www.czasopisma.pan.pl PAN www.journals.pan.pl
jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2011, VOL. 57, NO. 3, PP. 271–277
Manuscript received June 19, 2011; revised September 2011. DOI: 10.2478/v10177-011-0036-5

Document Clustering – Concepts, Metrics and Algorithms

Tomasz Tarczynski

***Abstract*—Document clustering, which is also refered to as *text clustering*, is a technique of unsupervised document organisation. Text clustering is used to group documents into subsets that consist of texts that are similar to each orher. These subsets are called clusters. Document clustering algorithms are widely used in web searching engines to produce results relevant to a query. An example of practical use of those techniques are Yahoo! hierarchies of documents [1]. Another application of document clustering is browsing which is defined as searching session without well specific goal. The browsing techniques heavily relies on document clustering. In this article we examine the most important concepts related to document clustering. Besides the algorithms we present comprehensive discussion about representation of documents, calculation of similarity between documents and evaluation of clusters quality.**

***Keywords*—Document clustering, text mining, kmeans, hierarchical clustersting, vector space model.**

I. Introduction

DOCUMENT clustering is a special version of data clustering problem. Data clustering is defined as an organization of a set of object into disjoined subsets called clusters (see [2]). The assignment should be made in such a way that objects within a cluster are similar to each other. In text clustering instead of using some general objects we use documents. Clustering algorithms are often used in web search engines to automatically group web pages into categories which can be browsed easily by a user.

The main purpose of using document clustering is to find documents relevant to a given query. To obtain such results a query must be formulated first. It is not always easy to define it. The example of such hard to define query is how to formulate a query if a user wants to retrieve articles describing the most important recent events in the world. Defining the query as *'recent most important events'* probably would not return anything useful, unless there would be an article that is similarly titled, this would also not guarantee the success because this article could be written 10 years ago. In such a case the user does not know exactly what he is looking for. This concept is called browsing. More formally it is defined as a search session without well specified goal (see [3]). This goal is being usually defined during the browsing process. The user's purpose might be also changed during the browsing process. The concept of browsing uses text clustering as a main subroutine of whole process. Those clustering algorithms usually differ a bit from the other ones because they must meet other requirements.

T. Tarczynski is with the Department of Applied Informatics, Warsaw University of Life Sciences, ul. Nowoursynowska 159, 02-767 Warsaw, Poland (e-mail: tomek.tarczynski@gmail.com).

Nowadays document clustering is becoming even more important, mostly because databases containing the documents are growing rapidly. That might cause two kinds of problems. First of them is the time needed to perform a search. Time complexity of most clustering algorithm depends linearly or quadraticaly on the number of documents which makes . The second issue is connected with an accuracy of results. Increase in number of documents increases the probability that document will be assigned to the wrong cluster.

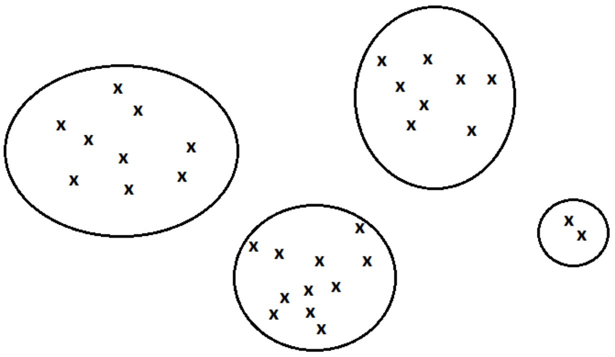

Fig. 1. An illustration of clustering results of objects that can be described using two attributes. Each of the elipses represenst different cluster containing similar objects.

This paper is divided into 6 sections. Section 2 presents the definitions and notations that are used in this paper, while Section 3 presents different representations of documents. Section 4 describes the concept of similarity and evaluation of cluster quality. Section 5, which is the main part of this paper, covers different algorithms used for clustering. Section 6 presents conclutions.

