ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **16**, Number 1, 2013, 81–102

Vector versus Tree Model Representation in Document Clustering[1]

Daniel I. MORARIU, Radu G. CREŢULESCU, Lucian N. VINŢAN

"*Lucian Blaga*" University of Sibiu, Engineering Faculty,
Computer Science and Electrical Engineering Department
E. Cioran Street, No. 4, 550025 Sibiu, ROMANIA
E-mail: {daniel.morariu, radu.kretzulescu, lucian.vintan}@ulbsibiu.ro

Abstract. The most used method for text documents representation is based on words frequency vectors named VSM (Vector Space Model). This representation, based only on words, loses any "word context" information which is found in the document and therefore reduces the learning efficiency in the process of classifying/clustering. In this article we intend to make a comparison between the classical method of document representation and a method called STDM (Suffix Tree Document Model) based on representing documents in the Suffix Tree format. For the second method we proposed a new approach for documents representation by performing suffix tree only for any two documents. This approach is faster and it has lower memory consumption in comparison with the common approaches that represent all documents from data set in the same suffix tree. Also for this method we have proposed a formula for computing the similarity between documents, which improves substantially the clustering quality. This representation method was validated using two clustering algorithms, HAC - Hierarchical Agglomerative Clustering and k-Medoids. In this context we experiment also the stemming influence in the document pre-processing step and highlight the difference between similarity or dissimilarity measures to find "closer" documents.

Key words: Data Mining, Text Clustering, Vector Space Model Representation, Suffix Tree Document Representation, k-Medoids, Hierarchical Agglomerative Clustering.

[1] A previous shorter version of this paper was presented in *"International Conference on Information Science"*, Paris, November 2011, with the title *"Using suffix tree document representation in hierarchical agglomerative clustering"*. The current paper provides significant additional experimental results with the k-Medoids algorithm.

1. Introduction

In recent years there can be observed a significant increase in WEB usage and the improving of the quality and speed of the internet have transformed our society into one that depends heavily on up to date information. Huge amount of data that is generated by this communication process contains important information that is daily accumulated in databases and it is not easy to be retrieved. Data mining domain has emerged as a new way of extracting information and knowledge from databases in order to find relevant patterns, correlations or concepts.

The most common method of representing text documents in the learning algorithms is vector representation, where the documents are represented as vectors of word frequencies [1],[2]. This vector representation combined with operations to eliminate the stop-words and extracting the roots of words (stemming) leads to the elimination of any "mean" that is given by the combination of the words from the sentence. These "meanings" or more adequate "word contexts" can be useful in the learning algorithms to improve the quality of learning.

In this paper we intend to conduct a comparative study of clustering algorithms for improving results by representing documents using the suffix tree document model (STDM), which takes into account the order of words in the sentence rather than, their frequency.

The suffix tree appears in the literature related to the Suffix Tree Clustering (STC) algorithm [7],[8],[9]. In this algorithm the documents are represented as suffixes of phrases in the suffix tree. The nodes from the suffix tree represent the common words of the documents and the leaves are all the words from documents. The researches usually are focused to improve the STC algorithm itself. The most important and often investigated is the score for the suffix tree nodes. This score will be used further to decide the content of clusters. This algorithm itself produces many nodes and many overlaps between clusters so that some documents may appear in several different clusters.

In our study we propose that in the stage of document representation to use the Suffix Tree Document Model (STDM) and then the data represented in this manner will be used in some clustering algorithms. Our idea is to use algorithms that are based on distance matrixes. We have decided to use two different clustering algorithms: a hierarchical algorithm and a k-Medoids (partitional) algorithm. Thus, we will not need to build the STDM model tree for the entire document collection; instead we will build one suffix tree for any two documents from the data set and compute the similarity between them. The advantage of this method is that the resulting tree size is much smaller than the tree for the entire data set.

In this article we propose to examine whether using the STDM model to represent documents in clustering algorithms can improve the clustering's results, compared with using vector representation model (VSM - Vector Space Model). STDM representation regarding the representation of text documents includes besides the words from the document, also the order of words from the phrase. Thus, by default, this model contains some elements of syntax of the document. In other words, although the STDM is a computational representation, by representing order of the words in a

tree, brings us somehow to a closer "ontological" representation of documents. This was the scientific hypothesis on which we started this research. In the STDM model we will represent all the phrases and all the words from the phrases contained in the documents. The nodes in the tree will contain common words or phrases from both documents. The more common nodes the tree has, the greater similarity between documents will be. Additionally, we propose to take into account the number of common words in each node, too.

Sections 2 and 3 contain prerequisites for the main work developed in this approach, presenting the document representation model and clustering algorithms used. In Section 4 we present the methodology used for computing the similarity matrix and our proposed formula. Section 5 presents the experimental framework and section 6 presents the main results of our experiments. Finally the last section debates and concludes on the most important results obtained and proposes some further work.

2. Document representation

2.1. Vector Space Model (VSM)

The most commonly method used for representing text documents is the Vector Space Model (VSM). In this method the text document is represented as a vector of terms with associated frequencies [10], [12]. The term definition is not imposed by the model, but the terms are usually words or word sequences. The chosen words from the entire data sets are considered to be terms and based on this terms a dictionary is created called the vocabulary V. In this vocabulary each term is an independent dimension of a k-dimensional space (vocabulary). Any text document can be represented as a high dimensional vector in this space.

Let $d_i = (t_{i1}, t_{i2}, ..., t_{in})$ the representation of a document where t_{ij} is the weight (frequency) of term j (from the vocabulary V) in document d_i

There are several ways of representing the weights of attributes from the vector [1], [7]. Depending on the chosen representation, some learning algorithms work better or worse. Using the **Nominal representation** – in the input vector the values for the term weights are computed using the formula:

$$TF(d,t) = \frac{n(d,t)}{max_\tau n(d,\tau)} \tag{1}$$

where *n(d, t)* is the number of times that term t occurs in the document d, and the denominator represents the value of the term that commonly mostly occurs in document d, and *TF(d,t)* is the term frequency. The weight can take values between 0 and 1.

2.2. Suffix Tree Data Model STDM

In the Suffix Tree Data Model (STDM) the documents are represented as a suffix tree in which the nodes are the common words from documents and the leaves are

all the words from the documents [5],[8], [9]. In this case the common words are the words that occur in the same order at least in two documents.

Let $d = c_1c_2c_3 \dots c_m$ be a representation of a document as a sequence of words c_i with $i = \overline{1,m}$ and a set S of n documents. The suffix tree for a set S containing n documents, each of n_m length, is a tree that contains a root node and exactly Σn_m leaves.

Rules for building the suffix tree are:

1. Each internal node other than root must have at least two children and each edge leaving a node is labeled with a nonempty substring of m.

