Keyword and Title Based Clustering (KTBC): An Easy and Effective Way to Dynamically Cluster Web Documents

Ashis Talukter¹
A K M Ziaur Rahman²

Abstract: Web search engine users are most often bound to search documents through a huge list of web documents returned by the search engine. With rapid proliferation of web documents on internet, fast and effective mining of information from this data sources scattered all over the world has become a challenge to the Information Retrieval (IR) community. The IR community has explored document clustering as an alternative way of organizing retrieval results but clustering has yet to be deployed on many search engines. In this research, an effective clustering approach: Keyword and Title Based Clustering (KTBC) algorithm has been proposed. The KTBC algorithm is a fast, post-retrieval web document clustering method, suitable to be used by web search engines. Instead of viewing an extremely large list of documents, the algorithm returns a smaller number of clusters which will help web users finding relevant information at more ease. Here we have provided an algorithmic methodology along with mathematical and logical analysis and finally simulation result of the algorithm.

Keywords: clustering, web mining, data mining, web clustering, KTBC, mining, KDD.

1. Introduction:

Definition of better living has been changed in the era of information technology. Now, the definition of pretty and precise living means that living in the information and updating it with the updated and latest information.

The evolution of internet into the global information infrastructure has lead to explosion in the amount of available information. Realizing the vision of distributed knowledge access in this scenario and its future evolution will need tools to “personalize” the information space (Krishnapuran et al, 2007). The major reason that data mining has attracted a great deal of attention in the Information Retrieval (IR) community in information industry in recent years is due to the wide availability of huge amount of data and the imminent need for turning such data into useful information and knowledge. The information and knowledge gained can be used for applications ranging from business

¹ Assistant Professor, Management Information Systems, University of Dhaka, Dhaka-1000
² Assistant Professor, Department of CSE, University of Dhaka, Dhaka-1000
management, production control, and market analysis, MIS, DSS to engineering design and science exploration.

Data can be stored in many different types of databases scattered all over the globe. One database architecture that has been recently emerged is the data warehouse, a repository of multiple heterogeneous data sources, organized under a unified schema called Star Schema (Silberschatz et al., 2006) at a single site in order to facilitate management decision-making. The abundance of data, coupled with powerful data analysis tools, has been described as a "data rich but information poor" situation. The fast-growing tremendous amount of data, collected and stored in a large and numerous databases, has far exceeded our human ability to comprehend without powerful tools. As a result, data collected in large databases become "Dam tomb"—data archives that are seldom visited (Han and Kamber, 2006). Important decisions are often made based not on the information-rich data stored in databases but rather on a decision maker's intuition, simply because the decision maker does not have the tools to extract the valuable knowledge embedded in the large amount of data.

Data mining tools perform data analysis and may unveil important data patterns, contributing greatly to business strategies, knowledge bases, and scientific and medical research. The widening gap between data and information calls for a systematic development of data mining tools that will turn data tombs into "golden nuggets" of knowledge (Han and Kamber, 2006).

Also it is the reality that conventional document retrieval systems return long lists of ranked documents that users are forced to sift through to find relevant documents. The majority of today's Web search engines (e.g., Excite, AltaVista, Google, and Yahoo) follow this paradigm. The low precision of Web search engines coupled with the ranked list presentation makes it hard for users to find the information they are looking for. The gradual data enrichment of Web day by day has made the situation more challenging to the IR community. The goal of this research is to make results of the search engine easy to browse and extract relevant Web information from internet by clustering.

2. Methodology

Methodology includes:

a) **Algorithmic methodology:** first an algorithm is devised to cluster the web documents.

b) **Logical Analysis:** Then a logical and mathematical analysis is included describing complexity, correctness and efficiency of the algorithm.

c) **Simulation Result:** Finally a simulation result of the algorithm is appended and a relative comparison is provided as well.
3. Preliminaries

In this section the concept of Knowledge Discovery in databases (KDD) and Data Mining will be introduced. In addition, an extensive and laconic overview of terminologies that had to be kept in mind through this research will be provided. Being an effective data-mining tool, Clustering Analysis will also be illustrated with care.

The technique involving discovery of useful summaries of data from large data base or data-warehouse is known as Data-Mining (See Figure 1). It is an intelligent method that is applied to extract data patterns. Many people treat data mining as a synonym for another popularly used Knowledge Discovery in Databases (KDD). A more or less official definition of KDD is: 'the non-trivial extraction of implicit, previously known, and potentially useful knowledge from data (Han and Kamber, 2006).’ So the knowledge must be new, not obvious, and one must be able to use it. A data warehouse is a repository of information collected from multiple sources, stored under a unified schema and which usually resides at single site.

