Iterative Text Clustering of Search Results

Gábor Szűcs and Zoltán Móczár

***Abstract*—** Search results clustering, which clusters returned documents, is the most preferred approach for re-organizing search results. This paper deals with text clustering problem, namely many parameters influence the inner operation of the well-known clustering algorithms, but an average user is not able to set these parameters accordingly. In this paper a novel approach is proposed, which solves this problem by an iterative method based on the user feedback. In our solution some questions are generated for the user in order to offer the most appropriate result selected from the possible ones. We have developed a complex clustering algorithm, which automatically optimizes the parameter values based on the user answers. Our method calculates some possible results, which are probably one of the best results from the user's point of view. The main purpose of applying the user feedback is to give possibility to interact the search results and to achieve a better topic understanding about the given query.

***Keywords*—** query, similarity**,** text clustering, user interaction, web search results

I. Introduction

Search results clustering, which clusters returned documents, is the most preferred approach for re-organizing search results. By finding natural groupings of the documents in the result, document clustering methods discovers hidden relationships between documents. Groups of documents consist a hierarchy which gives relationships between subtopics. By an abstract query the users can focus on a general topic and take more knowledge about the topic of his/her interest by drilling down to a specific cluster. Many commercial search engines succeeded to compose a better presentation method for search results by clustering them. Search results clustering is known to provide useful information to users without much domain knowledge [23].

Search result clustering provides an intuitive overview toward information contained in the search result. Document clustering engines have been summarizing documents in a cluster or extracting phrases shared by them to label the clusters. Commercial clustering engines including Vivisimo (www.vivisimo.com), WebClust (www.webclust.com), Fluster (www.funnelback.com) provides friendlier user experience by clustering the search results. By applying document clustering methods to web snippets returned by search engines, clustering engines gave a better topic understanding about the given query.

Time is an important dimension of any information space and can be very useful in information retrieval and in particular clustering and exploration of search results. In the paper [22] an add-on has been presented to traditional information retrieval applications in which various temporal information associated with documents has been exploited to present and cluster documents along timelines. Temporal information expressed in the form of, e.g., date and time tokens or temporal references, appear in documents as part of the textual context or metadata. It has been shown how such a time-based document clustering can be achieved when temporal expressions in documents are readily available. A cluster algorithm provides great flexibility and allows users to explore clusters of search result documents that are organized along well-defined timelines, supporting different levels of time granularity. The use of time and timelines for clustering and browsing nicely fits exploratory search systems that go beyond simply returning some documents or answer in response to a query.

The primary purpose of the development of a new algorithm was to find a novel solution for the common problem [1] of the (i) most well-known clustering algorithms (k-means [2], variation of k-means algorithm [3], SOM [4], hierarchical algorithm [5]) and (ii) new approaches (genetic algorithm [6], frequent itemsets [7][17], comparison of clustering algorithms [8]) in the literature. A lot of parameters influence the inner operation of the algorithms, and an average user is not able to set these parameters accordingly since he does not know how they should be changed to carry out the best results. In our approach this issue can be solved iteratively, in which the users can see the clusters (created by the clustering method) after each modification step, but this may take a long time. Hence, the aim is to optimize the parameters of the clustering algorithm for the user without looking at the results of each iterative step. The goal of our work is that the user does not have to deal with parameter tuning.

Text clustering is most commonly treated as a fully automated task without user feedback. However, some researchers [19] have explored mixed-initiative clustering methods which allow a user to interact with and advise the clustering algorithm. This mixed-initiative approach is especially attractive for text clustering tasks where the user is

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

G. Szűcs is with Inter-University Centre for Telecommunications and Informatics H-4028 Kassai út 26., Debrecen, Hungary and the Department of Telecommunications and Media Informatics, BME, Hungary (e-mail: szucs@tmit.bme.hu).

Z. Móczár is PhD student with Department of Telecommunications and Media Informatics, BME, Hungary.

trying to organize a corpus of documents into clusters for some particular purpose.

Document clustering engines will gain more accuracy by taking user feedback information into account, along with document features. Moreover, user feedback information can be used to measure similarity between web documents that have dynamically changing contents.

