
Production Systems and Information Engineering
Volume 12 (3), pp. 23–35.
https://doi.org/10.32968/psaie.2024.3.3

DISTANCE-BASED CLUSTERING OF DOCUMENT WORDS

LÁSZLÓ BEDNARIK
University of Miskolc, Hungary Institute of Information Technology
laszlo.bednarik@uni-miskolc.hu

PÉTER MILEFF
University of Miskolc, Hungary Institute of Information Technology
peter.mileff@uni-miskolc.hu

Abstract: The feasibility of automatic question generation relies heavily on organizing the words of the document into appropriate clusters. The primary aim is to form groups from the words of the document where the words within each group exhibit similarities based on certain predefined properties. Accurately uncovering similarities between words lays the groundwork for automatically determining the words to be highlighted as questions and offering alternatives for their substitution using knowledge-intensive methods.
The distance between words is calculated based on the frequency of their co-occurrence in sentences within the documents. Thus, two words are considered closer if there are more sentences in the documents where the two words appear together. The developed concept has been implemented with several different algorithms to enable comparison of results and to reveal their advantageous and disadvantageous properties.

Keywords: *clustering, search strategies, covering disks*

1. Literature review of clustering

Clustering means grouping the points and records of a data set based on similarity. Clustering is suitable for descriptive modeling of almost any large data set. At first, the task of clustering began to be dealt with in statistics, and then the questions of clustering very large data sets came to the fore in the context of data mining. Thus, clustering is one of the oldest and perhaps the most commonly used parts of data mining. During clustering and classification, data points are classified into disjoint groups, clusters, and classes, i.e., the elements of the data set are partitioned. The purpose of clustering is to form separate groups of documents in such a way that those in the same group are as similar as possible, and those in different groups are as different as possible.

Difficulty of the task:

- representing objects,
- there are different methods for measuring similarity,
- cluster boundaries are not clear,
- there is no clear metric for measuring the goodness of group training,
- the efficiency of handling large tasks is low.

During the clustering process, forming the right groups is not a straightforward task, because people often take different aspects into consideration when grouping.

Depending on the application and habits, people cluster the same data set differently. When clustering, in addition to the data set, we must specify how to define the similarity of the elements, and also on what basis to group the elements. We need to provide an objective definition of the degree of similarity of elements and the quality of clustering. If the task is embedded in a suitable mathematical model, it is possible to find algorithms that solve the task well and quickly. During clustering, we classify the elements into groups and classes, so we perform classification. What distinguishes clustering from the classification task is that it is not specified which element belongs to which class, so there is no expert who helps us learn with correct teaching samples. That is why clustering is called unsupervised learning.

Unlike classification, when grouping documents

- there are no teaching documents with known labels, and the groups usually cannot be labeled automatically even after completing the task,
- the number of groups is not fixed.

Clustering starts from the similarity relationships of the data points, so the first important step is to select the similarity function.

When clustering large data sets, we often encounter the problem that the calculation of the distance and similarity of the data points cannot be solved quickly enough for the given clustering algorithm to finish in an acceptable runtime. In many cases, this problem is caused by characterizing the data with many attributes. The problem of efficiently calculating similarity or distance can be solved by dimensionality reduction of the data, during which the data set is transformed into a form on which the desired clustering algorithm can be run efficiently.

2. Types of clustering methods

The most widely used standard methods are hierarchical [3], partitioning [2], hybrid, incremental and non-incremental, monothetic versus polythetic [7] and fuzzy [5] methods. Each method has its own scientific basis, but their effectiveness and complexity are different. The methods differ fundamentally in that they give a binary or fuzzy output as a final result. In the case of binary clustering, a given text can only be a member of one cluster, while in the case of Fuzzy output, we are talking about membership functions, when the extent to which each text belongs to a given cluster is expressed as a percentage for each cluster.

Hierarchical methods get their name from the fact that the elements are organized into a hierarchical data structure (tree, dendrogram, taxonomy). The data points are located in the leaves of the tree. Each internal point of the tree corresponds to a cluster, which contains the points that are located below it in the tree. The root includes all data points. The advantage of the method is that it gives more options for the solution, the disadvantage is that it requires more calculations.

