MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Ruse

# Information Retrieval

**Yordan Kalmukov**

May 2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

**Cluster Analysis**
**Grouping objects by common features**

Automated grouping of documents is a fundamental task that can be considered both as a *supplement to searching* for similar documents and as an *alternative way of searching*.

There are situations in which the user's work can be significantly facilitated if the results returned by the system are further grouped by certain characteristics. For example, if the user searching for "paper," the system will likely find and return results related to "paper – the material from cellulose pulp", "paper - scientific article" and "paper – set of exam questions". However, it is possible that the user is interested in only one of the meanings of the word "paper". If the results are mixed, he/she should review them all. It would be much more convenient if the returned results were further grouped by relevant features. When grouping is performed on the returned results, then it is considered an *augmentation/supplement to searching*, which aims to increase both the relevance of the obtained results and the convenience of the users' work.

It is possible to organize the search so that the user does not need to enter a search query at all. Instead, documents can be pre-grouped by common characteristics (at multiple levels if necessary), the groups could be presented to the user, who can review them and immediately navigate to the relevant group of interest. This action could be classified as "searching by browsing" and is an *alternative approach to searching*.

Besides searching, grouping documents by common features is very useful for implementing automated arrangement and categorization of documents within the collection. The process can be automated by using the so-called ***cluster analysis***. It aims to group *n* number of objects into *k* (k>1) number of groups, called *clusters*, based on *p* (p>0) number of features. Features can be explicitly specified by users (e.g., keywords) or implicitly extracted from the documents themselves by analyzing their content (e.g., vectors of unique words in the document, etc.).

Clustering methods and algorithms can be divided into 2 large groups:

- ***Flat clustering*** - objects are grouped into flat clusters (fig. 1). *It is known which object belongs to which cluster, but not how close the objects are within the cluster itself*. In this type of clustering, *it is necessary to specify the number of clusters in advance*.

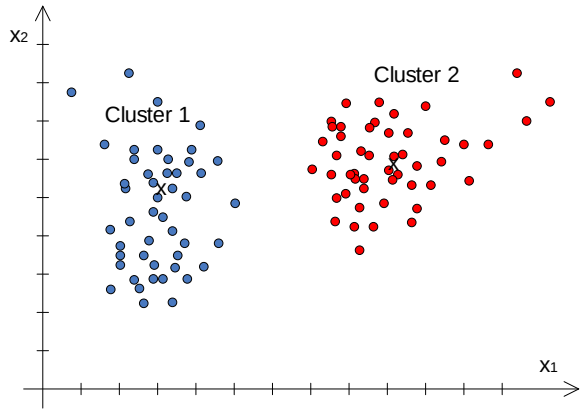



*Figure 1. Example of flat clustering*

- ***Hierarchical clustering*** - objects are grouped into hierarchical clusters (fig. 2). *It is known not only which objects belong to a given cluster, but also how close the objects are to each other within the cluster itself*. *And also how close clusters are to each other*. Hierarchical clustering algorithms can also be divided into two types - agglomerative (merging) and divisive. In agglomerative clustering, the grouping is "bottom-up", i.e. it starts with the set of individual objects, each forming its own cluster, and at each iteration the two closest clusters are merged. In divisive clustering, it is the other way round (from top to bottom) – one starts with a large common cluster and at each iteration, the most distinctive element is removed from it. In practice, agglomerative cluster analysis is more popular and more often implemented. In hierarchical clustering, it is not necessary to specify the number of clusters in advance, and the algorithm is given the freedom to follow the natural distribution of the data. Therefore, the clustering is more accurate, compared to flat clustering methods. The number of clusters can be determined at a later stage after analyzing the generated *dendrogram* or by setting a threshold distance when merging the clusters.

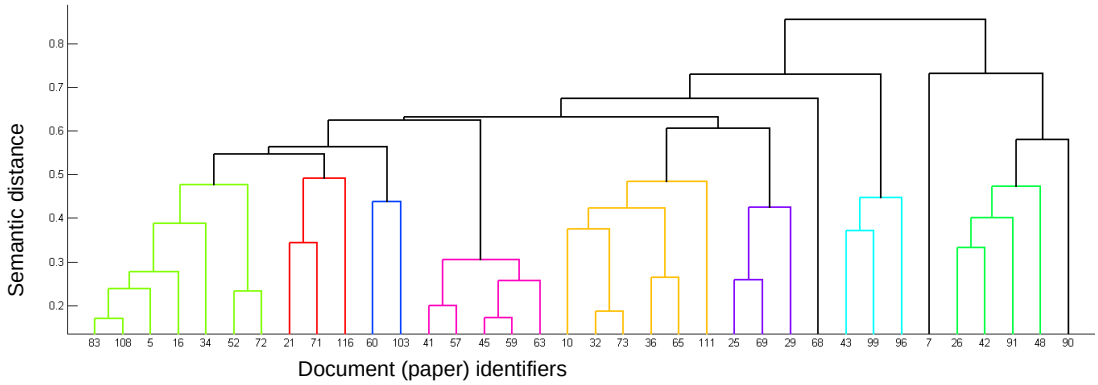



*Figure 2. An example of hierarchical clustering. Grouping papers submitted to the scientific conference CompSysTech 2017 in thematic fields and working sessions.*
*The graphic represent a colorful dendrogram (binary tree).*