Semi-supervised clustering [21] is between the extremes of totally unsupervised clustering and totally supervised learning. The main goal of semi-supervised clustering is to allow a user to control the clustering process. The secondary goal of semi-supervised clustering is to give the human a way to play and interact with data to better understanding.

Existing methods for semi-supervised clustering fall into two general approaches [20] that we call search-based and similarity-based methods. In search-based approaches, the clustering algorithm itself is modified so that user-provided labels or constraints are used to bias the search for an appropriate partitioning. In similarity-based approaches, an existing clustering algorithm that uses a similarity metric is employed; however, the similarity metric is first trained to satisfy the labels or constraints in the supervised data.

In the work [23] a search results clustering engine has been implemented for general objects. The proposed clustering method uses users' feedback information to compute similarity between objects. This clustering engine provides search result clustering for various search tasks retrieving items, or objects whose contents do not contain descriptive text.

But our method focuses only text documents. In this paper our goal is to allow a user to steer the clustering process with minimum time and human effort. The paper deals with document clustering problem where documents can be different kinds of text: web pages or simple text files. The task consists of two phases: text pre-processing and unsupervised learning. In text pre-processing phase we used a general procedure can be found in the literature [13][14], and for clustering we worked out a new algorithm instead of using available ones.

In this paper we present a complex clustering algorithm, which automatically optimizes the parameter values based on the user decisions. The parameters are estimated and our method calculates some possible results, which are probably one of the best results from the user's point of view. The software generates some questions for the user in order to offer the most appropriate result selected from the possible results based on the answers. The main purpose of applying the user feedback is to achieve a significant improvement in the accuracy of the clustering.

II. Clustering algorithm

A. *Text pre-processing for clustering*

Before clustering a text pre-processing and indexing phase should be executed. The pre-processing flow consists of five steps from text segmentation to stemming:

- segmentation: this step deals with separation of the continuous text into structural segments;
- dividing to sentences: this step divides these segments into sentences;
- dividing to tokens: this step divides the sentences into character series so called tokens;
- stop words filtering: this step is an important in the speech style classification since a lot of common words are the same in all speech styles;
- the last step is the stemming, i.e. reduction of words to their roots.

Excluding numbers, certain characters or sequences of characters can be done before the indexing of input documents starts. Furthermore, stop words, i.e. terms that are to be excluded from the indexing can be defined. Typically, a default list of English stop words includes "the", "a", "of", "since", and many other words that are used in the respective language very frequently, but communicate very little unique information about the contents of the document.

Stemming is an important pre-processing step before the indexing of input documents. The term "stemming" refers to the reduction of words to their roots so that, for example, different grammatical forms or declinations of verbs are identified and indexed as the same word. At the end of the pre-processing there is a cleaned text with stemmed words available for each original document, and the list of all these stemmed words (so called terms) in the set of documents will be a vocabulary.

We used vector space model for data representation [9]. In this model every document is represented by a feature vector of the terms (words) that appear in the document collection. Let D = (D1, D2, …, Dn) denote the collection of documents where n is the number of documents can be found in the collection. The feature vector should include proper features to represent the object. Assignment of weights to documents may improve the clustering solution [10]. The weights (usually term frequencies) of the terms are also contained in each feature vector. Term weighting is a process of calculating the degree of relationship (or association) between a term and a document. An advanced and widely used term weighting scheme is the TF–IDF (Term Frequency – Inverse Document Frequency), which can be defined in the following way [9]:

$$w_{ij} = tf_{ij} \cdot idf_j = tf_{ij} \cdot \log\left(\frac{n}{n_j}\right) \tag{1}$$

where tf_{ij} gives the term frequency (i.e. denotes the ratio of the term j in document D_i), and n_j is the number of documents in which the term j appears. We have implemented and compared other weighting schemes – investigated in the literature [15][16] – as well, but in this paper we have focused the core of the clustering algorithm. We have not used dimension reduction techniques, like in Tang's paper [18].

B. *The core of the clustering algorithm*

The input of the clustering algorithm is a document set (D) where the number of documents (n) must be at least three. The task is to cluster the text documents given by the user based on their topics at a threshold t. This threshold is a similarity value, and the similarity between all pairs of documents must be greater than or equal to it. Let us denote C the set of clusters during the algorithm, which is empty at the beginning of the procedure. Let I be the set of document indexes, i.e. $I = \{1, 2, \ldots, n\}$.

