Suffix Tree Clustering - Data mining algorithm

Milos Ilic[1], Petar Spalevic[2,] Mladen Veinovic[3]

[1,2] *Faculty of Technical Science Kosovska Mitrovica, University of Pristina-temporally seated in Kosovska Mitrovica*
[3] *Faculty of Informatics and Computing, Singidunum University, Belgrade*
E-mail: milos.ilic.pk@gmail.com, petar.spalevic@pr.ac.rs, mveinovic@singidunum.ac.rs

Abstract

Data Mining as a process of finding new, useful knowledge from data using different techniques. Using these techniques we getting faster and better search of large amounts of data that we facing every day. Clustering of data is one of the techniques that are used in data mining. Authors explore clustering algorithms and take suffix tree clustering algorithm for the best of them. Authors create an application that use this algorithm in the process of clustering, and search of clustered documents.

1 Introduction

The need to understand large, complex, information-rich data sets is common to virtually all fields of business, science, and engineering. The ability to extract useful knowledge hidden in these data and to act on that knowledge is becoming increasingly important in every day peoples life and work. The entire process of applying a computer - based methodology, including new techniques, for discovering knowledge from data is called data mining Data mining is an iterative process within which progress is defined by discovery, through either automatic or manual methods.
Data mining is most useful in an exploratory analysis scenario in which there are no predetermined notions about what will constitute an “interesting” outcome. Data mining is the search for new, valuable, and nontrivial information in large volumes of data. It is a cooperative effort of humans and computers. Best results are achieved by balancing the knowledge of human experts in describing problems and goals with the search capabilities of computers [1]. Data mining consist of two primary goal *prediction* and *description.* In that form *prediction* involves using some variables or fields in the data set to predict unknown or future values of other variables of interest. In other hand *description* focuses on finding patterns describing the data that can be interpreted by humans. One of data mining greatest strengths is reflected in its wide range of methodologies and techniques that can be applied to a host of problem sets. One of technics that is in use in data mining is clustering. Clustering is based on grouping data according to the character, or to any property that they have in common.
The paper is organized as follows. The second part describes suffix tree clustering algorithm witch is in use of data mining. The third section is explanation of created application. The fourth section presents the conclusion, and the fifth contains a list of references.

2 Suffix Tree Clustering

Suffix tree document model and Suffix Tree Clustering (STC) algorithm first were proposed and use in [2]. STC is a linear time clustering algorithm (linear in the size of the document set), which is based on identifying phrases that are common to groups of documents. A phrase is an ordered sequence of one or more words [3]. STC algorithm is different from the other kind of clustering algorithms. It is a data structure which contains all the suffixes of a given string, so as to run many important string operations more efficiently. This algorithm not treats documents as a collection of words but as a string of words. On that way thus operates using the proximity information between words. STC use suffix tree structure to efficiently identify sets of documents that share common phrases and terms, and uses this information to create clusters and to concisely present their contents to the users. STC meanly includes four logical steps: first, document “cleaning”; secondly, constructing a generalized suffix tree; thirdly, identifying base clusters; the last step is to combine these base clusters into clusters.

2.1 Cleaning

Document pretreatment intend to slim each document. In this step, the string of text representing the content of each document is transformed by using a light stemming algorithm. It is stripping the HTML tags, separator and common stop words, as well as extracting word stems (deleting word prefixes and suffixes and reducing plural to singular). The original string that represents the document is saved, as well as pointers to the beginning of each word in the transformed string to the position taken in the original string [4]. First phase in this process is HTML tag cleaning. In this phase algorithm removing all HTML tags form the document like that is represented on figure (Figure1).

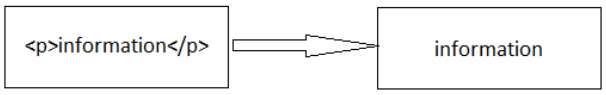


Figure 1: Process of cleaning HTML tags from document

In the next step words stemmers are using. A stemming algorithm is a process of linguistic normalization, in which the variant forms of a word are reduced to a common form. Stemming algorithms can be classified in three groups: truncating methods, statistical methods, and mixed methods. Each of these groups has a typical way of finding the stems of the word variants. Some of stemming algorithm are presented on the figure (Figure 2).