2. Any two edges that start from the same node cannot start with the same word.

The following steps are needed for building the suffix tree:

1. All the sentences (phrases) from document are delimitated by adding a special terminator (for example we introduce the character "$");

2. A root node is created noted with "NULL";

3. Extract backward one suffix (one or more words) from sentence (phrase);

4. It is checked if the first word or words from suffix already have an arc that starts from root node:

   - If YES, then the existing arc is used and goes into the next node, called current node. A new node is added starting from the current node only for the uncommon part, if it exists. If there is no uncommon part then in the leaf that was reached it is inserted the identifier for the current document (the end delimitation that was inserted in step 1);
   - If NO, a new arc is inserted and labeled with the current suffix;

5. If there are unexplored suffixes go to step 3 if not we have considered that the tree construction is completed.

Since each node contains at least two children it will contain the common part for at least two suffixes (parts from documents). All branches from the root node to the leaf are one document or a phrase in a document. As two documents have more and more common nodes these documents tend to be more similar.

In Fig. 1 we present an example with three documents and the resulting suffix tree.

D1: document classification is hard $0.

D2: document clustering is hard too $1.

D3: document classification and clustering is hard $2.

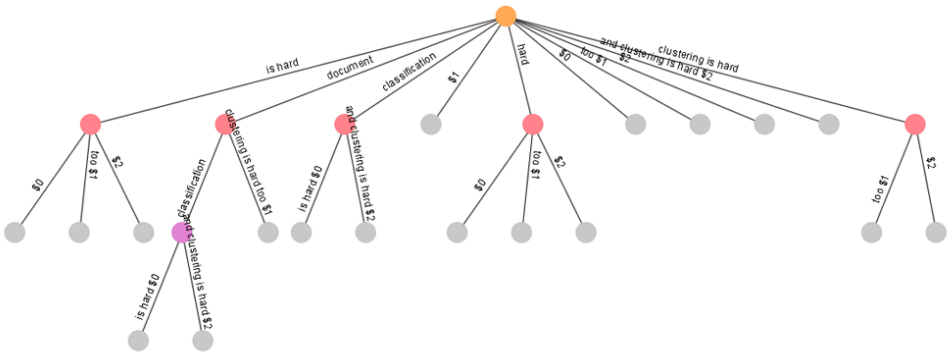

Fig. 1. The suffix tree example.

3. Clustering algorithms

In order to analyse the influence of text document representation for the proposed method we have decided to analyze the learning quality obtained by the clustering algorithms [13], [14]. In this paper we have decided to use two clustering algorithms, the hierarchical algorithm HAC – Agglomerative Clustering Hierarchical and the k-Medoids algorithm (using the PAM version – Partitioning Around Medoids). We will analyze the learning quality obtained by changing the representation methods and also some representation parameters.

In our STDM representation it is not needed to build the tree for the entire collection of documents, such as in the commonly approaches; instead of doing this, we will build one suffix tree for any two documents from the data set. The advantage of this representing method is that the resulting tree size is much smaller than the tree for the entire data set. However this method has the disadvantage to build *n(n-1)/2* small trees, but the construction time required for such a smaller tree is considerably reduced because the trees have few branches and then the search is much faster. Also, this approach is more feasible for implementing it for some parallel computing systems, and making the algorithm to run faster. In addition, at some point, we need to build and search only in one tree and therefore the required memory is much smaller, too. All commonly approaches that build suffix tree for all documents (see algorithm STC) finally use methods for disposing nodes so that the search can be made in a reasonable time. Unfortunately, removal of nodes can lead to loss of useful information. Because our trees are small (containing only two documents), it is not necessary to use any nodes removing methods.

As clustering algorithms we wanted to use two algorithms that are based on the distance matrix that is computed initially and, during running the algorithm this matrix does not change substantially. We chose this approach because in STDM representation is difficult to represent a "medoid" (sometimes it is the center of gravity of all documents from the same cluster) and restore the distance matrix. The selection of clustering algorithms that will be used will take into account the following require-

ments: they must be based on a matrix of distances between any two documents and this matrix should not change substantially throughout the running algorithm.

3.1. Hierarchic Agglomerative Clustering (HAC) single link type

In HAC each document is considered at the beginning as forming a cluster and then in the next level (a more general level) two clusters are joined based on their similarity. All the time it is selected the best similarity from the similarity matrix. After joining two clusters the matrix is recomputed. The algorithm is repeated until all clusters are joined into a single cluster. The similarity (dissimilarity) between two clusters is expressed as the distance between two clusters. There are some methods for computing the distance, for example, the distance can be computed based on Euclidean distance formula or cosine angle between two vectors.

In our approach for computing the similarity between two clusters we use the single link approach where the similarity between two clusters is given by the minimum distance between the most similar two documents contained in those clusters. For example for two clusters A and B with documents a $a \in A$ and documents $b \in B$ then:

$$sim(A, B) = min_{a \in A, b \in B} sim(a, b) \tag{2}$$

For the HAC algorithm we use the AGNES (AGglomerative NESting) implementation that starts with the positioning of each document in its own cluster – thus at beginning the number of clusters is equal to the number of documents – then two clusters are joined based on their similarity at a time until the stop criteria is accomplished or until we get a single cluster containing all documents.

The AGNES algorithm is a bottom-up approach in terms of building clusters. The pseudo code for the AGNES algorithm [11] is presented bellow:

AGNES Algorithm

Input: n objects (vector of documents)

Output: k clusters

Step 1: Note all documents from 1 to n. Each document is a cluster;

Step 2: Compute the distance *d(r, s)* between objects r and s with *r, s = 1, 2,..., n*;

be D = *(d(r, s))* the similarity matrix

Step 3: Find most similar clusters r, s, so *d(r, s)* is minim in D;

Step 4: Join r and s into a single new cluster note t. Compute *d(t, k)* for all clusters $k \neq r, s$. Delete the row and column for clusters r and s from D and add a new column and row for the new cluster t;

Step 5: Repeat from step 3 until the stopping criterion is accomplished

---

The structure returned by this algorithm provides more information than a set of clusters returned by the commonly algorithms. AGNES clustering result is a hierarchical structure in which the more we are on a higher level we obtain the bigger and more general clusters. The result of the algorithm is a dendrogram. The AGNES clustering algorithm does not require specifying, at the start, the number of clusters that are needed to be obtained. The disadvantage of this type of clustering is that it is less effective given the complexity at least quadratic on the number of documents.

3.2. The *k*-Medoids algorithm

The k-Medoids [4],[6] algorithm initially starts with a predefined number of clusters (k) positioned random (called medoids) into the data space representation. In the next step each object is grouped with the medoid that is most similar. After grouping all data for each cluster the algorithm will take all the objects grouped in that cluster and calculates the object which is most representatives for that cluster in order to become the new medoid and it will represent that cluster. The steps are repeated until the medoids no longer change.