Furthermore, we have a similarity function $s(i, j) = s_{ij}$ that gives the similarity value (falling in the interval $[0\ ,\ 1]$) between the documents d_i and d_j. Accordingly, the similarity matrix contains the similarity values of all document pairs. The similarity between two documents is computed with one of several similarity measures based on two corresponding feature vectors, e.g. cosine measure, Jaccard measure and Euclidean distance [9][11].

The core of the algorithm consists of six steps:

1. *Determination of the index of the reference item*

The reference item plays a role in the order of the placement of documents into different groups (clusters). Let us denote cs_j the sum of the column j, that is

$$cs_j = \sum_{i=1}^{n} s_{ij} \qquad j = 1, 2, \ldots, n \tag{2}$$

and the index of the reference item is $r = \text{argmax}\{cs_j\}$.

2. *Creating a new cluster*

The new cluster (c_1) will contain the reference item, and the set of clusters will be extended by this new cluster:

$$c_1 := \{r\} \text{ and } C := C \cup \{c_1\} \tag{3}$$

3. *Construction of a list from document indexes*

Let us construct a list L from the document indexes, which contains all indexes except the index of the reference item.

4. *Ordering of the list*

Let us sort the document indexes of L in decreasing order based on the similarity value between the given document and reference item. The ordered list will be: $l_1, l_2, \ldots\ l_{|L|-1}$. After the ordering (if the number of items in the list is greater than one) $s(r, l_i) > s(r, l_{i+1})$ is true for all i where $i = 1, 2, \ldots, |L|-1$.

5. *Calculation of the average similarity*

In this step the algorithm calculates the average similarity for all items in the list:

$$a_{ij} = \sum_{k=1}^{|c_j|} \frac{s(l_i, c_{jk})}{|\,c_j\,|} \tag{4}$$

where a_{ij} is the average similarity of document i in the cluster j, c_{jk} is the document k of the cluster j, and j runs from 1 to the number of clusters. Let M be the maximal value of a_{ij} ($M = \max_j\{a_{ij}\}$) and $m = \text{argmax}_j\{a_{ij}\}$. If $M > t$, i.e. the largest average similarity is greater than the similarity threshold, then the document l_i will get to the cluster c_m otherwise the algorithm will create a new cluster $c_{|C|+1}$, and the document l_i will be placed in this new cluster. Furthermore, the algorithm leaves the loop, the examined items will be removed from the list, and the reference item will be changed to $r = l_i$.

Formally:
if $M > t$ then $c_m := c_m \cup \{l_i\}$
otherwise $c_{|C|+1} := \{l_i\}$ and $C := C \cup c_{|C|+1}$,
$L := L \setminus \{l_i\}$
where i denotes the index of the currently examined document.

6. *End of the loop*

Repeat steps 4 and 5 until all items will be placed. The cluster set C will be the result of the algorithm at the threshold t.

C. *Clustering with different similarity thresholds*

Based on the clustering core described above a system was implemented where the end user can reach the functionality of the clustering by a graphical user interface (GUI). On the GUI the user can set the similarity threshold and can execute the clustering algorithm by the given value, or can choose the automatic clustering procedure. In case of automatic clustering the core algorithm runs at different thresholds from zero to hundred percent in little steps. The step unit should be a small value, but in order to avoid large calculation efforts later, this unit could not be too little; in our implementation the step unit was one percent. In this case the results of the clustering will be $R = \{R_1, R_2, \ldots, R_{100}\}$ where each R_i represents the generated clusters at the end of the core algorithm and R is the result set. After that the purpose is to find the most appropriate result R_i for the end user.

D. *Definition and calculation of the concentration measurement*

To select the most adequate result the algorithm calculates a concentration measurement for each result R_i, then estimates an interval between zero and hundred percent for the end user, in which the best results can be found with high probability. Then the algorithm orders the results based on their concentration measurements (the highest probability results will be found at the beginning of the list).