Clustering (See Figure 2) technique to mine web data is the heart of this research. It is the major step in KDD process. Clustering is the process of grouping the data into classes or clusters so that objects (here web pages) within the same cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters (Han and Kamber, 2006).

![Figure 1: Architecture of a typical data mining System (Han and Kamber, 2002)](image-url)
4. Existing Study:

Numerous document-clustering algorithms appear in the literature. Most clustering algorithms rely on an external similarity (e.g. Euclidian distance) measure between documents (Willet, 1988). This is typically calculated by representing each document as a weighted attribute vector, with each word in the entire document collection being an attribute in this vector (the vector-space model, Salton, 1989).

Existing Algorithms:

Several clustering algorithms like Agglomerative Hierarchical Clustering (AHC), Divisive Hierarchical Clustering (DHC), K-Means algorithm, Buckshot and Fractionation, Neural Network Approach, Suffix Tree Clustering (STC) etc. will be discussed in this chapter.

In general, there are two types of hierarchical clustering methods: Agglomerative (AHC) and Divisive (See Figure 3). Agglomerative method is a bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category.

Figure 2: A 2-D plot of customer data w.r.t customer located in a city, showing three clusters. A "C" marks each cluster "center" (Source: Inmon 1996)
On the other hand divisive method is a top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the distance between two closest clusters is above a certain threshold value.

The $k$-means algorithm takes the input parameter, $k$, and partitions a set of $n$ objects into $k$-clusters so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. Cluster similarity is measured in regard to the mean value of objects in a cluster, which can be viewed as cluster's center of gravity.

\[ \text{Figure 3: Agglomerative and divisive hierarchical clustering on data objects \{a, b, c, d, e\} (Han and Kamber, 2006).} \]

Suffix Tree Clustering (STC) is a clustering algorithm that is based on identifying the phrases that are common to groups of documents (Leouski and Croft, 1996). A phrase in this context is an ordered sequence of one or more words. We define a base cluster to be a set of documents that share a common phrase. STC has three logical steps (Zamir and Etzioni, 1998):

1. Document "cleaning",
2. Identifying base clusters using a suffix tree, and
3. Combining these base clusters into clusters.
Comparative Study:

In this sub-section we review previous work on document clustering algorithms and discuss their pros and cons. Document clustering has been traditionally investigated mainly as a means of improving the performance of search engines by pre-clustering the entire corpus. However, clustering has also been investigated as a post-retrieval document browsing technique (Allen et al., 1993).

Agglomerative Hierarchical Clustering (AHC) algorithms are probably most commonly used. These algorithms are typically slow when applied to large document collections. Single-link and group-average methods typically take $O(n^2)$ time, while complete-link methods typically take $O(n^3)$ time (Yoorhees, 1986). As Yoorhees’s experiments demonstrate, these algorithms are too slow [specially $O(n^3)$] to meet the speed requirement for one thousand documents.

Several halting criteria for AHC algorithms have been suggested (Milligan and Cooper, 1985), but they are typically based on predetermined constants (e.g., halt when 5 clusters remain). These algorithms are very sensitive to the halting criterion - when the algorithm mistakenly merges multiple "good" clusters, the resulting cluster could be meaningless to the user. In the Web domain, where the results of queries could be extremely varied (in the number, length, type and relevance of the documents), this sensitivity to the halting criterion often causes poor results. Another characteristic of the Web domain is that we often receive many outliers. This sort of "noise" reduces even further the effectiveness of commonly used halting criteria.

Linear time clustering algorithms are the best candidates to comply with the speed requirement of on-line clustering. These include the K-Means algorithm - $O(nkT)$ time complexity where $k$ is the number of desired clusters and $T$ is the number of iterations (Rocchio, 1966), and the Single-Pass method - $O(nK)$ were $K$ is the number of clusters created (Hill, 1968). One advantage of the K-Means algorithm is that, unlike AHC algorithms, it can produce overlapping clusters. The Single-Pass method also suffers from this disadvantage, as well as from being order dependant and from having a tendency to produce large clusters (Rasmus, 1992). It is, however, the most popular incremental clustering algorithm (as can be seen from its popularity in the event detection domain - see Korfhage, 1991). K-Means algorithm also suffers from biasness to the initial value of $k$. Even though STC is a linear time algorithm, when the number of snippets is small the STC algorithm may not include the proper documents or may not include proper documents in the cluster.
5. Proposed model: KTBC Algorithm

After studying existing techniques of clustering now a new clustering technique called **Keyword and Title-Based Clustering (KTBC)** algorithm is proposed in this context. The name implies the algorithm is based not only on keywords but also the title of the documents. Keywords determine the cluster and title gives the name to the cluster. The position of the KTBC module in the traditional search system is shown in the **Figure 4**.