Kaufman and Rousseeuw presented in [2], [3] an algorithm, called PAM (Partition Around Medoids) that is an implementation of the k-Medoids algorithm. In the original implementation of the k-Medoid algorithm needs to compute the "center" for all objects from the same cluster in order to find the medoid (the object that is closer to "center"). Using STDM representation of documents the step in which is computed the "center" became more difficult in this algorithm. So that we chose this implementation on k-Medoids that take sequentially all object combinations from the dataset and test if are the best chosen medoids.

PAM Algorithm

Input: n objects (documents), value of k (number of resulted clusters)

Output: k clusters

Step 1: Select randomly k object from data set to become medoids;

Step 2: Take each object (that was not selected as medoid in the step 1) and compute the distance between object and each medoid. The object is grouped in the cluster that corresponds to that medoid for which was obtained the minimum distance;

Step 3: Replace the medoid from the cluster with another object from that cluster; If the medoid is changed go to step 2; otherwise STOP.

Because k-Medoids work directly with medoids is not so much influenced by outliers. However in terms of k-Medoids algorithm for this implementation the computational cost is higher; for a single iteration the computational cost is $O(k(n-k)^2)$. For the k-Medoids algorithm it is needed to be specified, at the beginning of the algorithm, the number of clusters k that should be obtained.

4. The similarity matrix

The selected clustering algorithms require the computation of the *similarity matrix* that contains all the distances between any two documents. To calculate distances between any two distinct documents from the dataset we have chosen two types of measures:

1. Measures that describe the dissimilarity between documents - in this category we implement two distinct measures - called Euclidean Distance and Canberra Distance;
2. Measures that describe the similarity between documents - in this category we implement three type of measures: called Jaccard Distance, OLDST Distance and NEWST Distance;

For the first three distances (Euclidean, Canberra and Jaccard) are associated with the VSM representation of documents and for the last two distances are appropriate with the STDM representation of documents. The first distance category that represents dissimilarity returns results in [0, ∞) domain and we convert them into [0,1] domain.

Euclidean Distance

$$d_{Euc}(x_i, x_j) = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} \tag{3}$$

Canberra Distance

$$d_{Can}(x_i, x_j) = \sum_{k=1}^{p} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|} \tag{4}$$

where x_i, x_j are documents from the dataset and p is the number of elements (words) from vocabulary that represent the documents.

Jaccard Distance

$$d_{Jac}(x_i, x_j) = (1 - \frac{\#common\_elements}{\#total\_elements}) \tag{5}$$

where x_i, x_j are documents from the dataset, $\#common\_elements$ - represent the number of elements (words) that occur in both documents using VSM representation, and $\#total\_elements$ - represents the total number of elements needed for representing the documents x_i and x_j the total number of distinct elements from both documents.

For this metric the domain is [0,1], and because it represents the similarity between documents we decrease the result from 1 so it can be used in the dissimilarity matrix. Also for the following two distances we have converted the similarity to dissimilarity decreasing the result from 1.

OLDST Distance

This distance use the STDM representation of documents and is a most used method for computing similarity in this type of representation. In the document representation using STDM model we build separate one suffix tree for any two documents for the entire data set and we will calculate the distance between them using one of the formulas OLDST and NEWST.

$$d_{OLDST}(x_i, x_j) = (1 - \frac{\#common\_nodes}{\#total\_nodes}) \tag{6}$$

This distance returns the results in the [0,1] domain with 0 most similar.

NEWST Distance

The NEWST distance is proposed by us and it is used with the STDM representation of documents.

$$d_{NEWST}(x_i, y_j) = (1 - \frac{1 + \#common\_nodes}{1 + \#total\_nodes}) * \frac{1}{1 + \#words\_from\_common\_nodes} \tag{7}$$

where $\#common\_nodes$ represents the number of nodes from the suffix tree that are traversed by both documents x_i and x_j; $\#total\_nodes$ represents the total number of nodes (common nodes and uncommon nodes) from the suffix tree; $\#words\_from\_common\_node$ represents the total number of elements (words) belonging to the arcs that are going to a common node. (The arc between two nodes can contain one or more words.) The parameter $\#words\_from\_common\_nodes$ counts all words that are contained by all arcs between the root node and all common nodes from the suffix tree. More precisely, this parameter counts all words that are common in both documents (It represent the length of the commonly shared phrases or word strings).

For this type of representation in case when we have no common nodes the results tends to be 1, reach 1 when the total number of nodes tends to infinity. And the result is 0 when all nodes in the tree are common to both documents.

Our proposed formula (NEWST distance) for computing the similarity matrix has the following advantages in the used clustering algorithms:

1. If two documents have no common nodes then the distance between them depends on the number of nodes that are used to represent those two documents. If the documents are larger and do not have common nodes, the distance between them will be closer to 1 but still different depending on the documents dimension ($\#total\_nodes$). Compared with other classical metrics that always

return the value 1 if the documents have no common nodes (and decide to join all documents in one step), this small difference helps us to determine the order in which documents will be joined based on the distance matrix. Thus in the learning algorithms, if the algorithm will decide to merge documents that have very small number of common words, first it will merge small documents and after that it will merge the larger documents. Theoretically, small documents that have a very small number of common words tend to be more similar than larger documents with the same small number of common words. With the proposed NEWST formula we want to force that the large documents will be merged after the small ones. This is the reason why we decide to introduce de "+1" in the numerator of our proposed formula and consequently we introduce the same parameter at the denominator.

2. If the documents have common nodes we have weighted the returned value with the number of words that are into the common nodes (number of words that are common in both documents). This we can make the difference between documents that have small common parts and documents that have larger common parts.

This two advantages offer us the possibility to make a more accurate clustering.

5. Experimental Framework

5.1. The Dataset

For evaluating the algorithms we used data sets containing news obtained from RSS feeds from Reuters [17] and the BBC Agency's website [15]. The news were selected from several RSS feeds (only XML files), from 3 consecutive days, and have been pre-classified by the agencies. We have selected the following news category: *Libya, Motorsport, Japan, Israel, Syria, Yemen* and *EuropeanFootball*. From xml files we extracted only the $< title >$ node that contains the title of the news and $< description >$ node that contains the body of the news.

Each news was saved in a separate text file, and news from the same category were all saved in the same subdirectory. In the first phase of preprocessing we have removed special characters that appear in the news for reasons of Web interpreter and finally, in text files, on the first line we save the title and the remained lines contain the news - sentences separated by dots. We kept the category already established by news agencies and believe that this division is perfect. We will use this information in the final stage for evaluation the clustering results.