**1. Flat clustering**

The most popular flat clustering algorithm is ***k-means***. It is easy to implement and provide good results. Works as follows:

1. The user specifies the number of clusters in which objects/documents should be grouped.
2. The algorithm assigns arbitrary centers (random coordinates) to the specified number of clusters.
3. For each object/document:
    a. The distance from the object to the centers of each cluster is calculated. Different similarity measures can be used. The original algorithm relies on Euclidean distance, but other measures can be used as well.
    b. The object is assigned to the cluster to whose center it is closest.
    c. The coordinates of the center of the cluster to which the object is assigned are recalculated. The new coordinates are calculated so that the sum of the squares of the distances between the center and all objects belonging to the cluster is minimal. In statistics, this is known as the method of least squares (LSM) and is often used as optimization criterion. Therefore, when adding a new object, the cluster's center need to be recalculated.

Since the algorithm starts with random centers, if it is used to cluster a small number of objects, its accuracy will not be very high because of the random coordinates at the beginning. But *as the number of clustered objects increases, the centers will be constantly recalculated and their coordinates will begin to tend towards the optimal ones* for the given clusters. This is important to know and consider! The k-means algorithm is a representative of the unsupervised learning algorithms from the field of machine learning. I.e. the more data is processed, the more accurately objects will be grouped. If a small number of objects (for example 5-10) are planned to be grouped with this algorithm, high accuracy cannot be expected. In this case, it is better to group the objects manually by a person.

The k-means clustering algorithm works directly with the objects' features (characteristics), not with the distances between them. This means that each object/document must be described with a feature vector whose values can be:

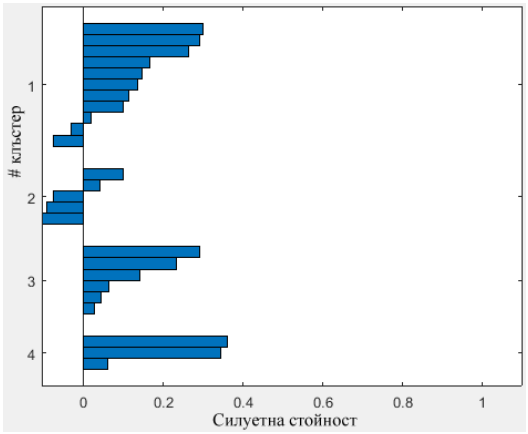
- *Binary values,* showing the presence or the absence of a given feature
  `objectId -> {0, 1, 0, 1, 1, 0}`
- *Real numbers*, chowing the degree of relevance (applicability) of each feature or tf-idf weights
  `objectId -> {0.75, 0, 0.25, 0.1, 0.5, 0}`
- *Semantic similarity/distance* between the current document and the other documents
  `objectId -> {0.65, 0.34, 0.09, 0.17, 0, 0.2}`
- *Explicitly stated keywords* (i.e. written in the form of ordinary strings) *
  `objectId -> {sweet, dessert, food}`

* The algorithm cannot directly work with keywords, so it must convert the set of keywords into a feature vector with binary values that indicate the presence or absence of the given keyword. Of course, an order of keywords must be introduced so that their positions in all vectors are identical. Otherwise, there is no way to know the presence (1) or absence (0) of which word it refers to.

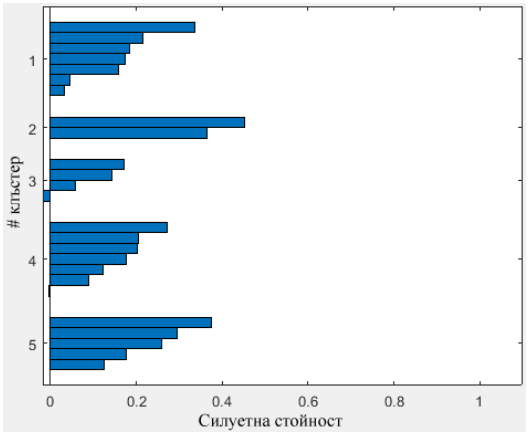
In general, the algorithm is not designed to work directly with distances or similarities between documents, but only with their features. However, in practice *it can also work with distances/similarities*, which has been experimentally confirmed by my own research. For this purpose, feature vectors must be constructed to contain the distances (similarities) from a given object to all others. As with real numbers and binary values, it is important that the elements are positionally ordered in the same way in all vectors.

