul, Korea, (pp. 113–120)

Kaser, O. & Lemire, D. (2007) TagCloud Drawing: Algorithms for Cloud Visualization, In Proc. of the 16th Int‘l Conf. on WWW, Canada

Kosovac, B., Vanier, D. J. & Froese, T. M. (2000). Use of Keyphrase Extraction Software for Creation of an AEC/FM Thesaurus. J. of Inf. Tech. in Const., 5, (pp. 25–36)

Koutrika, G., Zadeh, Z. M. & Garcia-Molina, H. (2009). Data clouds: summarizing keyword search results over structured data, In Proc. of the 12th Int‘l Conf. on Extending Database Technology: Advances in Database Technology, (pp. 391–402).

Kuo, B. Y. Hentrich, T., Good, B. M. & Wilkinson, M. D. (2007). Tag clouds for summarizing web search results. In Proc. of the 16th Int‘l Conf. on WWW, (pp. 1203–1204).

Kupiec, J., Pedersen, J. & Chen, F. (1995). A Trainable Document Summarizer. In Proc. of the SIGIR Conf., ACM Press, (pp. 68–73)

Li, Q., Wu, Y.B., Bot, R. & Chen, X. (2004). Incorporating Document Keyphrases in Search Results, In Proc. of the 10th Am. Conf. on Inf. Sys., New York

Light, M., Mann, G.S., Riloff, E. & Breck., E., (2001). Analyses for elucidating current question answering technology, Nat. Lang. Eng. 7(4), (pp. 325-342)

Lohmann, S., Ziegler, J., & Tetzlaff. L. (2009). Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration. In Proc. of the 12th IFIP TC 13 Int‘l Conf. on Human-Computer Interaction: Part I (INTERACT), (pp. 392-404)

Maron, M. E. & Kuhns, J. L. (1960). On Relevance, Probabilistic Indexing and Information Retrieval, J. of the ACM, 7(3), (pp. 216–244)

Medelyan, O. & Witten, I. H. (2006). Thesaurus-Based Automatic Keyphrase Indexing, In Proc. of the 6th ACM/IEEE-CS J. Conf. on Digital Libraries, New York, USA, (pp. 296–297)

Medelyan, O., Witten, I. H. & Milne, D. (2008). Topic Indexing with Wikipedia, In Proc. of AAAI Workshop on Wikipedia and AI: an Evolving Synergy, Chicago, USA, (pp. 19–24)

Medelyan, O., Frank, E., & Witten, I. H. (2009). Human-Competitive Tagging using Automatic Keyphrase Extraction. In Proc. of the Int‘l Conf. of Emp. Meth. in NLP (EMNLP), Singapore, (pp. 1318–1327)

Milgram, S., & Jodelet, D., (1976). Psychological maps of Paris, W.I.H. Proshansky and L. Rivlin, eds., Environmental psychology, New York: Holt, Rinehart, and Winston, (pp. 104-124)

Pang, B. & Lee, L. (2008). Opinion Mining and Sentiment Analysis, Found. Trends Inf. Retr. 2(1-2), (pp. 1-135)

Papapetrou, O., Papadakis, G., Ioannou, E., & Skoutas., D., (2010). Efficient term cloud generation for streaming web content. In Proc. of the 10th Int‘l Conf. on Web Engg (ICWE), Boualem Benatallah, Fabio Casati, Gerti Kappel, and Gustavo Rossi (Eds.). Springer-Verlag, Berlin, Heidelberg, (pp. 385-399)

Platakis, M., Kotsakos, D. & Gunopulos, D., (2009). Searching for events in the blogosphere. In Proc. of the Int'l Conf. on WWW, (pp. 1225-1226)

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

Salton, G., & McGill, M. J. (1983). Introduction to Modern Information Retrieval, McGraw-Hill, New York, NY

Seifert, C., Kump, B., Kienreich, W., Granitzer, G., & Granitzer, M., (2008). On the Beauty and Usability of Tag Clouds, In Proc. IV '08, (pp. 17-25)

Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W. & Giles, C. L. (2008). Realtime automatic tag recommendation. In Proc. of the 31st Ann. Int‘l ACM SIGIR Conf., Singapore, (pp. 515–522).

Turney, P. D. (2000). Learning Algorithm for Keyphrase Extraction, J. of Inf. Ret., 2(4), (pp. 303–36)

Turney, P. D. (1999). Learning to Extract Keyphrases from Text, National Research Council, Inst. for Inf. Tech., Techn. Rep. ERB-1057

Viégas, F.B., Wattenberg, M., & Feinberg, J., (2009). Participatory Visualization with Wordle, IEEE TVCG (InfoVis '09), 15(6), (pp. 1137-1144)

Zha, H. (2002). Generic Summarization and Keyphrase Extraction using Mutual Reinforcement Principle and Sentence Clustering. In Proc. of the 25th Ann. Int‘l ACM SIGIR Conf., (pp. 113–120)

---

[1] *http://www.flickr.com/photos/tags*
[2] *http://delicious.com/tag*
[3] *http://technorati.com/tag/*
[4] http://www.nzdl.org/Kea/