From all of the 193 documents obtained initially grouped into seven categories, we created multiple training sets that differ in the number of categories and the number of documents. Thus were created seven different training sets presented below in Table 1.

Table 1. The structure of datasets

Set Name	No. of categories	No. of documents	Set Structure	
			Category name	No. of documents/category
S01	3	151	EuropeanFootball / Lybia/ Motosport	47/62/42
S02	4	172	EuropeanFootball / Japan / Lybia / Motorsport	38/37/57/40
S03	4	156	EuropeanFootball / Israel / Lybia / Motorsport	47/8/59/42
S04	5	177	EuropeanFootball / Japan / Lybia / Motorsport / Israel	38/37/56/40/6
S05	5	193	EuropeanFootball / Japan / Lybia / Motorsport / Israel	47/37/59/42/8
S06	6	55	Japan / Lybia / Motorsport / Israel / Syria / Yemen	9/12/20/9/3/2
S07	6	179	EuropeanFootball / Japan / Lybia / Motorsport / Israel / Yemen	38/37/56/40/6/2

After the preprocessing step, we have generated two distinct datasets (each having seven sets) as follows: one set where we have eliminated only the stop-words from document and kept all remained words from documents and one set where we eliminated stop-words and extract the roots (stemming) of the remaining words. For root extraction we use the Porter implementation algorithm from the Information Retrieval package developed by the University of Texas [16] that was downloaded in 2007.

5.2. Evaluation of the resulted clusters

Each clustering algorithm applied to the same dataset will group data in different ways, depending on the similarity metric used. This makes analysis of algorithm efficiency very difficult. There are 2 types of quality measures: external measures when there is a prior knowledge about the clusters (we have a pre-labeled data) and internal measures which have no information about clusters (measure as Compactness, Separability, Balance, etc.).

The clustering algorithms receive, as input parameters, the data sets and the number of clusters that must be obtained.

For evaluation of the obtained clusters we have used, as base classes, the category from which the news were extracted. Thus we have used for evaluating the results of the clustering algorithms external measures like accuracy and F-measure score. Those two measures are generally used in the evaluation of clustering and classification algorithms for pre-labeled data. Because in the step of news selection we have chosen only pre-labeled data and the documents from the data sets were saved with the name of the category, we considered to have pre-labeled datasets. We believe that the labeling done by the news agencies sites was “perfect” and we have related to it. Next we have noted “label” the category which was specified by the news agency for the document and “class” the category where the document was included by the clustering algorithm. The name of the class was given by the label that appears most frequently in that cluster. If there was a cluster already labeled with that value then

we have used the next valid label or we have specified "No class" if we do not had available labels.

For evaluation of the clustering algorithms we have used external metrics as Accuracy and F-measure because we have considered it is more precisely. The *Accuracy* represents the percentage of documents that are correctly grouped in categories based on document labels. *F-measure* is a measure that combines precision and recall and is computed as:

$$F - measure(C_i, S_j) = \frac{2 * precision(C_i, S_j) * recall(C_i, S_j)}{precision(C_i, S_j) + recall(C_i, S_j)} \tag{8}$$

where C_i is a cluster and S_j is a label of a cluster. We preferred F-measure score as a measure comparatively with precision or recall because F-measure is a weighted average that includes and balances the influence of two measures (precision and recall).

Precision- is the percentage of retrieved documents that are in fact relevant to a query. *Precision* ranges from 1 (all retrieved documents are relevant) to 0 (none of relevant document is retrieved).

$$precision(C_i, S_j) = \frac{|C_i \bigcap S_j|}{|C_i|} \tag{9}$$

Recall- is the percentage of documents that are relevant to the query and were in fact retrieved. *Recall* range from 1 (all relevant documents are retrieved) to 0 (none of retrieved document is relevant).

$$recall(C_i, S_j) = \frac{|C_i \bigcap S_j|}{S_j} \tag{10}$$

In fact *precision* represents a quantitative measure of the information retrieval system while *recall* represents a qualitative measure of this system. At the final, the general measure F-measure for evaluating a clustering solution is weighted by the relative size of each cluster.

$$F - measure(S) = \sum_i \frac{n_i}{n}(F - measure(C_i, S_j)) \tag{11}$$

where n_i is the number of documents in cluster i and n is the total number of documents in the dataset. The value of *F-measure(S)* belongs to the [0,1] interval; the more the value approaches to 1 means a better result.

While accuracy for current category takes into account only the number of documents correctly grouped in that category, F-measure takes into account in addition the number of documents grouped in categories other than the current category and really not need to be grouped in the current category. Thus F-measure is a finer measure of quality data grouping.

6. Experimental Results

All results that are presented below were obtained using a computer with Intel (R) Core2 Duo T6600 processor at 2.20 GHz, 4 GB DRAM and Windows 7 Home Premium operating system.

The clustering algorithm was implemented in Java using the Eclipse platform. For the preprocessing step we use a set of classes from WEKA 3.7 package [18]. After the preprocessing step the documents were saved into a common format used in the text mining applications called "arff" format.

In the following subsections we will present detailed results on RSS datasets obtained by two clustering algorithms: HAC (Clustering Agglomerative Hierarchical) [11] and k-Medoids (with the PAM implementation - Partitioning Around Medoids). The results are presented separately for the algorithms and for the data representation (VSM-Vector Space Model and STDM-Suffix Tree Document Model). Finally we have presented some comparative results between the two types of representations and the two clustering algorithms used. Also, for each model of representation and each clustering algorithm we have presented results on the dataset without/with applying stemming. For evaluation of the clustering quality we will use accuracy and F-measure score, measures that were presented in Section 5.2. We have used both measures to ensure the validity of the clustering results. Accuracy and F-measure score should reflect basically the same trends. Results for accuracy were presented as percentages and the results for F-measure score are numbers in the interval [0, 1] value of 1 means the best value.

6.1. Results obtained using HAC algorithm

In the Table 2 we have presented all obtained results using HAC algorithm for each dataset separately and in last columns we have presented the average obtained over all datasets. Also we have presented results for the accuracy, with and without applying stemming in the preprocess step and using all metrics for distance, each metric with type of appropriate representation.