We distinguish two types of hierarchical procedures: the one built from below, also known as unifier, and the one built from above, or divider. In bottom-up accumulation procedures, each element is initially a separate cluster, and then the procedure combines the closest clusters, forming a new cluster one level higher in the hierarchy. Top-down decomposition methods work the other way around: they start from a single cluster containing all data points, which is partitioned into smaller clusters, and then further decomposed.

The key step in hierarchical algorithms is the selection of clusters to *merge* or *split*. After a merge (split) takes place, all further operations are performed on the new clusters. Different hierarchical clustering algorithms are used depending on the choice of the function that measures the distance of the clusters.

In the case of the HAC *(Hierarchical Agglomerative Clustering)* [4] algorithm, it combines two nearby clusters in each step. The stopping condition is the minimum number of clusters or the maximum merging distance.

Steps of the algorithm:

- each element is an independent cluster,
- determination of the two nearest clusters,
- merging the two nearest clusters into one,
- continuing the above procedure until the stop condition allows it.

The most well-known HAC clustering methods: the method based on the smallest, largest and average distance, the Ward method, the BIRCH *(Balanced Iterative Reducing and Clustering using Hierarchies)* algorithm, the CURE algorithm, the Chameleon algorithm.

If the distance d_{min} of two clusters is defined by the distance of their nearest points, then we are talking about a procedure using the smallest distance (*single linkage*). Algorithms based on the smallest distance are suitable for discovering well-separated clusters of arbitrary shape. Since they take into account the minimum distance between cluster elements, they tend to connect two clusters if one of their elements gets too close to each other.

If the maximum distance is used instead of the minimum distance to determine the distance between two clusters, then we are talking about a *complete linkage* method, and if the average similarity or the diameter of the combined cluster is used, then we are talking about an average linkage method.

Partitioning methods provide a possible division of documents, where the groups are independent of each other. The advantage of this solution is the favorable calculation requirement, the disadvantage is inflexibility, which also follows from the fact that the number of groups must be fixed in advance.

HDC *(Hierarchical Divisive Clustering)* clustering procedures: *k-means* and *k-medoid* algorithm.

3. Search strategies

The search strategy determines which of the nodes waiting to be expanded will be expanded next. The strategies can be evaluated according to the following aspects:

- Completeness: if there is a solution, the search engine is guaranteed to find it.
- Optimality: if there are several different solutions, the search engine will find the best of them (shortest, cheapest).
- Time complexity: how long does the search for a solution take, and in any case, does the search run within a foreseeable time?
- Space complexity: the amount of memory required for the search.

Search algorithms are distinguished from the point of view of whether they are provided with knowledge that helps to find a solution. Those search engines that only know about the current state, whether it is a target state, are called *non-informed search* strategies. If the searcher can draw additional conclusions about a state, then we are talking about *informed search* or *heuristic search* strategies.

The most well-known uninformed search strategies are: breadth-first, depth-first, depth-constrained, iterative deepening, uniform cost, and two-way search.

Searchers that have extra information about the states in the nodes are called informed or heuristic.

Informed search strategies: Iteratively improving searchers, Hill-climbing search, Simulated cooling, Best-first searchers, Greedy search and A* search.

To implement the task, I worked out a further development of the best-first search strategy method, which has not been published in the literature so far.

In this publication, a new type of the best-first search strategy method was investigated and developed for clustering of document words.

4. Methods and algorithms developed for word clustering

Two significantly different concepts were developed for clustering the words of the documents. According to one, the distance between two words is defined by the frequency of their common occurrence in sentences, while according to the other, it is defined by the geometrical distance between the words' positions in objective space. In the paper, the concept of the distance between two words and their frequency of co-occurrence in sentences is presented.

4.1. Distance determination based on the frequency of common occurrence of words

According to the distance determination concept based on the frequency of co-occurrence of words, the distance between two words can be defined by the number of sentences in which both words appear at the same time. The distance data determined in this way can be described with a square matrix (distance matrix). Both the rows and columns of the matrix are indexed with the words of the

document. The matrix *i.* in row and *j.* the value in column 4.1. was determined on the basis of correlation [9], [10].

$$S_{i,j} = f_{i,j} \; / \; max(f_i, f_j) \qquad d_{ij} = 1 - S_{i,j} \tag{4.1}$$

Where: S_{ij} is $i.$ and $j.$ a numerical value in the interval [0...1] representing the word distance relationship (a larger value indicates a smaller distance between words); $f_{ij} f_{i,j}$ is the number of sentences in the document in which $i.$ and $j.$ word is also included; f_i is the number of sentences in the document in which $i.$ word is included; f_j s the number of sentences in the document in which $j.$ word is included.