II. Definitions and Notations

The following conventions are used throughout the document. Capital letters are used to denote sets, while the lowercase letters are used for vecors are scalars. Let D be a set of all documents that are to be clustered. It is assumed that this set is always a finite set with cardinality n. Depending on the context d_j can denote the jth document or the vector that represents the jth document (representation of documents is explained in the next section). C is the set of clusters, while C_i refers to the ith cluster. The cardinality of set C is equal to k. Let T be the set of all possible words that can occur in documents. This set is also finite and the cardinality of this set is dentoed by t. It is assumed that the size of this set does not depend on the number of documents. Most of the time this set is referred to as *dictionary*.The ith word in the dictionary is denoted by t_i. The number of occurences of the word t_i in the document d_j is denoted by n_{ij}, while n_j is the total number of words in the jth document. Symbol $||x||$ refers to the length of vector x and $|A|$ denotes the cardinality of the set A. A set of clusters produced during a clustering process is called clustering results.

III. Representation of a Document

Computers process numbers much better than documents and therefore documents are usually represented using numbers. Many different models that represent documents were proposed. The most widespread representation of documents is called *Vector Space Model*, which was presented by Salton, Wong and Yang in [4]. In this model the documents are represented by nonnegative sparse vectors in a t-dimensional space. Such vectors will be denoted by d_j, where j denotes the jth document in the set.

$$d_j = (w_{1j}, w_{2j}, ..., w_{tj}) \tag{1}$$

where w_{ij} is the weigth of ith term in jth document. Each term usually refers to a single word from the dictionary, but some algorithms like *Phrase-Intersection clustering* use phrases instead of single words. In such a case the dictionary would consist of phrases and t would denote the number of possible phrases.

There are several ways of calculating those weights, but typically *term frequency-inverse document frequency* method is used (see [5]). In this method each weight is a normalized product of two components: 'term frequency' and 'inverse document frequency'. Term frequency is calculated as a normalized number of occurrences of the term in the document. The term frequency of the i-th term in the j-th document, T_{ij}, is defined as:

$$T_{ij} = \frac{n_{ij}}{n_j} \tag{2}$$

Term frequency represents how often a word is used in a document, the more often the higher the weight. To distinguish common words, which do not say anything useful in the context of document clustering, from the important words inverse document frequency is used. The more documents contain particular word the lower the value of this term, the number of occurences of this word within single document is not relevant, only whether the word occured or not. This term is calculated as a logarithm of the inverse of a fraction of documents in which this word occurred. For the i-th word, the inverse document frequency, I_i, is defined as:

$$I_i = log\left(\frac{n}{|\{j : t_i \in d_j\}|}\right) \tag{3}$$

There is a common problem associated with similar equations namely division by 0. To overcome this problem two approaches are possible. The idea of the first one is to add 1 to the denominator, but it changes the results. The second one, which is more common, is to assume that words that do not occure in any documents are removed. Inverse term frequency solves a problem with common words, which should not have any influence on the clustering process. If a word occurs in all documents then the nominator and denominator are equal and the value of the whole term is 0. It removes the influence of words such as: 'a', 'the', 'so', 'are', etc. According to [4] the a model without Itf has lower precision and recall[1] by 14% on average. The improvement was measured over the standard term frequency weighting. More about the VSM can be found in [6] and [7].

IV. Document Similarity and Evaluation of Cluster Quality

The first part of this section discusses the document similarity measures. Presented similarity measures assumes that documents are represented using VSM. Similarity measures are one of the key parts of almost every clustering algorithms. They can be used not only to determine the similarity between two documents but also to calculate the smilarity between a document and a cluster and between two clusters. The most popular document similarity measures, according to [8], are *Cosine*, *Dice*, *Jaccard*. They are all based on the dot product, the only difference is the normalization factor. It is worth noticing that if the vectors are normalized then cosine and dice measures produce the same result.

$$Sim_c(d_j, d_q) = \frac{\sum_i w_{ij} \cdot w_{iq}}{\sqrt{\sum_i w_{ij}^2 \cdot \sum_i w_{iq}^2}} \tag{4}$$

$$Sim_d(d_j, d_q) = \frac{2\sum_i w_{ij} \cdot w_{iq}}{\sum_i w_{ij}^2 + \sum_i w_{iq}^2} \tag{5}$$

$$Sim_j(d_j, d_q) = \frac{\sum_i w_{ij} \cdot w_{iq}}{\sum_i w_{ij}^2 + \sum_i w_{iq}^2 - \sum_i w_{ij} \cdot w_{iq}} \tag{6}$$

It is common to calculate document dissimilarity (distance) instead of the similarity. This is a bit different approach to the same problem, because instead of saying how similar the documents are we are saying how different they are. Usually the Euclidian distance is used in this approach (see [9]), although some other metrics like *Manhattan distance* or *Minkowski distance* can be used, but they are not so commonly used as they are in data clustering.