Let γ_i be the concentration measurement of the result R_i:

$$\gamma_i = \frac{|R_i|}{n} \sum_{k=1}^{\max_j \{|R_{ij}|\}} I_{\{n_k > 0\}} (k - n_k)^2 \tag{5}$$

where $I_{\{n_k>0\}} = 1$ if $n_k > 0$ otherwise $I_{\{n_k>0\}} = 0$, R_{ij} is the cluster j of the result R_i, n_k is the number of clusters containing k documents, and n is the number of items. This measurement helps to find the optimal result, so the results with a lot of small clusters or a very few large clusters are not preferred. At the compromised solution when we get small concentration measurement value, both of the requirements will be almost satisfactory.

Table 1 shows an example for the calculation of the concentration measurement. It can be seen that the smallest value is zero. In other cases where a lot of small clusters or a very few large clusters can be found, we get higher concentration measurement values. Our tests presented in the next section confirmed that the heuristic concentration measurement formula is an excellent indicator for showing the goodness of the clustering output.

Table 1. Calculation of the concentration measurement (example)

ID	Sizes of the clusters	Concentration measurement
1	1, 1, 1, 1	$\frac{4}{4}(1-4)^2 = 9$
2	1, 3	$\frac{2}{4}[(1-1)^2 + (3-1)^2] = 2$
3	1, 1, 2	$\frac{3}{4}[(1-2)^2 + (2-1)^2] = 1.5$
4	2, 2	$\frac{2}{4}(2-2)^2 = 0$
5	4	$\frac{1}{4}(4-1)^2 = 2.25$

E. *Proper interval estimation*

The goal of the interval estimation is to find the interval (so called proper interval) in the range of the threshold t (between 0 and 1), in which the most appropriate results can be found with high probability. Let n_d be the number of items located below the diagonal of the similarity matrix, and a_d is the average of these items. The a_d is calculated by the following equation:

$$a_d = \frac{1}{n_d} \sum_{\forall j<i} s_{ij} \tag{6}$$

where $n_d = n\,(n-1) / 2$. Let n_t be the number of items where $s_{ij} > a_d$ is true, and n_b the number of items where $s_{ij} \le a_d$ can be satisfied. Furthermore, the average of the difference between a_d and the items greater than a_d (less than or equal to a_d) is denoted by a_t (and a_b, respectively) as can be seen in the next equations:

$$a_t = \frac{1}{n_t} \sum_{\forall i \forall j : s_{ij} > a_d} | a_d - s_{ij} | \tag{7}$$

$$a_b = \frac{1}{n_b} \sum_{\forall i \forall j : s_{ij} \le a_d} | a_d - s_{ij} | \tag{8}$$

Based on these notations the algorithm determines the upper and lower limits of the proper interval in the range of the threshold t by the following way:

$$i_t = \min\{100, \lfloor 100(a_d + c \cdot a_t) \rfloor\} \tag{9}$$

$$i_b = \max\{1, \lceil 100(a_d - c \cdot a_b) \rceil\} \tag{10}$$

where c is a constant (in our experience c = 0.2 was an appropriate value). We considered some different aspects for the determination of the value c. If the parameter is well-chosen, (i) the minimal number of questions is generated in the user interaction phase and (ii) the best results will be fallen in the proper interval with a very high probability.

F. *Selection of the potential results*

The results R_1, R_2, …, R_{100} and the proper interval with limits (i_b and i_t where $i_b \le i_t$) are calculated as described above, thus the most appropriate results can be found among the results R_{ib}, …, R_{it}. The next task of the algorithm is the filtering where the redundancy is reduced: from the same results only one result is remained, others are eliminated. In the next step the algorithm orders these filtered results in increasing order based on the concentration measurements ($\gamma_1 \le \gamma_2 \le \ldots \le \gamma_q$), and we get the potential results P_1, P_2, …, P_q where q denotes the number of potential results.

G. *User interaction*

The algorithm examines the potential results P_1, P_2, …, P_q produced at the end of the previous phase. The software takes questions, and based on the answers it can choose the best potential result.

In order to explore the differences, the algorithm creates a list – for each potential result – from the item pairs, which are in the same cluster. Since each potential result is unique, therefore there is at least one difference between two potential results. The program finds these differences, then generates questions for the users. The form of the questions can be seen in Table 2 where [*x*, *y*] represents that these items can be found in the same cluster while *x*],[*y* indicates the opposite one.