The main goal of measuring the distance between words is to uncover the relationships between them, such as which words frequently appear together and how strong the connection between them is. The solution is based on the idea that if two words often occur together in the same sentence, there is likely a connection between them. This connection is measured by the co-occurrence frequency (f_{ij}).

The value of f_j is defined because the goal of determining the distance is to find out how closely different words are related. The distance between words, d_{ij}, is related to the number of co-occurrences (f_{ij}). The more often two words appear together in sentences, the smaller the distance between them will be. However, it is also important to consider how many times a given word appears in the entire document (f_i or f_j).

To elaborate, f_{ij} shows the number of co-occurrences, indicating how many times two words appear in the same sentence. If they appear together frequently, it can be assumed that there is some connection between them, meaning their distance is smaller.

The other factor is the sentence count (f_i, f_j), which reflects the frequency of the words. If a word is very common, it will naturally occur with many other words, but this does not necessarily mean that there is a strong connection between them. This is corrected by dividing f_{ij} by f_j (or f_i).

This concept has several advantages:

- Normalization by taking word frequency into account
- Ensuring scale independence
- Providing more reliable results.

4.2. Extending distance computation to clusters

Two task-specific further developments of the general method of the HAC algorithm [1] were implemented. The first method makes use of the property of the words to be clustered that their relationship can be characterized by their belonging to the sentence in addition to the distance. This makes them different from points that can be clustered based on their distance alone without a deeper connection. This method extends the use of relation (4.1) to clusters instead of words.

According to this, f_i represents the number of sentences in which all words belonging to i. a cluster are included; f_j represents the number of sentences in which all words belonging to j. cluster are included; and f_{ij} represents the number of sentences in which all words belonging to both cluster i. as well as j. are included.

By consistently applying the relation, only words that appear together in at least one sentence of the document can be included in each cluster. This concept, in addition to preserving the property of words belonging to sentences, also sets a limit to the maximum number of words that can be placed in clusters. The disadvantage of the method is that the entire distance matrix needs to be recalculated after each cluster merging step. By extending the relation to clusters, the distance matrix no longer stores the distance of words, but of clusters, and the number and content of clusters changes after each merging.

In each step of the clustering process, the clusters with the smallest distance based on the correlation are merged. Any case where the distance of a cluster to several other clusters is equally the smallest results in a branching of the search process. In this way, several solutions to the same task are possible. Finding solutions requires the use of tree traversal algorithms. Since the goodness of a solution in the current task depends solely on its correlation with the subjective space, the goodness of the intermediate states obtained during the clustering process cannot be estimated even with a heuristic function.

In order to increase the speed of the algorithm, in addition to the traditional *"breadth-first-search"* [6] algorithm, clustering was also implemented with the *"depth limited depth-first-search"* [6] algorithm. The objective function of the "breadth-first and depth limited depth-first" algorithm is to find the very first state in which all words and elements of the document to be clustered belong to exactly one cluster, and there are no clusters where theren't two words whose distance exceeds the value specified as a parameter. The two developed algorithms differ from each other in the strategy used to determine the next clustering step, starting from the initial unclustered state. According to this, in each step, the breadth-first search produces all the states that are created from the initial state by organizing the same number of words as the number of the given step into a cluster. In contrast, the "depth limited depth-first" search continues to cluster the currently determined state, until the predefined number of steps or full clustering is reached.

In the case of applying the depth search, after each cluster merging step, a new node number b (branching factor) is created. Denoting the depth of the solution requiring the maximum number of steps by m, the storage requirement of the search will be $O(b*m)$, i.e. the space complexity is linear. The storage requirements of the search are very modest, as it only needs to store the path from the root node to a leaf node, completed with the unexplored nodes next to each node in the path.