Document similarity measures can be extended to a measure resemblance of all documents in a cluster. The whole group of such measures is called *intra-cluster similarity measures* (see [10]). The most widely used intra-cluster similarity measure is group average similarity. It is defined as an average of pairwise similarity of documents in the cluster, it is given by:

$$d(C_i) = \frac{\sum_{d_j, d_q \in C_i} sim(d_j, d_q)}{|C_i|^2} \tag{7}$$

If we use a cosine as a document similarity measure then somewhat suprisingly this formula can be significantly simplified. To simplify this formula we need to introduce the concept

[1]These terms are explained in the next section.

of a centroid. The centroid of a cluster is a vector that is used to represent the whole cluster, it is defined as an average of all vectors that represent documents in a cluster, in this paper we denote the centroid of the ith cluster by c_i.

$$c_i = \frac{1}{|C_i|} \cdot \sum_{d_j \in C_i} d_j \tag{8}$$

The centroid does not have to have normalized length, in fact it almost never does. It can be easily proved that the intra-cluster similarity is equal to the squared length of a centroid vector.

$$d(C_i) = ||c_i||^2 \tag{9}$$

The calculation of the similarity between two clusters can be simplified to the calculation of the similarity of their centroids. Another approach was presented in [11], where Olsen introduced new clusters similarity measuring metrics: single link, average link, complete link and median. They were defined as the minimum distance between documents in two clusters, an average distance, a maximum distance and a median of distances (see also [3]).

The last part of this section is devoted to external quality measures. In contrast to the previously mentioned measures these ones need an external knowledge about real classes. This assumption prevents from using them in an clustering algorithm as a criterion function. Those measures reflect the similarity between real classes and the ones obtained in the clustering process.

One the most fundamental measure is the entropy, which is based on the termodynamical entropy. The entropy of a single cluster is defined as:

$$E_j = -\sum_i p_{ij} \cdot ln(p_{ij}) \tag{10}$$

Where p_{ij} is the probability that a document which is in the cluster j belongs to the class i. Here we assume that $0 \cdot ln(0) = 0$. To obtain the quality of all clusters the weighted sum of entropies of all clusters must be calculated. The weight is equal to the size of the cluster.

$$E = \frac{1}{n} \sum_i E_i \cdot n_i \tag{11}$$

F measure, was first introduced by C. J. van Rijsberge in [12], is a different external quality measure. To understand this measure two concepts must be introduced. First of them is precision. *Precision(i,j)* can be interpreted as a probability that randomly chosen document from jth cluster belongs to the ith class. The second concept is called *recall* and it can be interpreted as a probability that randomly chosen document from the ith class belongs to the jth cluster.

The F-measure of the cluster j and class i is calculated as harmonic mean of precision and recall, which is given by:

$$F(i,j) = \frac{2 \cdot Recall(i,j) \cdot Precision(i,j)}{Precision(i,j) + Recall(i,j)} \tag{12}$$

The F-measure of whole cluster result is given in the following form

$$F = \frac{1}{n} \sum_i^k \frac{n_i}{\max_j F(i,j)} \tag{13}$$

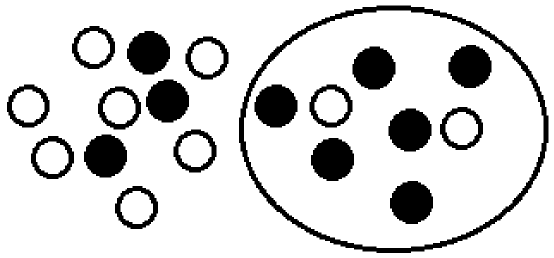

Fig. 2. An illustration of recall and precision. Black dots represent documents that are in one category and the white ones represent documents that are in another category. Dots that are in an ellipse corespond to the documents in one cluster. Recall of this cluster is equal to $\frac{2}{3}$ and the precision is equal to $\frac{3}{4}$.