Table 2. Questions generated by the algorithm (example)

Order	Question
1	[2,6] and 3],[4 ?
2	[3,4] ?
3	[1,2] ?
4	1],[2 ?

The user should answer these questions in the given order. In the case when the user's answer is yes, the program will not take further questions, and the potential result related to the corresponding question will be the final output for the end user.

III. Testing of the implemented clustering algorithm

A. Test bed

In order to verify the effectiveness of our algorithm we performed many tests based on an online thematic article database (*articledashboard.com*), which provides one label for each article category. More precisely, we made 50 unique test sets each of them composed of 100 documents. Within a test set the number of different article categories was chosen from the interval 6 and 14. At the same time, the article categories and the number of documents related to a particular category were determined randomly.

Since the clustering algorithm can give heterogeneous clusters as result, a cluster can contain documents from different article categories (henceforward the set of documents related to the same category in a cluster is called subgroup). Consequently, the first issue was to carry out a mechanism that can determine the category of a cluster. It is not easy, because there can be some subgroups within a cluster and we have to choose the most appropriate one, which describes the topic of the cluster. To solve this problem, our method assigns a priority value to each subgroup as follows (*i* and *j* represent the identifiers of the cluster and the given subgroup, respectively):

$$p_{ij} = |t|\left(\frac{1}{|c|} + \frac{1}{10|g|c_{\max}}\right) \quad (11)$$

where *t* is a set of documents related to the given subgroup, *c* is a set composed of the documents found in the examined cluster, *g* is a set of documents containing all of the articles related to the given subgroup by the original labelling and $c_{\max}$ denotes the size of the largest cluster created by the clustering algorithm. The constant $10c_{max}$ is used to refine the ranking of possible categories in case of a particular cluster if the method assigns the same priority value to two or more subgroups. It can be clearly seen that a priority value depends on the subgroup size in proportion to the cluster size and the number of documents related to the same category. A higher priority value increases the probability of assigning the given subgroup to the cluster.

To compute the accuracy of the clustering the method identifies the subgroups of each cluster and calculates their priority values, then creates a list of subgroups for each cluster sorted in descending order based on the priority values. After that constructs a new list of the first elements of these lists and arranges it in descending order. Thus, we have a list, which contains the "highest priority – subgroup" pairs for each cluster. The method scans this list item by item and in every step it checks whether the category of the examined cluster is determined. If it is not determined the currently examined category identified by the subgroup is assigned to this cluster, otherwise the method continues the scanning without any modification. After scanning is finished new lists are constructed (as described above), which contain the second, third, etc. elements of the lists related to the clusters. Finally, the method gives a list of "cluster – category" pairs, and the accuracy can be calculated in the following way:

$$r = 1 - \frac{|d_e|}{|D|} \quad (12)$$

where d_e is the set of misclassified documents. In other words, the classification is right if the given document is placed in the cluster, which has the same category.

B. Results

Figure 1 shows the accuracy of the clustering for each test set. In this interpretation accuracy means the accuracy of the result selected by the user. It can be observed in the figure that the average accuracy was near to 98%, and the result was perfect (100% accuracy) for almost one third of tests.

Figure 2 indicates the highest, lowest and average accuracy values related to the potential results. As can be seen in this figure the average accuracy falls in the interval 82% and 100% while the mean value is 95%. Thus, we can conclude that the algorithm produced a relatively high accuracy for all of the potential results.

Figure 3 depicts the accuracy values in a pie chart split into intervals. In this figure we can see that the clustering algorithm produced at least 96% accuracy for more than 80% of test sets, and the lowest accuracy value was greater than 86%.