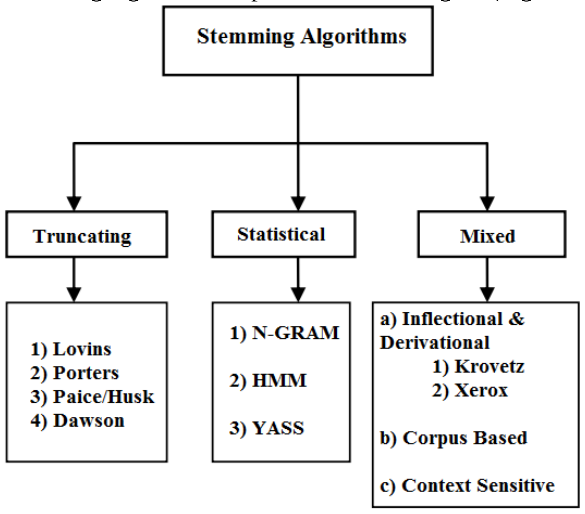


Figure 2: Classification of stemming algorithms

Here we will describe just *Paice/Husk* algorithm because we use this algorithm in our application. This is an iterative algorithm using the same rules and suffixes in every loop [5]. This algorithm consist of one table that containing about 120 rules indexed by the last letter of a suffix. On each iteration, it tries to find an applicable rule by the last character of the word. Each rule specifies either a deletion or replacement of an ending. If there is no such rule, it terminates. It also terminates if a word starts with a vowel and there are only two letters left or if a word starts with a consonant and there are only three characters left [6]. Otherwise, the rule is applied and the process repeats. The advantage is its simple form and every iteration taking care of both deletion and replacement as per the rule applied. The disadvantage is it is a very heavy algorithm and over stemming may occur.

2.2 Constructing generalized suffix tree

In the process of constructing generalized suffix tree the first step is construction of suffix tree. The suffix tree is a data structure which contains all the suffixes of a given string, so as to run many important string operations more efficiently. The string may be a string of characters or string of words [7]. The suffix tree for the string S is defined as a tree such that: the paths from the root to the leaves have a one-to-one relationship with the suffixes of S, all edges are labeled with non-empty strings, all internal nodes (except perhaps the root) have at least two children [7]. In figure (Figure 3) we can see suffix tree for string *"information"*. Here string *information* is represented as string of characters. Documents treats as strings of words, not characters, thus suffixes contain one or more whole words. Construction of generalized (compact) suffix tree going in that way.

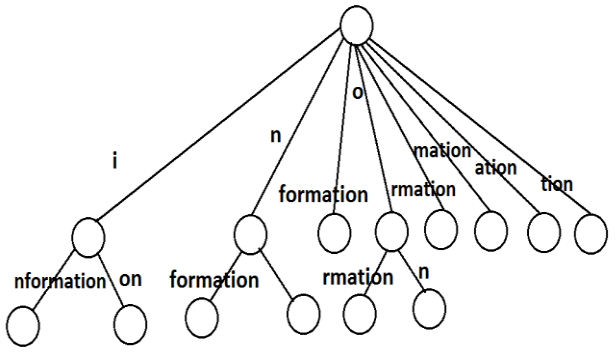


Figure 3: Suffix tree for string "information"

A suffix tree is a rooted, directed tree in which every internal node has at least two children nodes. Every edge in the tree is labeled with a non-empty sub-string of S (hence it is a tree). The label of a node is defined to be the concatenation of the edge-labels on the path from the root to that node [8]. There are no two edges out of the same node can have edge-labels that begin with the same word. This feature makes the tree compact or generalized. Every suffix-node is marked to designate from which string (or strings) it originated from. Maybe the best algorithm for compact suffix tree clustering is Ukkonen's algorithm. His method also builds the tree by the most compact and technical representation, as described previously. Ukkonen's algorithm uses suffix links during the process of building the tree. Each node may have only one suffix link pointing to a node which is nearer to the root node and which has the same subtree, i.e. the same branches, and in general it usually has more branches. More about Ukkonen's algorithm readers can find in [9,10,11]. The example of generalizes suffix tree for strings "cat ate cheese", "mouse ate cheese too" and "cat ate mouse too" is given on figure (Figure 4).