More about F-measure and other quality measures in infromation retrieval can be found in [13]

V. Clustering Algorithms

This section describes the most important clustering algorithms. Each of presented algorithms fits into one of two categories: *hierarchical clustering* and *non-hierarchical clustering* (see [14]). Non-hierarchical clustering algorithms are often called flat methods or partitional methods. Hierarchical methods in contrast to the partitional methods do not create a single clustering result, but the whole hierarchy of clustering. There are two basic approaches to hierarchical clustering. First of them is hierarchical agglomerative clustering (HAC) and the second one is hierarchical divise clustering (HDC).

A. Hierarchical Agglomerative Algorithms

Hierarchical methods are often believed to produce better clustering results (see [15]), but their main disadvantage is the complexity. The lower bound on time complexity in hierarchical clustering is $O(n^2)$. The results obtained by a hierarchical clustering algorithm can be viewed at different levels. Each of the levels can be seen as separate clustering result with different number of clusters, although those levels are strongly connected because the result on each level is based on the result on previous level. This hierarchy of results is usually graphically presented using a tree. This tree is called dendrogram. It is a tree with a single node at the top and n nodes at the bottom. Each of the nodes that are on the bottom represent a cluster with a single document in it. At the beginning of the agglomerative approach each document is in its own cluster, so in the first level number of clusters is equal to n. In each iteration the two most similar clusters are merged. This process continues until only one cluster is left. That strategy is also called bottom-up approach. Exhaustive survey on agglomerative hierarchical methods can be found in [14]. More about the efficiency of HAC can be found in [16]. The traditional agglomerative algorithm can be summarized as follows:

1) Compute the similarity matrix A, whose entry $[a_{ij}]$ is the similarity between ith and jth cluster.
2) Merge the two most similar clusters.

3) Update the similarity matrix (only the column that represents new cluster must be updated)
4) Repeat steps 2 and 3 until all documents are in one cluster.

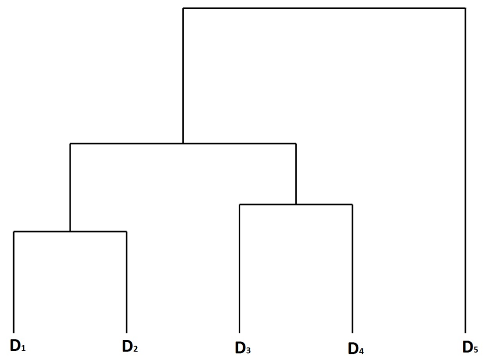

Fig. 3. Sample dendrogram. Looking from the bottom-up approach in the first iteration documents D_1 and D_2 were merged into one cluster, in the second one documents D_3 and D_4 were grouped into one cluster. In the third iteration cluster containing D_1 and D_2 was merged with cluster that contains D_3 and D_4 and finally in the last iteration all documents were put into one cluster.

B. *Bisecting K-means*

Bisecting K-means is a hierarchical divisive algorithm, which was introduced in [15]. Bisecting K-means should not be confused with K-means algorithm which is partitional algorithm. The divisive strategy is in some sense a reverse of the agglomerative technique. At the beginning of the algorithm there is only one cluster which contains all documents. In each iteration a chosen cluster is divided into two clusters. The process stops when each cluster contains only a single document. The bisecting K-means is described below:

1) Choose a cluster to split.
2) Apply K-means clustering with $k = 2$ to this cluster. *(bisecting step)*
3) Repeat the bisecting step *ITER* times and choose the split that produces clustering with the highest overall similarity.
4) Repeat all steps until the desired number of clusters is obtained.