The time required for the search is $O(b^m)$, which is the same as the worst-case time required for the breadth-first search [8]. Depth-limited search is almost identical to depth-first search, but it does not expand nodes after a certain depth, thus forcing backtracking. Similar to the depth finder, its time requirement is $O(b^k)$, storage requirement is $O(b * k)$, where k denotes the depth limit.

Choosing a good limit is key when searching with a depth limit. The essence of the iterative deepening search is that it combines the advantageous properties of the breadth and depth search, i.e. it tries all possible depth limits until a solution is found. The expansion is done according to the width search, with the difference that here you can expand a node several times, and thus the search is complete and optimal, but, like the depth search, its memory requirement is relatively small.

Table 1 summarizes the concepts used in the implemented clustering algorithms.

Table 1. Concepts used in the implemented clustering algorithms

Concepts	Definition of the concept
word	according to the rules of Hungarian spelling, it is a meaningful word
sentence	*list of words (1 or more words)*
document	*list of sentences (1 or more sentences)*
cluster	*set of words (1 or more words)*
state	*set of clusters where every word of the document is an element of exactly one cluster*
merging two clusters	The condition for merging clusters is that both clusters are in the same state. During the merging of two clusters, both clusters are terminated and a new cluster is created in their place, the words of which can be given by the union of the words of the merged clusters.
frequency of cluster occurrence	the number of sentences in the document that contain all the words in the cluster
co-occurrence of two clusters	the number of sentences in the document that contain all words in both clusters
distance between two clusters	the distance between two clusters can be determined by a division, in which the numerator is the common frequency of occurrence of the two clusters, and the denominator is the maximum frequency of occurrence of the two clusters
initial state	a condition where each cluster contains exactly one word
end state	a state in which there are no two clusters whose distance is smaller than a predetermined value
current status	the state under analysis
child state	the child state of a state is the state obtained by merging two clusters of the given state

Table 2 shows the clustering algorithm implemented by breadth-first search.

Table 2. Algorithm of clustering implemented by breadth-first search

Step	Action
1.	Creating the initial state. (Collecting all the different words of the document and putting each of the resulting words into a separate cluster.)
2.	If the initial state fulfills the condition of the final state, then the initial state is added to the list of solutions. Go to step 19.
3.	Adding the initial state to the list of states to be analyzed.
4.	The current state should be the very first item in the list of states to be analyzed.
5.	After all matching of the clusters of the current state is done, the distance between the two nearest clusters is determined.

Step	Action
6.	After completing all pairings of the clusters of the current state, declare those pairs of clusters whose distance is the same as the distance of the nearest clusters to be merged.
7.	Merge all cluster pairs declared to be mergeable, the merging of which does not prevent the merging of other cluster pairs declared to be merged. (Example: mark A, B, C, D, E, F as the six clusters of the current state. Also mark A-B, A-D, D-F, C-E as the three pairs of clusters declared to be merged in the current state. Then the merging of clusters A-B will fail the merging of clusters A-D, The Merging clusters A-D will fail the merging of clusters A-B and D-F, but merging clusters C-E will not fail the merging of any pair of clusters to be merged. Such cluster pairs must be merged in this step.)
8.	If there are any cluster pairs declared to be mergeable that are not merged in step 7, go to step 12.
9.	If the current state does not meet the final state condition, go to step 5.
10.	If the current state is not included in the list of solutions, the current state is added to the list of solutions.
11.	If the number of states in the solution list has reached the set maximum value, go to step 19, otherwise go to step 17.
12.	The cluster pairs declared to be mergeable that are not merged in step 7 are merged in such a way that a new state is created for each cluster pair declared to be mergeable, in which only the cluster pair that triggered the creation of the new state is merged from the clusters of the current state.
13.	Adding each new state created in step 12 to the list containing the child states of the current state.
14.	Among the child states of the current state, those that fulfill the condition of the final state and are not included in the list of solutions, add them to the list of solutions.
15.	If the number of states in the list of solutions has reached the set maximum value, go to step 19.
16.	Among the child states of the current state, those that do not meet the condition of the end state and are not included in the list of states to be analyzed, add them to the list of states to be analyzed.
17.	Delete the current status from the list of statuses to be analyzed.
18.	If the list of states to be analyzed is not empty, go to step 4.
19.	End.

Table 3 shows the clustering algorithm implemented by depth search.