![Figure 4: Position of KTBC module in the searching system.](image)

As illustrated, the result (large list of web documents) of the search engine is passed as input to the KTBC module and the output of the KTBC module (compared short list of clusters of web documents) is presented to the search engine users.

**The Philosophy of KTBC Algorithm:**

The KTBC works with all the keywords of a document (web page) at a time, and determines in which cluster the document should be placed. Here the distance/similarity to calculate cluster is measured by number of common keywords of the documents. The algorithm includes the following steps:

- a) Gathering data (keywords and title of web documents)
- b) Synonym calculation
- c) Cleaning data
- d) Clustering data
- e) Naming the clusters
Gathering Keywords and Title:

The first step is to gather the keywords of a web page. Reading keywords from meta-tag and title from the HTML file of the web page can do this, which is like:

```html
<title>Data Mining Community's Top Resource</title>
<meta name="keywords" content="data mining, analytics, web mining, data mining software, data mining jobs, data mining consulting, data mining courses, datamining, knowledge discovery, KDD"> 
```

Also with the keywords the URL of the web page also recorded, such as for the above document the URL, http://www.kdnuggets.com/ is also recorded.

Synonym Calculation:

Then all the set of keywords are populated with all the sets of keywords and their synonyms using traditional method. This strategy adds strength and intelligence to the KTBC module.

Cleaning Data:

We consider each word as a keyword and refresh the list of keywords for each web page after deleting the words, which will not be considered as keywords, for instance articles (‘a’, ‘an’, ‘the’), all pronouns, all auxiliary verbs, all conjunctions, all internet buzzwords like www, http, content etc.

Clustering Data:

This is the basic steps of the KTBC algorithm (Also See the Figure 5):

Step 1: All the documents are marked as non-clustered

Step 2: First documents is added to first cluster and marked as clustered and its keywords are added in the set CU.

Step 3: All the documents Di (2 ≤ i ≤ N) of the large list are checked whether to join in the cluster by the logic below:

\[ D_i = \text{the set of keywords containing the document no } i, \text{ and} \]

\[ n(D_i) = \text{the no of keywords in the set } D_i, \]

\[ CI = \text{Intersect of } CU \text{ and } D_i \]
IF ( \( n(CU) > n(Di) \) )
  IF ( \( \frac{n(Di)}{n(CU)} < 0.1 \) )
    IF( \( \frac{n(CI)}{n(Di)} > 0.25 \) )
      Di in included in the cluster
    ELSE IF ( \( \frac{n(CI)}{n(Di)} >= 0.5 \) )
      Di in included in the cluster
  ELSE
    IF ( \( \frac{n(CU)}{n(CI)} < 0.1 \) )
      IF( \( \frac{n(CU)}{n(CI)} > 0.25 \) )
        Di in included in the cluster
      ELSE If ( \( \frac{n(CU)}{n(CI)} >= 0.5 \) )
        Di in included in the cluster

Step 4: If Di is added to the cluster, then Di’s keywords are added to the cluster’s keyword list \( CU = CU U Di \) and Di is marked as clustered.

Step 5: This process continues until any all documents are marked as clustered.

**Naming the clusters:**

To assign a self-explanatory name to the clusters determined by the KTBC algorithm (See Figure 6) then considers the title the web document as a phrase and after cleaning the phrase (as mentioned in the KTBC algorithm) each part of a phrase is considered to be a "word". After determining a set of phrases in a cluster, and since each cluster can contain many phrases (size of cluster) that in turn contains many words, the KTBC displays at most the seven words (the number seven is so selected to keep the name of standard size, to avoid a very large name) with higher frequencies (if several have the same frequency, the words are selected according to their scanning sequence) because our goal is to create a compact cluster summary. The algorithm is stated in the Figure 6.
6. Performance Analysis

The KTBC algorithm is a fast post-retrieval document-clustering interface to a Web search engine. Here post-retrieval means that at first, web documents are searched through a web search engine with a phrase of interest, and the KTBC is the applied upon the documents returned by the search engine before viewing the huge list to the user. Thus the user will see a relatively small number of clusters containing relatively more similar documents as compared to a random and huge list returned by the traditional search engine.