Empirical research in [15] shows that ITER $= 5$ is a good compromise between the quality and the time complexity. The increase of the ITER parameter would increase both the quality and the time complexity. This algorithm is very flexible because steps 1 and 2 can be implemented in many different ways. The most intuitive way to choose a cluster in step 1 is to choose the largest cluster. That was the method used in [15]. The other possible approaches mentioned in [15] are take the cluster with the least overall similarity, use a criterion based on both size and overall similarity. The bisecting step is even more adjustable since any nondeterministic partitional clustering algorithm can be used in this step, although taking K-means seems reasonable because it is a simple and effective algorithm. The K-means algorithm itselft can be also adjusted in many ways, what will be described in the next subsection.

Another huge advantage of this algorithm is that it can be seen as either flat clustering or hierarchical clustering method. Using it as a partitional clustering might even produce better results because the obtained results can be improved using K-means algorithm on the desired number of clusters. In this case the bisecting K-means would just lead to finding centroid for clusters.

C. *K-means*

K-means clustering (see [17], [18]) is the most widely used partitional clustering technique. Its popularity is due to its simplicity and good results. The general description of k-means algorithm is as follows:

1) Choose k – the number of clusters
2) Choose k points as a center of clusters
3) Assign each document to the nearest cluster
4) If stop criterium is met then stop
5) Recalculate center of clusters
6) Go to the step 3

Choosing k in the first step depends strongly on the dataset and motivations that are behind clustering, although some systematic approaches for choosing the number of clusters are given in [19]. The points in the second step are usually chosen at random for simplicity, but some modifications such as k-means++ [20] might be used. A comprehensive survey on different methods for initializing the K-means clustering can be found in [21]. Authors compare there 11 different initializing methods. From those experiments it might be concluded that the best initializing methods are the ones proposed in [22], [23] and [24]. In order to assign a document to the nearest cluster any of the similarity measure mentioned in chapter 3 can be used. The stop criterion is usually defined as a stop when no documents have been moved from one cluster to another. In the standard k-means algorithm, the center of a cluster is calculated as a centroid of the documents in that cluster, but many modifications to this point were proposed. One of them is the k-medians algorithm which calculates in this step the median instead of the mean, this median does not have to be an actual document because median of each term is calculated separately. Another modification was presented as the k-medoid algorithm where one of the documents is chosen to be a center of a cluster. This document is usually chosen as a document with the lowest pairwise dissimilarity.

D. *Algorithms Based on Criterion Functions*

Many partitional clustering algorithms are based on optimization of a global criterion function. In some of these algorithms the criterion function might be exchanged easily. Examples of those algorithms are *CobWeb* [25] and *AutoClass* [26]. Such algorithms consist of two main components. One of them is the criterion function used for an optimization and the second one is the algorithm itself. Many different clustering algorithms can easily be created if a list of criterion functions and optization algorithms are available because every combination of the optimization algorithm and the criterion function creates new clustering algorithm.

In this chapter we will focus mostly on the criterion function instead of algorithms, because according to [9] and [27] greedy strategy produces comparable results to the results obtained using more complicated algorithms. These more complicated algorithms include concepts like iterative schemes based on hill-climbing methods, spectral-based optimizers and so on. The reader more interested in these algorithms are reffered to [28], [29], [30], [31], [32] and [33].

A great survey on criterion functions can be found in [9], Table contating all criterion functions tested in [9] is presented below.

$$f_1 \quad \text{maximize} \quad \sum_{r=1}^{k} n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in C_r} cos(d_i, d_j) \right) \tag{14}$$

$$f_2 \quad \text{maximize} \quad \sum_{r=1}^{k} \sum_{d_i \in C_r} cos(d_i, C_r) \tag{15}$$

$$f_3 \quad \text{minimize} \quad \sum_{r=1}^{k} n_r \cdot cos(c_r, c) \tag{16}$$

$$f_4 \quad \text{maximize} \quad \frac{\sum_{r=1}^{k} n_r \left(\frac{1}{n_r^2} \sum_{d_i, d_j \in C_r} cos(d_i, d_j) \right)}{\sum_{r=1}^{k} n_r \cdot cos(c_r, c)} \tag{17}$$

$$f_5 \quad \text{maximize} \quad \frac{\sum_{r=1}^{k} \sum_{d_i \in C_r} cos(d_i, C_r)}{\sum_{r=1}^{k} n_r \cdot cos(c_r, c)} \tag{18}$$