Table 3. Algorithm of clustering implemented by depth search

Step	Action
1.	Creating the initial state. (Collecting all the different words of the document and putting each of the resulting words into a separate cluster.)
2.	Adding the initial state to the list of states to be analyzed.
3.	The current state should be the last item in the list of states to be analyzed.
4.	After all matching of the clusters of the current state is done, the distance between the two nearest clusters is determined.
5.	If the current state does not meet the end state condition and the number of states in the list of states to be analyzed is less than the depth limit of the search, go to step 13.

Step	Action
6.	If the current state does not meet the final state condition, go to step 9.
7.	If the current state is not included in the list of solutions, the current state is added to the list of solutions.
8.	If the number of states in the list of solutions has reached the specified maximum number of solutions, go to step 18.
9.	Delete the last state in the list of states to be analyzed.
10.	If the list of states to be analyzed does not contain more states, go to step 18.
11.	If the last state of the list of states to be analyzed does not contain a child state for which the final state examination has not yet been performed, proceed to step 9.
12.	Among the child states of the last state on the list of states to be analyzed, on which the final state test has not yet been performed, the next one is added to the end of the list of states to be analyzed. Go to step 3.
13.	After all pairings of the clusters of the current state have been performed, those clusters whose distance is the same as the distance of the nearest clusters should be declared to be merged.
14.	Merge all cluster pairs declared to be mergeable, the merging of which does not prevent the merging of other cluster pairs declared to be merged. (Example: mark A, B, C, D, E, F as the six clusters of the current state. Also mark A-B, A-D, D-F, C-E as the three pairs of clusters declared to be merged in the current state. Then the merging of clusters A-B will fail the merging of clusters A-D, The Merging clusters A-D will fail the merging of clusters A-B and D-F, but merging clusters C-E will not fail the merging of any pair of clusters to be merged. Such cluster pairs must be merged in this step.)
15.	The cluster pairs declared to be mergeable that are not merged in step 14 are merged in such a way that a new state is created for each cluster pair declared to be merged, in which only the cluster pair that triggered the creation of the new state is merged from the clusters of the current state. Adding each new state created in this step to the list of child states of the current state.
16.	Adding the very first child state of the current state to the end of the list of states to be analyzed.
17.	Go to step 3.

4.3. The method of covering with discs

Since the cost of recalculating the distance matrix after each cluster merge exceeds the computing capacity of a high performance computer, even with a sentence number of around one hundred, it is necessary to develop an algorithm that performs all the calculations required for clustering based solely on the initial distance matrix. By avoiding the continuous recalculation of the distance matrix, it is possible to significantly reduce both the calculation time and the memory requirements of the clustering process. However, by merely using the distance matrix created at the beginning of the clustering, the information that can be used to ensure that only words that appear together in at least one sentence of the document are included in a cluster is lost.

In the breadth and depth search algorithm, this information was used not only to determine the distance between the clusters, but also to keep the size of the clusters

within limits. Since, according to the rules of the HAC algorithm, each point (word) is located in a separate cluster in the initial state, it is therefore not possible to find out from the distance matrix created in the initial state, in the case of more than two words, whether there is a sentence in which they appear together.

In the case of clustering based on the initial distance matrix, a known method to keep the size of the clusters within limits is to perform clustering based on the distance between the most distant words of the clusters. The prepared algorithm contains a further development of this method based on a new idea.

During clustering, each cluster was modeled with a disc with a predefined diameter [1]. This is similar to clustering based on the distance of the farthest words, since the diameter of the disks can limit the distance of the farthest points (words) that can be placed in the cluster. In addition to this, however, this algorithm also opens up the possibility of managing points belonging to several clusters. This additional information can be used so that the clustering of words that can be assigned to several clusters can be dynamically changed according to current needs. The objective function of the clustering optimization algorithm is to gather all the words of the document to be clustered into a minimum number of clusters. Adherence to a predefined uniform diameter for the clusters appears as a limiting condition. In the initial state of the clustering process – according to the rules of the general HAC algorithm – each word is placed in a separate cluster. During the algorithm, the diameter of the disks representing the clusters is continuously increased, thus the clusters include the words one after the other. If during the process every word of a cluster is an element of a cluster other than the given cluster, then the given cluster is eliminated. The algorithm was implemented using the best-first-search [6] strategy. During the process, the algorithm always increases the diameter of the cluster that results in the elimination of several clusters after the increase.