![Algorithm KTBC()](image_url)

**Figure 5:** The KTBC algorithm.
The key task that is performed by the KTBC algorithm is to find the similarity of the documents keeping all the keywords of a document at a time and the cluster to which the document belongs. Conventional document retrieval systems return long lists of ranked documents that users are forced to shift through to find relevant documents. The majority of today's Web search engines (e.g., Excite, AltaVista, Google and Yahoo) follow this paradigm. The notoriously low precision of Web search engines coupled with the ranked list presentation make it hard for users to find the information they are looking for. Our goal is to make search engine results easy to browse by clustering them.

We have identified two key requirements for a post-retrieval document clustering system:

a) Efficiency
b) Speed

**Efficiency**

The user needs to determine at a glance whether the contents of a cluster are of interest. Therefore, the system has to provide concise and accurate cluster descriptions. If we analyze the KTBC algorithm, efficiency and improvement are obvious. The cluster name should be self-explanatory so as to be easily browse-able which is carefully ensured in the KTBC algorithm. Again the condition checking at the steps 8, 9, 14, 15 (Figure 5)

```
Begin
    Initialize all values of frequency [] to be 0;
    Repeat For each Cluster i do
        Repeat For each Document Di do
            Repeat For each word j in phrase title[i] of Di do
                If the word is already in title_word[j] in j th position
                    Then do
                        Increment frequency [j] by 1;
                Else do
                    Add the word at the end of title_word[k];
                    Increment frequency [k] by 1;
                End-if
            End-for-j
        End-for-Di
    End-for-i
    Pick seven words from title_word[] with higher frequency;
```

**Figure 6: Algorithm for naming a cluster.**
makes the algorithm more efficient than STC Algorithm. This ensures the inclusion of the proper documents in the clusters at the rudimentary level of cluster calculation when a new document may have more keywords (90%) than that of existing cluster (10%). Here selection criteria are relaxed to 25% (steps 8, 9, 14, 15 of Figure 5) from 50% (steps 11 and 17 of Figure 5).

**Speed**

The clustering system should not introduce a substantial delay before displaying the results. The clustering algorithm used must be able to cluster hundreds or even thousands of keywords seconds. The running time of our KTBC algorithm is $O(n^2)$, the running time of the naming algorithm is $O(knT)$, where $n = \text{number of documents}$, $k = \text{number of clusters}$ and $T = \text{number of words in the title of a document or in a cluster}$ both $k$ and $T$ are much less than $n$. Hence, the running time of KTBC algorithm can be approximated as $[O(n^2)]$ which is quite satisfactory.

*Figure 7* shows the simulation result that presents the time taken by the KTBC algorithm to cluster retrieved documents as a function of the number of documents retrieved. We recorded the clustering time of about one thousand documents and grouped them by the number of documents retrieved (by steps of 50); the time reported is the time required by the clustering module "the KTBC" (As feature of Simulation: C Program, Intel (R) Core(TM) CPU E7400 @ 2.80 GHz, 2.8GHz speed with 2BG RAM).

![Comparison Of Running Time](image)  
*Figure 7:* Average clustering time of the KTBC Algorithm
7. Conclusion

The KTBC algorithm is a post-retrieval document-clustering algorithm that does the cluster of web documents returned by a search engine effectively. In doing this challenging job of web documents clustering challenge was really faced with the fact that a search with a single phrase returns thousands of documents by any known search engine. Several characteristics make the KTBC a promising candidate for the clustering of search results.

First it is keyword-based, generating clusters even by grouping documents that share many phrases. Keywords are also useful in constructing concise and accurate descriptions of the clusters.

Second it does not adhere to any model of the data. Its only assumption is that documents on the same topic will share common keywords.

Third The KTBC uses a simple cluster definition - all documents containing one of the cluster's phrases are members of the cluster. There remains no trivial cluster (single documents in single cluster).

Finally The KTBC is a fast post-retrieval algorithm, which makes it suitable for online clustering of Web searches.

Limitation and Future Direction:

The clustering algorithm should group similar documents together. As documents often have multiple topics, they should not be confined to a single cluster (Rocchio, 1966); the clustering algorithm should generate overlapping clusters when appropriate. The KTBC does not create overlapping clusters. This is a limitation of the KTBC algorithm and left a future scope.

References:


Korfhage R., 1991, To see. or not to see - is that the query-?, in: Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'91), pp 134-141.