$$f_6 \quad \text{minimize} \quad \sum_{r=1}^{k} \frac{\text{cut}(S_r, S - S_r)}{\sum_{d_i, d_j in C_r} \cos(c_i, c_j)} \tag{19}$$

$$f_7 \quad \text{minimize} \quad \sum_{r=1}^{k} \frac{\text{cut}(V_r, V - V_r)}{W(V_r)} \tag{20}$$

where c denotes the centroid of all documents, V_r is the set of vertices assigned to the rth cluser. and $W(V_r)$ is the sum of the weights of the adjacenct lists of V_r.

Here we describe an idea that is behind each of these functions:

- f_1 – it is weighted sum of the average pairwise similarities between the documents in each cluster. The weights are normalized due to the size of the cluster.
- f_2 – it is the sum of the cosine between each document and the centroid of the cluster of this document.
- f_3 – it is the sum of the cosine between each centroid of clusters and the centroid of all documents.
- f_4 – it is a quotient of f_1 and f_3.
- f_5 – it is a quotient of f_2 and f_3.
- f_6 – it is edge-cut of each partition scaled by the sum of the cluster's internal edges.
- f_7 – it is normalized edge-cut of the partitioning.

The entropy was used as a measure of quality of the results. Extensive experiments made on 15 different datasets showed that none of these functions is a superior to the others, although some functions performed better. In almost all tested f_2, f_4 and f_5 outperformed all other functions. The difference between the quality of those functions was not significant. It is interesting that some functions, which seem to be very similar like f_1 and f_2, produced very different results. The results varied by up to twenty percent.

E. Word-Intersection Clustering

Oren Zamir in [34] outlined that reducing the number of returned documents not necessarily makes browsing easier for user. He claims that efficient document clustering might be used as a good method for navigating through a large collection of documents, furthermore it might find some patterns that would be normally missed. Zamir also created a 6 point list of requirements for clustering algorithm in context of document browsing.

1) Ease-of-browsing: Results must be presented in such a way that user immediately can decide whether some content is useful or not.
2) Speed: The clustering should be fast enough to create results within seconds.
3) Scalability: The method should be able to cluster large set of documents.
4) No preprocessing: The algorithm should not rely on the preprocessing, because it is applied to dynamically generated set of documents.
5) Client side execution: The system should be able to process the clustering algorithm on the client side. It is motivated by a fact that server might not have enought computation power to produce clustering for each user.
6) Snippet-Capable: The algorithm should not require whole documents, but only small snippets of those documents. Due to client side execution this requirement is necessary, beacuse transfering whole documents could be more time consuming that the algorithms itself.

Word-Intersection Clustering is a hierarchical agglomerative algorithm with a few new concepts. First of them is that clusters are characterized by the set of words that are in every document in the cluster. This automatically solves a problem with the description, because those common words might be used as a description of a cluster. The numbers of words that are shared by all the documents in the cluster is called *cohesion* and it is denoted by $h(C_i)$. Another new concept is the score function $s(c)$, which expresses the decreasing marginal signigicance of the cohesion.

$$s(C_i) = |C_i| \cdot \frac{1 - \exp(-\beta h(C_i))}{1 + \exp(-\beta h(C_i))} \tag{21}$$

The most important new feauture of this algorithm is *Global Quality Function* which quantifies the quality of a clustering. It is defined as:

$$GQF(C) = \frac{f(C)}{g(|C|)} \sum_{C_i \in C} s(C_i) \tag{22}$$

Where the $f(C)$ is a function that is proportional to the normalized number of clusterd documents. A document is considered to be clustered when the size of the cluster that contains is greater than 1, while the $g(|C|)$ is an increasing function in the number of clusters.

The algorithm itself can be summarized as:

1) Put all documents in different clusters.
2) If there are no two clusters whose merge would increse GQF halt.
3) Merge two clusters that icreases GQF the most.
4) Go to the step 2.

The experiments made in [34] showed that this algorithm is faster that the classical hierachical agglomerative algorithm with the cosine similarity measure, which is referred in that paper as *COS-GAVG*. In addition the quality of the results obtained by word-intersection clustering was significantly better than the resuls produced by COS-GAVG. These results prove that this algorithm is very promising.