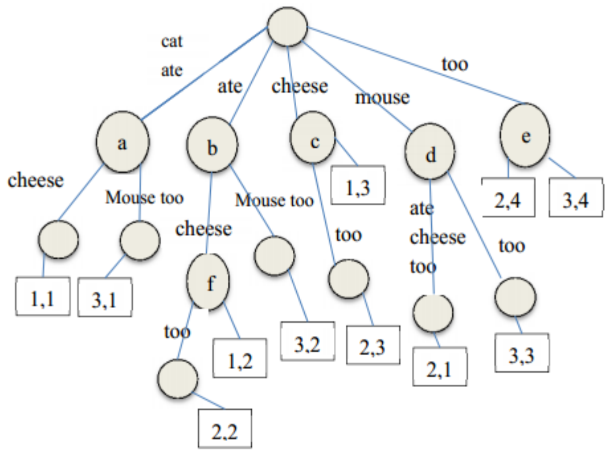


Figure 4: Example of generalized suffix tree for the given strings

In this figure, the circle represents node, the numbers in the square represent document group. Each node of the suffix tree represents a group of documents and a phrase that is common to all of them.

2.3 Identifying Base Clusters

In the generalized suffix tree each node represents a base cluster. To each base cluster is assigned a score s(B) that is a function of the number of documents it contains, and the words that make up its phrase. The function is given in (1):

$$s(B) = |B| \bullet f(|P|) \qquad (1)$$

Here |B| indicates the number of document in a base cluster and |P| is the number of word in a phrase. In the table (Table 1) we can see six nodes from the example shown in figure (Figure 5) and their corresponding base clusters.

Table 1: Representation of nodes and corresponding clusters

Node	Phrase	Documents
a	cat ate	1,3
b	ate	1,2,3
c	cheese	1,2
d	mouse	2,3
e	too	2,3
f	ate cheese	1,2

2.4 Combining base clusters into clusters

In this step, documents may be sharing more than one phrase. To avoid the document overlapping and a nearly identical cluster, this step is assigned to merge base cluster with high overlap in the document set. They defined a binary similarity measure to calculate whether base clusters should be merged or not. The binary similarity will be 1 if the conditions in formulas (2) and (3) are fulfilled.

$$|B_m \cap B_n| / |B_m| \succ 0.5 \qquad (2)$$

$$|B_m \cap B_n| / |B_n| \succ 0.5 \qquad (3)$$

Otherwise their similarity will be defined as 0. This step is presented on the figure (Figure 5). Each cluster consists of the union of the document of all its base clusters. This figure explains the base cluster graph of the six base clusters.