The resulting states represent the possible next states of the current state. Among these states, the cluster growth step resulting in the one containing the fewest clusters is executed. In the case of several possibilities with the least number of clusters, the one found first will be implemented.

The condition for stopping the process is that the diameter of each cluster is equal to the predetermined value. As a result, each cluster contains one or more words that the other clusters do not, but may also contain words that belong to other clusters. *Table 4* illustrates the clustering algorithm implemented with the best-first search strategy.

Table 4. A clustering algorithm implemented with a best-first search strategy

Step	Action
1.	Creating the initial state. (Collecting all the different words of the document and putting each of the resulting words into a separate cluster.)
2.	Investigating the possibility of merging each cluster with all other clusters. (Two clusters can be merged if the distance between their most distant words is not greater than the permitted disk diameter.)
3.	Gathering the cluster pairs classified as mergeable in point 2 in one list.

Step	Action
4.	If the list does not contain at least one cluster pair, go to step 7
5.	For each pair of clusters on the list, it is determined how many clusters would be eliminated if they were merged. (By merging all cluster pairs, both elements of the cluster pair are eliminated, a new cluster is created, which contains both elements of the cluster pair at the same time, and all clusters whose elements are contained by at least one other cluster are eliminated.)
6.	Merging the one of the cluster pairs in the list that results in the elimination of the most clusters. In the case of several identical values, merging the first pair of such clusters found. Go to step 2.
7.	End.

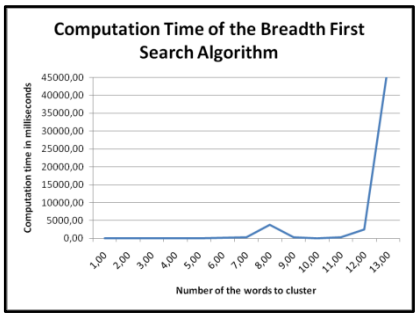
The best-first search estimates the cost of achieving the goal from state n with a heuristic evaluation function and moves to the state for which this cost is the smallest. The storage and time cost of the search is $O(b^m)$, where m is the maximum depth of the search space. However, the complexity can be significantly reduced with a well-chosen heuristic function. The amount of reduction depends on the given problem and the quality of the heuristic reduction [6].

5. Comparison of the results of the implemented algorithms

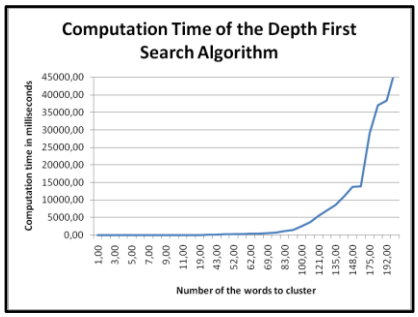
In the implemented clustering algorithms, a possible solution to the task is any state in which the elements of all the words received from the preprocessing module belong to one or more clusters and, due to the previously set size limit, no cluster can store more words. Several solutions to the same clustering task are possible, which differ in their parameters (number of clusters; distance of the most distant words within a cluster; number of sentences in the document in which the words in the clusters appear together, etc.).

Although the implemented algorithms are suitable for exploring all possible solutions of the task, only the parameters of the first solution have been compared in the article – due to reasons of scope. By extending the formula to clusters, the time to find the solution – due to the continuous recalculation of the distance matrix – depends squarely on the number of words to be clustered. The branching factor of the search tree is added to the power exponent of the value thus obtained. This number depends on the nature of the document's sentences (word repetition) and is difficult to estimate in advance.

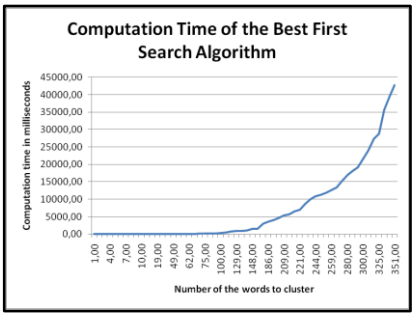
Figure 1 shows a comparison of the running time requirements of the implemented algorithms as a function of the number of words to be clustered. It can be seen that the time required to find the solution only partially depends on the number of words to be clustered. Since the solutions are created as a result of several merging processes of the clusters, it is clear that the breadth-first strategy must traverse almost the entire tree to find the very first solution. The diagram clearly shows the additional time required by the continuous recalculation of the distance matrix compared to the method of covering the best first with disks based on a strategy.