Table 2. Results obtained by HAC algorithm – accuracy

Data Representation	Stemming	Metric	Data Sets							
			S01	S02	S03	S04	S05	S06	S07	Average
VSM	No	Jaccard	71.52%	53.80%	66.67%	79.10%	31.09%	65.52%	79.33%	63.86%
	Yes	Jaccard	41.72%	33.92%	39.10%	34.46%	33.68%	68.97%	34.64%	40.93%
	No	Euclidian	41.06%	35.09%	39.74%	34.46%	32.64%	36.21%	34.64%	36.26%
	Yes	Euclidian	43.05%	35.09%	39.74%	34.46%	32.64%	39.66%	34.64%	37.04%
	No	Canberra	42.38%	34.50%	39.74%	33.33%	32.12%	25.86%	32.96%	34.42%
	Yes	Canberra	42.38%	33.92%	39.10%	33.33%	32.12%	25.86%	32.96%	34.24%
STDM	No	OLDST	**100.00%**	**100.00%**	65.38%	**96.05%**	72.02%	70.69%	**96.09%**	85.75%
	Yes	OLDST	**100.00%**	**100.00%**	65.38%	**96.05%**	72.02%	70.69%	**96.09%**	85.75%
	No	NEWST	**100.00%**	99.42%	**95.51%**	77.40%	**76.17%**	**84.48%**	77.65%	**87.23%**
	Yes	NEWST	**100.00%**	99.42%	**95.51%**	77.40%	**76.17%**	**84.48%**	77.65%	**87.23%**

With bold we have marked the better results obtained for each set. As it can be observed the STDM representation obtains better results all the time. From the results shown in Table 2 we have noted that using the STDM model the accuracy of clustering algorithm was not affected by the extraction of words root in the preprocessing step. This is understandable because the extraction of roots didn't lost the word order that somewhat will keep the syntax of document. For the VSM model words root extraction is beneficial only for Euclidean distance and there was a small improving, with only 0.78% in accuracy. For other distances used in the VSM model there was a worsening of 0.18% in Canberra distance and 22.93% in the Jaccard distance. The last one is understandable because the Jaccard distance is based on the number of common elements between the two documents and with roots extraction we have obtained more identical words, thereby losing, in this case, the difference between documents and so that the accuracy suffered.

Table 3. Results obtained by HAC algorithm F-measure

Data Representation	Stemming	Metric	Data Sets S01	S02	S03	S04	S05	S06	S07	Average
VSM	No	Jaccard	0.8078	0.5645	0.7731	0.7600	0.3739	0.3744	0.7627	0.6309
	Yes	Jaccard	0.4735	0.3532	0.4515	0.3416	0.3808	0.6827	0.3379	0.4316
	No	Euclidian	0.4738	0.3393	0.4598	0.3528	0.3812	0.2714	0.3513	0.3757
	Yes	Euclidian	0.4720	0.3622	0.4598	0.3528	0.3812	0.2058	0.3513	0.3693
	No	Canberra	0.4728	0.3491	0.5463	0.3427	0.3816	0.2669	0.3301	0.3842
	Yes	Canberra	0.4728	0.3623	0.4605	0.3427	0.3816	0.2669	0.3301	0.3739
STDM	No	OLDST	**1.0000**	**1.0000**	0.6756	**0.9738**	0.5461	0.6927	0.8129	0.8144
	Yes	OLDST	**1.0000**	**1.0000**	0.6756	0.8581	0.7730	0.6927	0.8129	0.8303
	No	NEWST	**1.0000**	0.9942	**0.9718**	0.8357	0.5222	**0.8576**	**0.8375**	**0.8599**
	Yes	NEWST	**1.0000**	0.8100	0.8084	0.8357	**0.8407**	**0.8576**	**0.8375**	0.8557

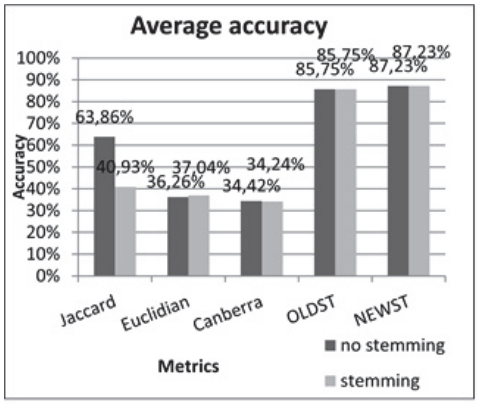

Fig. 2.1 HAC – Average accuracy for all 7 datasets

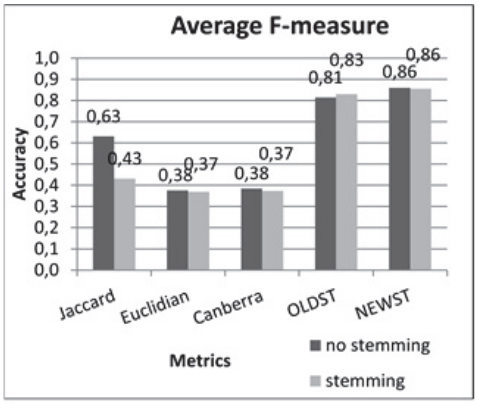

Fig. 2.2 HAC – Average F-measure for all 7 datasets

Fig. 2. HAC Algorithm – Results.

In the Table 3 are presented results obtained from F-measure point of view. As it can be observed we have obtained the best results again for STDM representation; also the NEWST distance has obtained better result for almost all datasets. In case with stemming we have obtained with OLDST metric the results with 0.016 better, and from NEWST metric the difference was insignificant.

We have seen that on the S05 set the HAC algorithm have obtained the weakest results. This can be explained by the fact that this set contains documents that are "outlier" for the HAC algorithm and because this type of algorithm is sensitive to "outlier", we haven't obtained very good results.

The results presented above show that the STDM model used to represent text documents is applicable to other clustering algorithms, than STC. Also an advantage brought by our implementation is that the suffix tree was calculated only for two documents that has reduced time and memory requirements.

Introducing some syntactic elements in document representation - in this case the words order - led to a significant improvement in clustering. We have shown in Fig. 2.1 averages results obtained over all seven data sets as accuracy and in Fig. 2.2 the averages obtained from F-measure point of view for each used metrics. From the results presented in the figures above we can observe that in case of using STDM model the quality of clustering has average improvements of 46.30% in accuracy and 0.4121 for F-measure score.

6.2. Results obtained by *k*-Medoids algorithm

To validate the utility of STDM representation model we have applied the same formulas for calculating the similarity (OLDST and NEWST formulas) in the k-Medoids clustering algorithm (PAM version), that use, like the HAC algorithm, the distance matrix between two documents. In Table 4 are presented results obtained from accuracy point of view, with and without applying stemming in the preprocess step and using all metrics for distance.