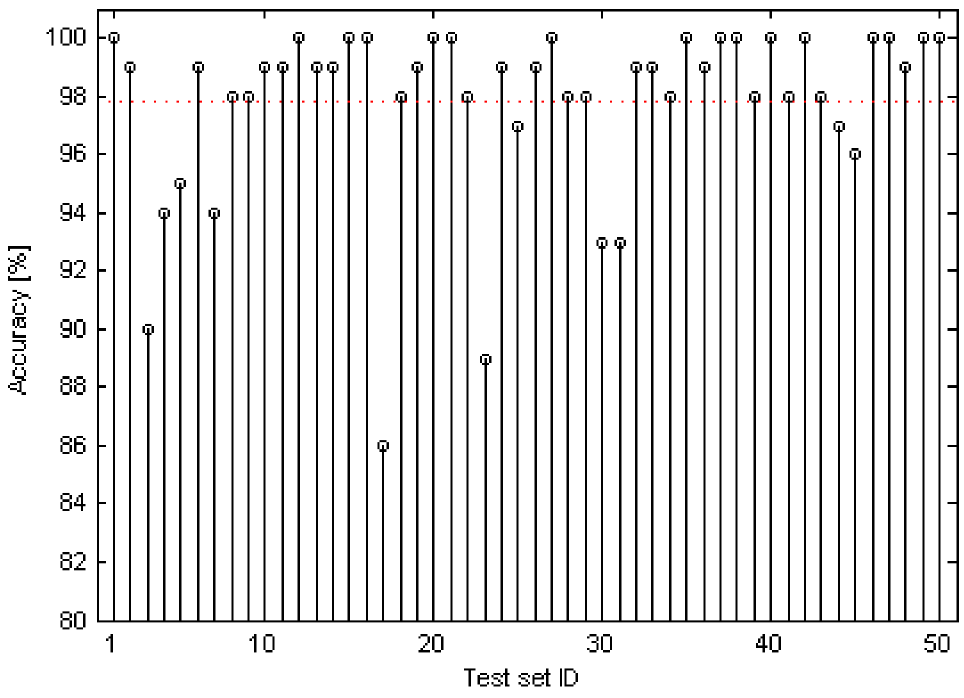

Figure 1. Accuracy of the clustering algorithm

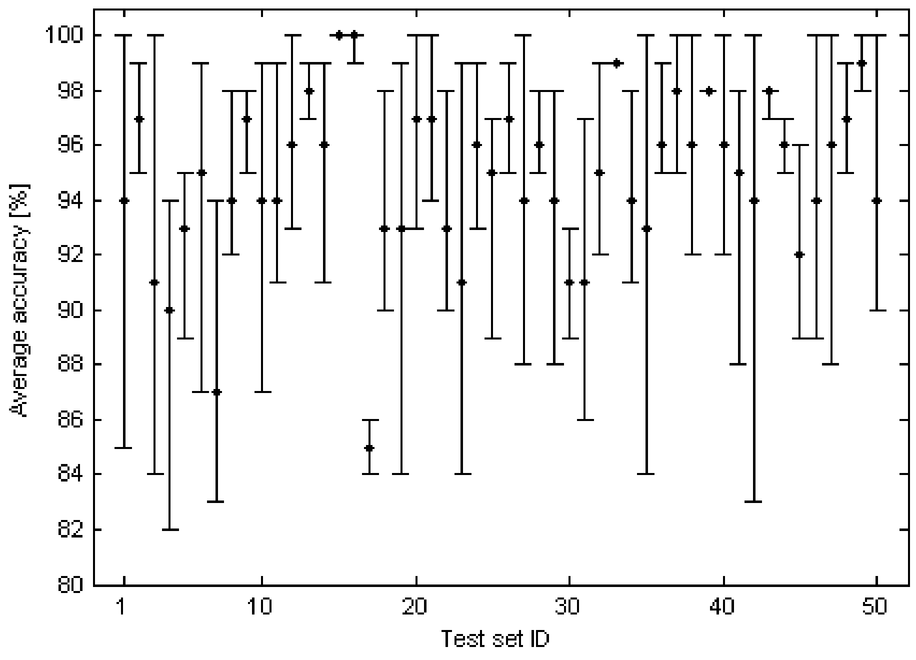

Figure 2. Average accuracy of the clustering algorithm

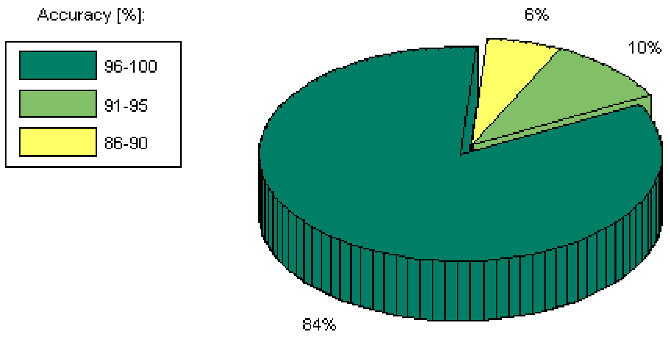

Figure 3. Distribution of the accuracy

Figure 4 demonstrates the difference between the accuracy of the result chosen by the user and the average accuracy of the potential results. The horizontal axis represents the identifier of the test set, and the vertical axis gives the increase of accuracy. The figure clearly indicates that the user feedback has unambiguously positive effects on the accuracy. We can see that 7% is the highest increase and there are only two cases where an infinitesimal decrease can be observed.