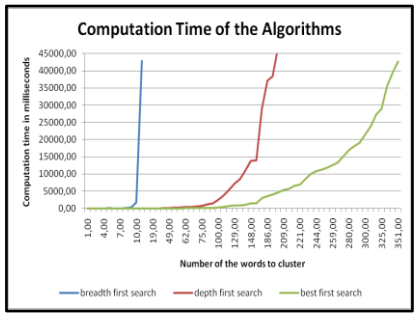
1.a



1.b



1.c



1.d

Figure 1. The time complexity of the implemented algorithms

Figure 1 shows the results of clustering methods by extending the formula of the distance matrix to clusters and by covering them with disks. The number of clusters included in the solutions is *Figure 1.a*, and the distance of the furthest words in the cluster containing the furthest words is shown in *Figure 1.b.* The distance of the most distant clusters and words to be merged was set to 0.25 for both methods after the normalized distance data for the [0...1] interval.

Conclusion

Based on the information obtained by testing the presented clustering methods, the clustering task of the automated question generation sample system was implemented with the concept of distance formation based on word frequency. For this, we applied a task-specific improvement of clustering with disks of uniform diameter using a best-first search strategy. The task of the clustering methods selected on the basis of testing is to reduce the units to be examined in documents consisting of a large number of words by arranging the words of the document that are close to each other in certain aspects into clusters. In this way, instead of words, the smallest unit that can be examined is represented by clusters, which means significant resource savings. The clustering was implemented based on strategies based on the frequency of co-occurrence of words. The algorithms implemented with different search strategies were presented in detail, during which the running time costs of each algorithm were analyzed.

References

[1] Dr. Bodon Ferenc (2010). *Adatbányászati algoritmusok*. Free Software Foundation, Budapest, pp. 175–181.

[2] Tikk Domonkos (2007). *Szövegbányászat*. Typotex, Budapest.

[3] Jain, A., Murty, M., & Flynn, P. J. (1999). Data Clustering: A review. *ACM Computing Surveys*, Vol. 31, No 3, pp. 264–321.

[4] Manning, C., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[5] Ruspini, E. (1969). A new approach to clustering. *Information Control*, Vol. 15., pp. 22–32.

[6] Russell, Stuart, & Norvig, Peter (2005). *Artificial Intelligence*. A Modern Approach (2nd Edition), Pearson, Italia Rome.

[7] Salton, G. (1991). Developments in automatic text retrieval. *Science*, Vol. 253., pp. 974–980.

[8] Russel, S. J., & Norvig, P.: *Mesterséges Intelligencia modern megközelítésben*. Panem-Prentice-Hall, Budapest, 2000.

[9] Kovacs, L., & Bednarik, L. (2011). Extension of HAC clustering method with quality threshold, *9th IEEE International Symposium on Intelligent Systems and Informatics (SISY 2011)*, Subotica, Serbia, pp. 257–261. https://doi.org/10.1109/SISY.2011.6034333

[10] Bednarik L. (2012). *Automatizált kérdésgenerálás annotált szövegből*. PhD-értekezés, Miskolci Egyetem.

[11] Kovacs, L., Repasi, T., & Baksa-Varga, E. (2006). Clustering Based on context similarity. *Proceedings of the TMCE 2006*, April 18–22.

[12] Shin, S.-I., Choi, K.-S. (2004). Automatic Word Sense Clustering using Collocation for Sense Adaptation.

[13] Cawsey, A. (2002). *Mesterséges Intelligencia Alapismeretek*. Panem Könyvkiadó Kft. Budapest.

[14] Jain, A. K., & Dubes, R. C. (1988). *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series.

[15] Gonda L, Fülöp A, Hajas Cs. Jeszenszky P. (2011). *Bevezetés az adatbányászatba*. Panem Könyvkiadó Kft.

[16] Aggarwal, C. C. (Editor), Reddy, C. K. (2018). *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series Book 31, Chapman and Hall/CRC; 1st edition.