Table 4. Results obtained by K-Medoids algorithm – accuracy

Data Representation	Stemming	Metric	Data Sets							
			S01	S02	S03	S04	S05	S06	S07	Average
VSM	No	Jaccard	89.40%	85.96%	71.79%	73.45%	74.61%	63.79%	62.01%	74.43%
	Yes	Jaccard	92.05%	87.72%	**91.03%**	76.84%	77.72%	65.52%	59.78%	78.66%
	No	Euclidian	56.29%	34.50%	56.41%	22.03%	42.49%	53.45%	22.91%	41.15%
	Yes	Euclidian	61.59%	45.03%	50.00%	51.41%	39.90%	51.72%	36.31%	47.99%
	No	Canberra	35.76%	32.75%	31.41%	31.07%	33.16%	41.38%	27.37%	33.27%
	Yes	Canberra	39.07%	32.75%	40.38%	31.07%	39.90%	41.38%	29.05%	36.23%
STDM	No	OLDST	**92.72%**	87.13%	81.41%	46.89%	**78.76%**	72.41%	63.13%	74.64%
	Yes	OLDST	91.39%	70.18%	83.33%	46.89%	77.72%	72.41%	63.13%	72.15%
	No	NEWST	89.40%	**88.89%**	86.54%	**77.97%**	75.13%	**87.93%**	64.25%	81.44%
	Yes	NEWST	88.08%	**88.89%**	86.54%	**77.97%**	75.13%	**87.93%**	**73.74%**	**82.61%**

Note that, as for the HAC algorithm, the STDM model with distances NEWST obtained better results. We have observed that, as in the case of HAC algorithm,

the extraction of root words does not had significantly influence in the STDM model. We have noticed a slight decrease in accuracy even for datasets where the stemming algorithm was applied (2.90% for OLDST and 0.22% for NEWST). The situation was completely changed in case of VSM representation model. In case of k-Medoids algorithm that is a partitional algorithm and the overlap in results is not accepted - we have obtained better results when we have used the root word extraction module. An explanation that the average accuracy decreases on dataset without stemming algorithm in case of using k-Medoids algorithm compared with the HAC algorithm is that the k-Medoids algorithm does not accept overlapping in creating the groups. Using a smaller number of words helped the algorithm to find its separation lines between the created groups. Thus we have obtained an increasing in accuracy of 5.75% in case of Euclidean distance, of 5.31% in case of Jaccard distance and 3.17% in case of Canberra distance.

From the presented results it can be observed that in case of using the similarity measures (OLDST, NEWST and Jaccard) we have obtained better results comparatively with dissimilarity measures (Euclidean and Canberra). This can be explained by the fact that the distances computed, for distance matrix, in case of first category is uniformly distributed in the domain [0, 1]. Distance metrics computed in case of second category can return values closer to 0 for the most cases. When there are two documents in the dataset and the distance between them is very high compared with the other distances this can be a problem of representing the small numbers. Another difference between the metrics OLDST, NEWST and other used metrics was that the OLDST and NEWST metrics use STDM representation of documents having some syntactic information (only words order in the document) compared with other metrics that use VSM representation. The Jaccard distance got better results comparatively with other metrics that use VSM representation model, but got worse results comparatively with metrics that use STDM representation model.

When using k-Medoids algorithm the STDM model representation together with NEWST metric obtained superior results comparatively with the VSM model or Jaccard metric. Improvement of 5.04% for NEWST metrics for k-Medoids algorithm was less impressive than the 34.84% improvement by the metric NEWST for HAC algorithm.

In Table 5 we have presented results obtained from F-measure point of view. As it can be observed we have obtained the best results again for the STDM representation; also the NEWST distance has obtained better results for almost all datasets. We have shown in Fig. 3.1 the average results obtained over all seven data sets as accuracy and in Fig. 3.2 the averages obtained by F-measure for each used metrics using the k-Medoids algorithm.

If we compare the improvements of NEWST with OLDST in case of not using the stemming, they were better with 6.81% comparatively with 10.46% improvement made by NEWST using the stemming algorithm. We can also say that the applying of the stemming algorithm on the data sets was beneficial in case of k-Medoids algorithm. This appeared because the partitional k-Medoids algorithm don't supports overlapping clusters such as the HAC algorithm do, and using fewer attributes allows easier separation of the resulted cluster.

Table 5. Results obtained by k-Medoids algorithm F-measure

Data Representation	stemming	Metric	Data Sets S01	S02	S03	S04	S05	S06	S07	Average
VSM	No	Jaccard	0.9070	0.8746	0.6374	0.6438	0.6437	0.5140	0.5055	0.6751
	Yes	Jaccard	0.9169	0.8826	**0.8792**	**0.6619**	0.6611	0.5153	0.5041	0.7173
	No	Euclidian	0.6218	0.3758	0.6401	0.2391	0.4816	0.4849	0.2282	0.4388
	Yes	Euclidian	0.6687	0.4471	0.4814	0.4926	0.3751	0.5185	0.3405	0.4748
	No	Canberra	0.3702	0.3551	0.2802	0.2796	0.3254	0.4010	0.2131	0.3178
	Yes	Canberra	0.3975	0.3547	0.3753	0.2793	0.3511	0.4012	0.2237	0.3404
STDM	No	OLDST	**0.9288**	0.8708	0.6783	0.4111	**0.6926**	0.7311	0.5105	0.6890
	Yes	OLDST	0.9183	0.7101	0.6981	0.4111	0.6632	0.5466	0.5105	0.6368
	No	NEWST	0.8958	**0.8903**	0.8390	**0.6619**	0.6455	**0.8883**	0.5167	0.7625
	Yes	NEWST	0.8882	**0.8903**	0.8390	**0.6619**	0.6555	**0.8883**	**0.5513**	**0.7678**

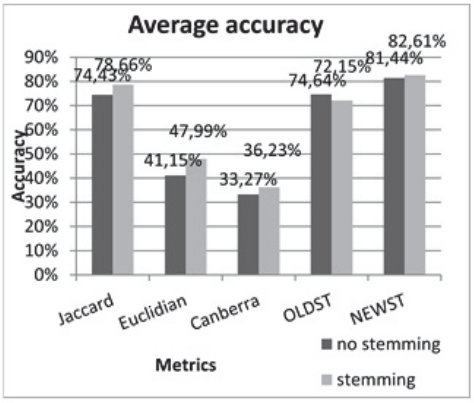

Fig. 3.1 k-Medoids – Average accuracy for all 7 datasets

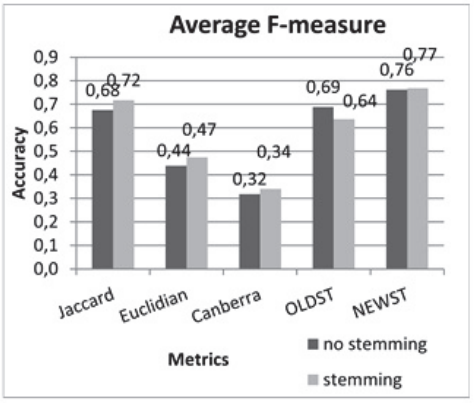

Fig. 3.2 k-Medoids–Average F-measure for all 7 datasets

Fig. 3. k-Medoids Algorithm – Results.