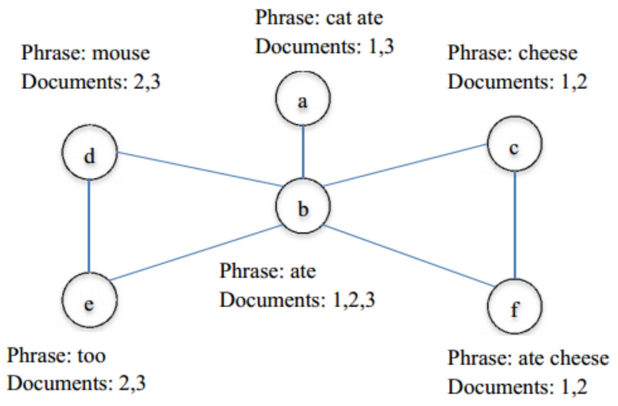


Figure 5: Result of combining based clusters

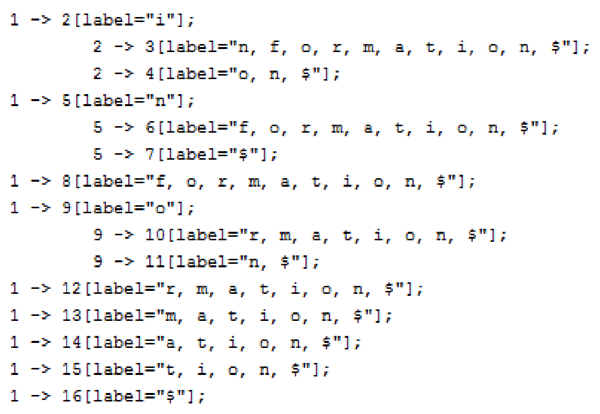
Basically the clustering of basic clusters is performed by single-link algorithm, more accurate its equivalent where the minimum similarity between the based cluster is used as a criterion for the end of the algorithm. The final clusters are sorted according to the number of hits their basic clusters as well as to the mutual overlap of basic clusters. STC clustering algorithm does not require from user to specify the number of clusters. In the next section we will describe created application.

3 Explanation of application for STC

When we planning implementation of suffix tree clustering algorithm we decide to test this algorithm on web page documents and text documents. The reason is simple, we want to test execution time for documents that have *HTML* tags and for documents that do not have html tags. Theoretical execution time for web document should be higher because application first must remove all html tags from document, then stem the plain text in the document, and at in the end create suffix tree and clusters. Text documents don't have *HTML* tags so we expected that the execution time should be shorter.

Application is design in C# environment. Console application is used for interaction with user, and to present the results. We implement two projects in it. First represent *<suffixtree>* with the classes *<Edge>*, *<Node>*, *<Suffix>* and *<Suffixtree>*. Objective of the project is to create suffix tree, generalized suffix tree and clusters. The second project use created suffix tree to test time execution. In this project we first open document that need to be clustered then application clean document content and stem the plain text.

For stemming algorithm we implementing *Paice/Husk* algorithm like we said earlier. After that all text in document is concatenating in one string of words. For given string of words application first create suffix tree, and then generalizes suffix tree. Generalized suffix tree for word *"information"* created and printed in application is presented on figure (Figure6). If you look better the generalized tree created by application have the same structure like theoretical tree on figure (Figure 3).



```
1 -> 2[label="i"];
        2 -> 3[label="n, f, o, r, m, a, t, i, o, n, $"];
        2 -> 4[label="o, n, $"];
1 -> 5[label="n"];
        5 -> 6[label="f, o, r, m, a, t, i, o, n, $"];
        5 -> 7[label="$"];
1 -> 8[label="f, o, r, m, a, t, i, o, n, $"];
1 -> 9[label="o"];
        9 -> 10[label="r, m, a, t, i, o, n, $"];
        9 -> 11[label="n, $"];
1 -> 12[label="r, m, a, t, i, o, n, $"];
1 -> 13[label="m, a, t, i, o, n, $"];
1 -> 14[label="a, t, i, o, n, $"];
1 -> 15[label="t, i, o, n, $"];
1 -> 16[label="$"];
```

Figure 6: Application generalized tree result for string *information*

We measured the execution time for both group of documents. First, we measured creation time of suffix tree and clusters. In the second round we measure search time for some string in the documents. In both cases, the execution is repeated ten times for same example, and for same search string. Average execution time for those ten execution rounds is calculated and presented in the diagram (Diagram 1). Both web and text documents are taken in the way that they have approximately the same size.