VI. Conclusions

In this article we have presented the most important concepts related to document clustering. Our research confirmed that there is no superior text clustering algorithm. Document clustering algorithms vary in both the complexity and the quality of results. Hierarchical algorithms in general are considered as the ones that performs better than the flat methods, but on the other hand partitional algorithms are much faster. It seems that the most promising algorithms are hybrid methods like bistecting k-means. This algorithm outperformed partitional clustering algorithms without significant increase in complexity. The clustering algorithm should be chosen depending on the available time, hierarchical algorithms are well suited for a precomputation, while partitional algorithms are good for an online computation.

The complexity of clustering algorithms will play a crucial role in the future, this will be due to continuous increse in the number of ducuments. On the other hand the speed of the computers will increase more slowly because the Moore's law[2] is considered to be broken. This premises lead to the conclusion that parallel and distributed clustering algorithms should be more stressed in the future. Many parallel data clustering algorithms were already designed (see [35], [11], [36]), but not much has been done in the field of parallel document clustering.

Beside the complexity another potential problem is associated with higher demands on the quality. These demands might lead to the development of a more linguistic approach to the document clustering problem. Present algorithms do not take into account an order of the words in a sentence. In some cases the order of words might be relevant in the process of clustering. This task is extremely hard to implement, because each language has its own grammar rules which determine whether the order in particular case is important or not.

References

[1] Y. Labrou and T. Finin, "Yahoo! as an ontology: using yahoo! categories to describe documents," in *Proceedings of the eighth international conference on Information and knowledge management*, ser. CIKM '99. New York, NY, USA: ACM, 1999, pp. 180–187. [Online]. Available: http://doi.acm.org/10.1145/319950.319976

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, September 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[3] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey, "Scatter/gather: a cluster-based approach to browsing large document collections," in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '92, New York, NY, USA, 1992, pp. 318–329. [Online]. Available: http://doi.acm.org/10.1145/133160.133214

[4] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, pp. 613–620, November 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[5] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Cornell University, Ithaca, NY, USA, Tech. Rep., 1987.

[6] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, "On modeling of information retrieval concepts in vector spaces," *ACM Trans. Database Syst.*, vol. 12, pp. 299–321, June 1987. [Online]. Available: http://doi.acm.org/10.1145/22952.22957

[7] X. Tai, M. Sasaki, Y. Tanaka, and K. Kita, "Improvement of vector space information retrieval model based on supervised learning," in *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, ser. IRAL '00, New York, NY, USA, 2000, pp. 69–74. [Online]. Available: http://doi.acm.org/10.1145/355214.355224

[8] G. Salton, Ed., *Automatic text processing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988.

[9] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Mach. Learn.*, vol. 55, pp. 311–331, June 2004. [Online]. Available: http://portal.acm.org/citation.cfm?id=990375.990398

[10] H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma, and J. Ma, "Learning to cluster web search results," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '04, New York, NY, USA, 2004, pp. 210–217. [Online]. Available: http://doi.acm.org/10.1145/1008992.1009030

[11] C. F. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Comput.*, vol. 21, 1995.

[12] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, 1979.

[13] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, "Performance measures for information extraction," in *In Proceedings of DARPA Broadcast News Workshop*, 1999, pp. 249–252.

[14] A. El-Hamdouchi and P. Willett, "Comparison of hierarchic agglomerative clustering methods for document retrieval," *The Computer Journal*, vol. 32, pp. 220–227, jan 1989. [Online]. Available: http://dx.doi.org/10.1093/comjnl/32.3.220

[15] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.1505

[16] W. H. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, pp. 7–24, 1984. [Online]. Available: http://dx.doi.org/10.1007/BF01890115

[17] G. A. Wilkin and X. Huang, "A practical comparison of two k-means clustering algorithms," *BMC Bioinformatics*, vol. 9, p. S19, 2008. [Online]. Available: http://www.biomedcentral.com/1471-2105/9/S6/S19

[18] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09, New York, NY, USA, 2009, pp. 877–886. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557115