From the presented results we have noted that, when the STDM model was used, the accuracy of the clustering algorithm was not affected by preprocessing data (stemming or not stemming). This is understandable because by extracting words root, the order of words was kept and also something from the document syntax was represented. For VSM model the root extraction was beneficial only for Euclidean distance which obtained a small improving, with only 0.78% in accuracy. For other distances the results were worse: in case of Canberra distance with 0.18% and in case of the Jaccard distance there was a 22.93% worsening. This can be explained by the fact that the Jaccard distance was based on the number of common elements between two sets and when we have extracted the words root we have obtained more identical words, thus losing the difference between documents.

6.3. Comparison between the used representation models and clustering algorithms

The results presented above show that the STDM model used for the text document representation is applicable also to other clustering algorithm not only to the STC algorithm. The essential condition is that the chosen algorithm has to use the distance matrix in training part. The STDM model becomes more difficult to be used in the clustering algorithms that are based on centroids, as k-Means. For the implementation of k-Medoids algorithm we have used the PAM version because this implementation does not require the computing of the center of gravity and chose as medoid the document that is closed to it, which by default led to increased execution time. Entering, by default, some syntactic elements in this case the word order led to a significant improvement in clustering results.

Below we will present comparative results for OLDST and NEWST metrics used with HAC algorithm (NEWST has obtained best results with this algorithm) and k-Medoids algorithm (NEWST having in this case also the best results). The results are presented for each data set and in average.

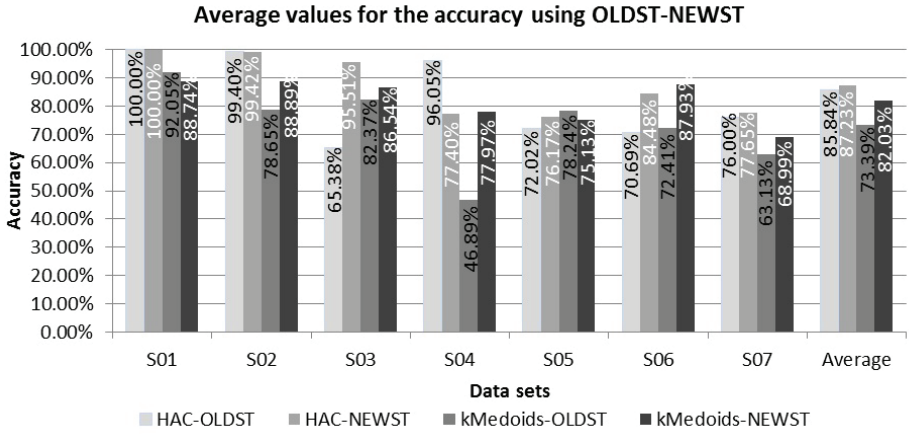

Fig. 4. Comparison from accuracy point of view for NEWST and OLDST in HAC and k-Medoids algorithms.

From Figure 4 it can be observed that the HAC algorithm with NEWST metric and STDM representation get the best accuracy of classification. Only with S06 set, which is a set that contains six classes but few documents, the k-Medoids algorithm got better results because the documents are not "syntactically" overlapped. Figure 5 shows the F-measure score obtained by NEWST in case of using HAC and k-Medoids algorithms.

We can conclude that for the F-measure score the metric NEWST with STDM representation using the HAC algorithm for documents that were "more like" got better results.

In Table 6 are presented the execution time needed for HAC algorithm. The time in the table is expressed in seconds.

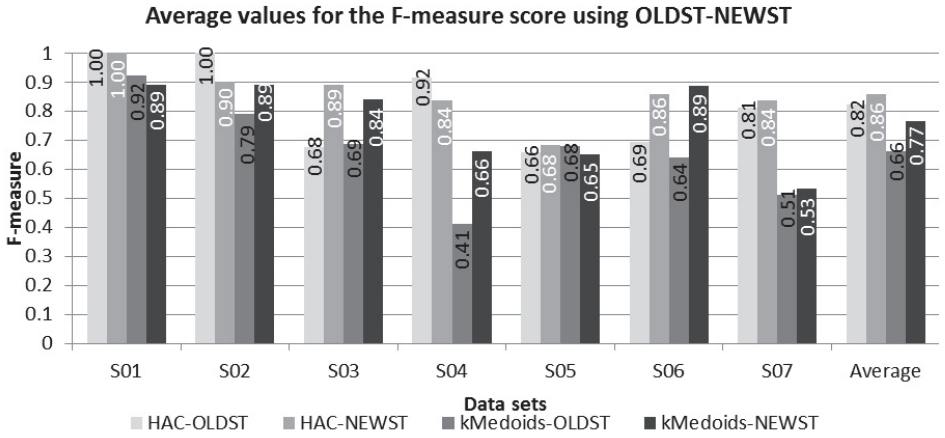

Fig. 5. Comparison from F-measure point of view for NEWST and OLDST in HAC and k-Medoids algorithms.

Table 6. Execution time (seconds) for HAC algorithm

Data Representation	stemming	Metric	Data Sets S01	S02	S03	S04	S05	S06	S07
VSM	No	Jaccard	0.9	1.2	0.9	1.2	1.3	0.5	1.3
	Yes	Jaccard	1.0	1.0	0.9	1.1	1.2	0.3	1.0
	No	Euclidian	4.6	6.2	4.8	6.8	8.5	0.5	6.9
	Yes	Euclidian	4.3	5.7	4.4	5.8	7.2	0.5	6.1
	No	Canberra	1.1	1.4	1.1	1.4	1.8	0.3	1.4
	Yes	Canberra	1.0	1.2	1.0	1.3	1.5	0.3	1.3
STDM	No	OLDST	66.3	87.7	68.3	93.8	116.0	6.6	96.2
	Yes	OLDST	66.7	92.1	68.7	95.8	116.5	6.5	122.3
	No	NEWST	67.9	85.6	67.3	93.5	117.0	6.5	96.1
	Yes	NEWST	66.7	92.6	68.6	97.5	116.9	6.5	146.3

As it can be seen the time values obtained by the algorithm using the Jaccard, Euclidean and Camberra measures with VSM representation was within the range 0.3 to 8.5 seconds. In case of using the OLDST and NEWST metrics with STDM model the time increased significantly, reaching more than two minutes. This was expected since it was necessary to build $n(n-1)/2$ different trees. It can be seen that the execution time for these two distances was influenced by the number of documents rather than the number of resulting clusters. It may be noted in case of set S06, which is a small set: the algorithm obtained very good execution times even if there were six clusters that need to be computed. In this moment, even if the HAC algorithm with STDM representation and NEWST distance obtained the better results in terms of accuracy, unfortunately it required more execution time. An

optimized implementation would substantially reduce the computing time and will be used in online clustering. Further developments should pay attention to this aspect.