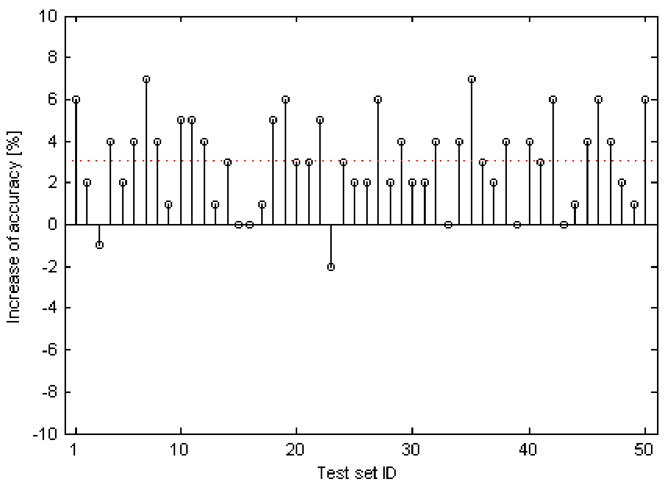

Figure 2. Impact of user interaction on the accuracy

IV. CONCLUSION

The output of text clustering can be used in solving other problems like abstracting [12]. In case of automatic clustering algorithms can be found in the literature, a lot of parameters influence the inner operation of the methods and the accuracy as well. In contrast, manual clustering when users perform the grouping procedure is excellent for the accuracy (accuracy can reach the maximum), but it takes a long time, needs many human resources, and it is almost impossible to carry out for large corpus. Automatic and manual approaches have some advantages, but half-automatic concept such as our solution has benefits from both of them. The largest challenge in our method is to determine the potential results for users, and to find out the questions based on these results. If the potential results are weak (accuracy is low), then the user will not get right solution. We developed a complex algorithm, which can produce very high accuracy for the potential results.

Test results showed that the average accuracy was near to 98%, and the result was perfect (100% accuracy) for almost one third of tests. However, the accuracy of our algorithm is excellent, we analyzed its reliability as well. We calculated the highest, lowest and average accuracy values related to the potential results. The average accuracy fell in the interval 82% and 100% while the mean value was 95%. Thus, we can conclude that the algorithm produced a relatively high accuracy for all of the potential results. Furthermore, we investigated the difference between the accuracy of the result chosen by the user and the average accuracy of the potential results. The user feedback had unambiguously positive effects on the accuracy with a maximum increase of 7%.