[19] M. Chiang and B. Mirkin, "Experiments for the number of clusters in k-means," in *Progress in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Neves, M. Santos, and J. Machado, Eds. Springer Berlin / Heidelberg, 2007, vol. 4874, pp. 395–405. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77002-2_33

[20] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '07, Philadelphia, PA, USA, 2007, pp. 1027–1035. [Online]. Available: http://portal.acm.org/citation.cfm?id=1283383.1283494

[21] R. Maitra, A. Peterson, and A. Ghosh, "A systematic evaluation of different methods for initializing the k-means clustering algorithm," *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[2]Moore's law states that the number of transistors that can be placed on integrated circuits, without any additional costs, doubles every two year

[22] G. W. Milligan and P. D. Isaac, “The validation of four ultrametric clustering algorithms,” *Pattern Recognition*, vol. 12, no. 2, pp. 41–50, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0031320380900011

[23] P. S. Bradley and U. M. Fayyad, “Refining initial points for k-means clustering,” in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML ’98, San Francisco, CA, USA, 1998, pp. 91–99. [Online]. Available: http://portal.acm.org/citation.cfm?id=645527.657466

[24] B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach*. Chapman and Hall/CRC, 2005.

[25] D. H. Fisher, “Knowledge acquisition via incremental conceptual clustering,” *Mach. Learn.*, vol. 2, pp. 139–172, September 1987. [Online]. Available: http://portal.acm.org/citation.cfm?id=639960.639990

[26] P. Cheeseman and J. Stutz, “Advances in knowledge discovery and data mining,” U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, ch. Bayesian classification (AutoClass): theory and results, pp. 153–180. [Online]. Available: http://portal.acm.org/citation.cfm?id=257938.257954

[27] S. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga, “Choosing the cluster to split in bisecting divisive clustering algorithms,” in *Second SIAM International Conference on Data Mining*, 2000.

[28] M. Meila and D. Heckerman, “An experimental comparison of model-based clustering methods,” *Mach. Learn.*, vol. 42, pp. 9–29, January 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=599609.599627

[29] G. Karypis, E. Han, and V. Kumar, “Chameleon: Hierarchical clustering using dynamic modeling,” *Computer*, vol. 32, pp. 68–75, August 1999. [Online]. Available: http://dx.doi.org/10.1109/2.781637

[30] D. Boley, “Principal direction divisive partitioning,” *Data Min. Knowl. Discov.*, vol. 2, pp. 325–344, December 1998. [Online]. Available: http://portal.acm.org/citation.cfm?id=593421.593471

[31] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, “Bipartite graph partitioning and data clustering,” in *Proceedings of the tenth international conference on Information and knowledge management*, ser. CIKM ’01, New York, NY, USA, 2001, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/502585.502591

[32] C. H. Zha, H. Zha, X. He, C. Ding, H. Simon, and M. Gu, “Spectral relaxation for k-means clustering,” in *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 1057–1064.

[33] I. S. Dhillon and D. S. Modha, “Concept decompositions for large sparse text data using clustering,” *Mach. Learn.*, vol. 42, pp. 143–175, January 2001. [Online]. Available: http://portal.acm.org/citation.cfm?id=370660.370699

[34] O. Zamir, O. Etzioni, O. Madani, and R. M. Karp, “Fast and intuitive clustering of web documents,” in *In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 287–290.

[35] M. Dash, S. Petrutiu, and P. Scheuermann, “Efficient parallel hierarchical clustering,” in *International Europar Conference*, 2004.

[36] Y. Song, W. Chen, H. Bai, C. Lin, and E. Chang, “Parallel spectral clustering,” *Machine Learning and Knowledge Discovery in Databases*, pp. 374–389, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87481-2_25

[37] Y. Liu, J. Mostafa, and W. Ke, “A fast online clustering algorithm for scatter/gather browsing,” 2007.

[38] D. R. Cutting, D. R. Karger, and J. O. Pedersen, “Constant interaction-time scatter/gather browsing of very large document collections,” in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR ’93, New York, NY, USA, 1993, pp. 126–134. [Online]. Available: http://doi.acm.org/10.1145/160688.160706