For the k-Medoids algorithm, due to the implemented method, the execution time has increased for STDM representation and OLDST or NEWST metrics to approximately ten minutes. For VSM representation we got smaller time values around five minutes. This can be explained by the PAM implementation used for k-Medoids algorithm. Recalling that, in PAM implementation all distances were computed for each medoid separately. In the classical implementation of the k-Medoid algorithm in the first step it is computed the membership of documents for chosen medoids and in the second step the medoids are computed using the center of gravity for all documents from same cluster and it is chosen as medoid the document that is closest to that center. Using this implementation is leading more quickly to the situation when medoids no longer change. But considering the current tendency, where the parallel computing becomes an important issue, the PAM implementation offers a better approach for parallel computing comparing with the classical k-Medoids implementation.

7. Conclusions and Further Work

In this paper we have examined the possibility of using in the clustering algorithms like k-Medoids and HAC the STDM model (Suffix Tree document model) for representation of documents and we have compared the results with results obtained when we have used the VSM (Vector Space Model) representation of documents. For suffix tree model we have developed two formulas for calculating the similarity between two documents, called OLDST and NEWST. For the VSM model we have used for computing the distance between two documents the Euclidean distance and Canberra distance (which are measures of dissimilarity), and Jaccard measure that is a similarity measure.

For evaluation of the results we have used two metrics: the accuracy and the F-measure score. The dataset that we have used comes from specific news feeds from Reuters and BBC agencies and each news item was pre-labeled and we have considered that was "perfect".

The Suffix Tree representation model for the documents proved to be superior to the vector model noticing an improvement on average for 34.84% of HAC algorithm, the maximum recorded is for the S02 dataset where there was an improvement of 55.56%. Also to this result contributed the OLDST and NEWST formulas of similarity used by us. In case of k-Medoids algorithm we have obtained an improvement of 25.63%, the maximum being achieved throughout the set S02 where there was obtained a 30.65% improvement in accuracy. Another interesting observation was that in both clustering algorithm the similarity measures obtained better results comparing with dissimilarity measures.

We have proposed a new metric (called NEWST) used to calculate the similarity between two documents represented in the STDM model. This metric applied to STDM model representation for HAC clustering algorithm obtained on data sets S01-S07 an average improvement of 34.84% from accuracy of clustering point of view com-

pared to Jaccard metric used in VSM representation. Average accuracy on S01-S07 sets calculated for metric NEWST with STDM model representation was of 87.23% comparatively to 52.39% that was obtained with Jaccard metric and VSM representation.

With k-Medoids algorithm the NEWST metric obtained a 5.04% improvement comparatively with the Jaccard metric. The average accuracy for the NEWST with k-Medois algorithm on S01-S07 sets was 84.20% and for the Jaccard metric was 79.16%. In case of k-Medoids algorithm the root word extraction led to better results for all used metrics.

As a global conclusion, HAC algorithm with STDM representation and our proposed NEWST distance obtained the best results in terms of accuracy and F-measure score. As a consequence, unfortunately it required more execution time, but this time should be further reduced through parallel processing on some High Performance Computing Systems.

Our proposed method (STDM) for representing the text data set and compute the similarity matrix has two major advantages:

1. Because we have built separately the suffix tree for any two text documents for data set, this step can be implemented using massive parallel computing methods running on High Performance Computers. We intend to follow this idea for gaining speed in our approach. We expect to obtain substantial improvement through parallel processing because the computing of the similarity matrix is the most time consuming step. After this improvement we can test our approaches on larger data sets.

2. Because in our suffix tree we have included only two documents we didn't need to make any nodes reduction so we had not lose any valuable information.

As further work it would be interesting to change the representation of STDM for clustering algorithms in order to contain more powerful semantic information. Developing some domain ontologies to guide the clustering algorithm would be very useful for improving results.

Acknowledgements This work was partially supported by CNCSIS-UEFISCSU, project number PN II-RU code PD_670/2010.

References

[1] Chakrabarti S., *Mining the Web- Discovering Knowledge from hypertext data*, Morgan Kaufmann Press, 2003.

[2] Kaufman L., Rousseeuw P.J., *Clustering by means of medoids*, Statistical Data Analysis based on the L, Norm, edited by Y. Dodge, Elsevier/North-Holland, Amsterdam, 1987.

[3] Kaufman L., Rousseeuw P.J., *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience, New York (Series in Applied Probability and Statistics), 1990.

[4] Hartigan J.A., Wong M., *Algorithm as 136: A* k*-means clustering algorithm*, Journal of the Royal Statistical Society. Series C (Applied Statistics), London, 1979.

[5] Janruang J., Guha S., *Semantic Suffix Tree Clustering*, Proceedings of 2011 International Conference on Data Engineering and Internet Technology (DEIT 2011), Bali, Indonesia, 2011.

[6] MacQueen J.B., *Some Methods for classification and Analysis of Multivariate Observations*, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, **1**:281–297, 1967.

[7] Manning C., *An Introduction to Information Retrieval*, Cambridge University Press, 2009.

[8] Manning C., Raghavan P., Schütze H., *Introduction to Information Retrieval*, Cambridge University Press, ISBN 978-0-521-86571, 2008.

[9] Meyer S., Stein B., Potthast M., *The Suffix Tree Document Model Revisited*, Proceedings of the I-KNOW 05, 5th International Conference on Knowlegdge Management, Journal of Universal Computer Science, pp. 596–603, Graz, 2005.

[10] Morariu D., *Text Mining Methods based on Support Vector Machine*, MatrixRom, Bucharest, 2008.

[11] Morariu D., Cretulescu R., Vintan L., *Using Suffix Tree Document Representation in Hierarchical Agglomerative Clustering*, International Conference on Intelligent Systems – ICIS Conference, Paris, November 2011.

[12] Salton G., Wong A., Yang C.S., *A vector space model for information retrieval*, Communications of the ACM, **18**(11), 1975.

[13] Zamir O., Etzoni O., *Web Document Clustering: A Feasibility Demonstration*, Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 1998.

[14] Wen H., *Web Snippets Clustering Based on an Improved Suffix Tree Algorithm*, Proceedings of FSKD 2009, Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, China, 14–16 August 2009.

[15] `http://feeds.bbci.co.uk/news/rss.xml`

[16] IR University of Texas, `http://www.cs.utexas.edu/users/mooney/ir-course/Information Retrieval java Application`, accessed in December 2006.

[17] `http://www.reuters.com/tools/rss`

[18] `http://www.cs.waikato.ac.nz/ml/weka/index.html`, accessed in October 2011.