REFERENCES

[1] J. Kleinberg, “An Impossibility Theorem for Clustering,” *Proceedings of the 15th International Conference on Neural Information Processing Systems*, (Vancouver, Canada,) 2002, pp. 446–453.
[2] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall Advanced Reference Series, 1988.
[3] C. Luo, Y. Li, and S. M. Chung, “Text Document Clustering Based on Neighbors,” *Data & Knowledge Engineering*, Volume 68, Issue 11, 2009, pp. 1271–1288.
[4] D. Isa, V. P. Kallimani, and L. H. Lee, “Using the Self Organizing Map for Clustering of Text Documents,” *Expert Systems with Applications*, Volume 36, Issue 5, 2009, pp. 9584–9591.
[5] R. Gil-García and A. Pons-Porrata, “Dynamic Hierarchical Algorithms for Document Clustering,” *Pattern Recognition Letters*, Volume 31, Issue 6, 2010, pp. 469–477.
[6] W. Song, and S. C. Park, “Genetic Algorithm for Text Clustering Based on Latent Semantic Indexing,” *Computers & Mathematics with Applications*, Volume 57, Issues 11–12, 2009, pp. 1901–1907.
[7] W. Zhang, T. Yoshida, X. Tang, and Q. Wang, “Text Clustering Using Frequent Itemsets,” *Knowledge-Based Systems*, Volume 23, Issue 5, 2010, pp. 379–388.
[8] M. Rosell, V. Kann, and J.-E. Litton, “Comparing Comparisons: Document Clustering Evaluation Using Two Manual Classifications,” *Proceedings of the 3rd International Conference on Natural Language Processing*, (Hyderabad, India,) 2004, pp. 207–216.
[9] R. Baeza-Yates, and R. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, ACM Press, New York, 1999.
[10] R. M. Aliguliyev, “Clustering of Document Collection – A Weighting Approach,” *Expert Systems with Applications*, Volume 36, Issue 4, 2009, pp. 7904–7916.
[11] J. Walters-Williams, and Y. Li, “Comparative Study of Distance Functions for Nearest Neighbors,” In: K. Elleithy (ed.), *Advanced Techniques in Computing Sciences and Software Engineering*, Springer, New York, 2010, pp. 79–84.
[12] Q. Guo, and M. Zhang, “Multi-documents Automatic Abstracting Based on Text Clustering and Semantic Analysis,” *Knowledge-Based Systems*, Volume 22, Issue 6, 2009, pp. 482–485.
[13] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*, Cambridge University Press, 2009.
[14] C. T. Meadow, B. R. Boyce, D. H. Kraft, and C. Barry: *Text Information Retrieval Systems*, Academic Press, London, UK, 2007.
[15] T. Liu, S. Liu, Z. Chen, and W.Y. Ma, “An Evaluation on Feature Selection for Text Clustering,” *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.
[16] M. W. Berry, and M. Castellanos, *Survey of text mining II: clustering, classification, and retrieval*, Springer, London, UK, 2007.
[17] F. Beil, M. Ester, and X. Xu, “Frequent term-based text clustering,” *KDD '02 Proceedings of the eighth ACM SIGKDD international*

conference on Knowledge discovery and data mining, 2002, pp. 436-442.

[18] B. Tang, M. Shepherd, E. Milios, and M. I. Heywood, “Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering,” *Proc. of Feature Selection for Data Mining: Interfacing Machine Learning and Statistics in conjunction with the 2005 SIAM International Conference on Data Mining*, April 23, 2005, Newport Beach, CA, pp. 17-26.

[19] Y. Huang, and T. M. Mitchell, “Text clustering with extended user feedback,” *Proceeding of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06)*, 2006, pp. 413 - 420.

[20] S. Basu, “Semi-supervised Clustering: Learning with Limited User Feedback,” PhD Doctoral Dissertation, The University of Texas at Austin, 2003.

[21] D., Cohn, R., Caruana, and A. McCallum, “Semi-supervised clustering with user feedback,” (Tech. Report TR2003-1892). Cornell University, 2003.

[22] O. Alonso, M. Gertz, R. Baeza-Yates, “Clustering and exploring search results using timeline constructions,” *Proceedings of the 18th ACM conference on Information and knowledge management, (CIKM '09)*, 2009, pp. 97-106.

[23] I. Hwang, M. Kahng, and S-g Lee, “Exploiting user feedback to improve quality of search results clustering,” *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication (ICUIMC '11)*, Article No. 68.

Gábor Szűcs was born in Hungary in 1970. He has received MSc in Electrical Engineering from Budapest University of Technology and Economics (BME) in Budapest in Hungary in 1994.

He is experienced in modeling and simulation, railway systems, traffic systems; he has received PhD degree in this field from BME in 2002. His further and currently research areas are data mining, multimedia mining, content based image retrieval, semantic search. He is associate professor at Department of Telecommunications and Media Informatics of BME. The number of his publications is more than 80.

Dr. Szűcs is vice president of the Hungarian Simulation Society (EUROSIM), deputy director of the McLeod Institute of Simulation Sciences Hungarian Center. He has earned János Bolyai Research Scholarship of the Hungarian Academy of Science in 2008.

Zoltán Móczár has received MSc in Computer Science from Budapest University of Technology and Economics (BME) in Budapest in Hungary in 2011.

He is PhD student at Department of Telecommunications and Media Informatics of BME.