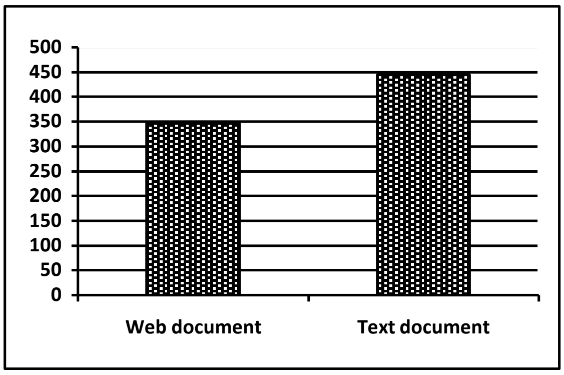


Diagram 1: Average time execution

Both web and text document was approximately about seventy five kilo bytes. In same time web and text document have approximately the same number of words. These documents have five thousand words. Words in web documents was numbered with HTML tags.
Our results are not similar to our expectations. We expect that suffix tree clustering algorithm will give better results for text documents, but not so. We expect that because of time needed for cleaning documents. The cleaning process of *HTML* tags do not last too long, but on the other hand reduces the number of unnecessary words for clustering. Searching time for some random string is about the same.
The search time is in the range of zero milliseconds to the four milliseconds for web documents, and to the max nine milliseconds for text documents. All these parameters show that the STC algorithm is perfect for both text documents and the web documents.

4 Conclusion

Suffix tree clustering is one of the most important algorithm that is in use in the process of clustering. This algorithm have linear complexity O(n).
Linear complexity puts this algorithm in the top of clustering algorithms. Response time in the process of clustering is minimal compared with other clustering algorithms.
This low complexity return fast search in suffix tree. Because of that reasons this algorithm is used in online clustering, and in web document clustering. In the process of searching the first ten results shall be taken as the best. Application created by authors tests other side of suffix tree clustering too.
That other side is pure text documents. Test was build in parallel with web documents. In practice clustering and search using suffix tree clustering is very well. Execution time for text documents is slightly higher than the execution time for the web documents.
Corpus of words that is selected is not so big but in other hand is representative. That corps is quite sufficient for successful testing.
Test result show that suffix tree clustering is good algorithm for all data types, and all documents types. Authors plans new research in this field. Maybe parallel testing with some other clustering algorithm and some improvements of suffix tree clustering.
Some interesting idea may be testing suffix tree clustering on the big knowledge database or parallel execution.

References

[1] M. Kantardzic, “Data mining concepts models methods and algorithms,” John Wiley & Sons, Inc., Hoboken, New Jersey, pp. 5-25, 2011.

[2] O. Zamir, O. Etzioni, “Web Document Clustering: A Feasibility Demonstration,” Proc. ACM SIGIR conference on Research and development in information retrieval, New York, USA, pp. 46-54, 1998.

[3] O. Zamir, O.Etzioni, “A Dynamic Clustering interface to Web search results,” Computer Networks, Netherlands, Amsterdam, 31(11-16):1361-1374, 1999.

[4] D. Sharma, “Stemming Algorithms: A Comparative Study and their Analysis,” International Journal of applied information systems, Foundation of Computer Science FCS, New York, USA, Volume 4, Number 3, pp. 7-12, 2012.

[5] A. Ganesh, “A Comparative Study of Stemming Algorithms” International Journal of Computer Technologies and Application, vol. 2. num. 6, pp.1930-1938, 2012.

[6] R. Chatterjee, “Auto-assemblage for Suffix Tree Clustering,” International Journal of Advanced Research in Computer Engineering & Technology, volume 1, Issue 4, June 2012.

[7] F. Deng, “Web Service Matching based on Semantic Classification,” Master Thesis, School of Health and Society, Department of Computer Science, pp. 9-20, 2012

[8] M. Galaen, “Document klynging (document clustering),” Master of Science in Informatics, Norwegian University of Science and Technology, pp. 19-42, 2008.

[9] C. Kennington, “Application of Suffix Trees as an Implementation Technique for Varied-Length N-gram Language Models,” Master’s Thesis, Saarland University, pp. 9-13, 2011.

[10] E. Ukkonen, “On-line construction of suffix tree,” Algoritmica, 1995.

[11] D. Gusfield, “Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology,” Published by the Press Syndicate of the University of Cambridge, New York, USA, pp. 90-207, 1997.