One of the main problems of flat clustering and the k-means algorithm is that the user must specify in advance the number of clusters in which objects should be grouped to. However, the user, in general, has no way of knowing what would be the optimal number of clusters. Therefore, one should experiment with different numbers of clusters, compare the results, and thus determine the most suitable number. To compare the results (for different number of clusters) the user can rely on the so-called *silhouette diagram*. It shows how close each cluster representative is to its neighboring clusters. The silhouette value ranges from 1 (the object in the cluster is very far from the neighboring clusters), through 0 (the object is very close to another cluster or clusters and could practically be assigned to them as well) to -1 (the object is wrongly assigned to the given cluster since it is much closer to another).

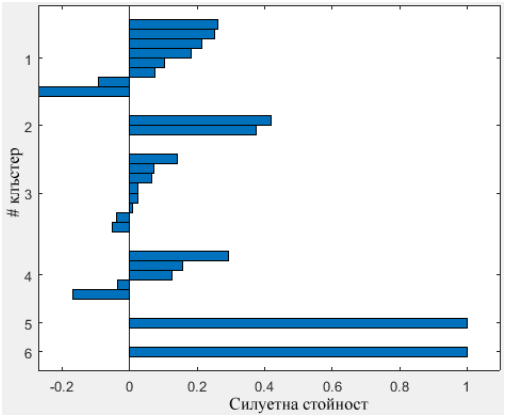
It is appropriate to present the silhouette diagram with an example: The results of a survey completed by 25 people regarding their hobbies are subject to grouping. Each respondent can choose any number of interests from a list of 30 predefined ones. The goal is to group people by interests/hobbies. The survey was conducted by the student Denitsa Vasileva in the spring of 2020. The results of grouping for 4, 5 and 6 clusters are presented by the silhouette diagram in fig. 3. Since people's interests are described in 30-dimensional space, there is obviously no way to represent grouping in a plane diagram like fig. 1.




*a) in 4 clusters*




*b) in 5 clusters*




*c) in 6 clusters*

*Figure 3. Silhouette diagram showing the quality of grouping people by interests in 4, 5 and 6 clusters.*

With 4 clusters, in groups 1, 2 and 3 there are objects that are very close to other clusters and can be safely assigned to them, i.e. it cannot be reliably said whether they are correctly classified or not. At the same time, there are clearly misclassified objects in clusters 1 and 2, and in cluster 2 the misclassified ones are even more than the correctly assigned ones. Apparently 4 clusters is not an appropriate number. With 5 clusters, the results are significantly better, with only one document (in cluster 3) that can be transferred to another cluster. At 6 clusters, the results deteriorate again. Obviously, the optimal number of clusters where the clustering seems best in this case is 5.

As a second test, using the k-means algorithm, 42 scientific papers from the CompSysTech 2016 conference, from all sections except for e-learning, were grouped by thematic fields. Such an automated grouping is very useful as a basis for quick and easy conference program generation. According to the silhouette diagram, the best grouping is achieved in 6 clusters (fig. 4).

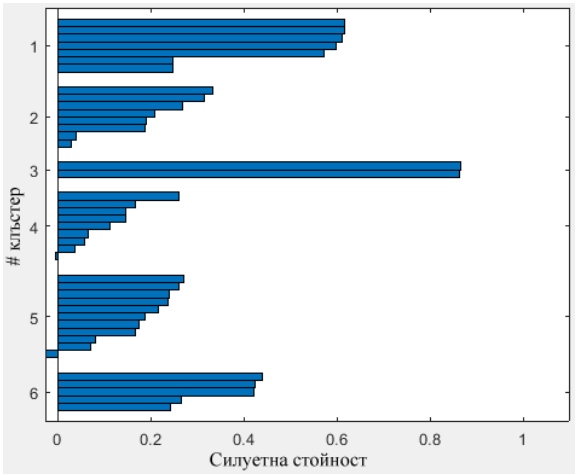



*Figure 4. Silhouette diagram showing the quality of grouping of 42 scientific papers in 6 thematic fields (clusters)*

Each cluster includes certain papers (publications) that are identified by their entry numbers (act as unique identifiers). The exact papers that make up the respective clusters will be purposefully specified, as subsequently the same 42 papers will be also grouped by agglomerative hierarchical cluster analysis and the results of the two types of clustering will be compared accordingly.

The six groups consist of the following papers:

Group 1: 7, 10, 24, 87, 89, 94, 106

Group 2: 3, 33, 67, 82, 92, 100, 119, 101

Group 3: 35, 75

Group 4: 5, 19, 23, 29, 57, 59, 68, 107, 111

Group 5: 6, 17, 20, 38, 40, 41, 64, 65, 76, 95, 121

Group 6: 12, 25, 42, 60, 103

**2. Hierarchical clustering**

In contrast to flat clustering, *with hierarchical clustering one knows not only which object/document belongs to which group, but also what the relationship between the objects is within the cluster itself*, i.e. how close they are to each other. This is very important information that *allows the user to do further analysis and possibly move documents, resize clusters, split or merge clusters, and arrange documents within the cluster itself*. All this is impossible with flat clustering, because the user does not know how close the documents are to each other inside the cluster.

Hierarchical clustering is usually done by the so-called agglomerative hierarchical cluster analysis (*agglomerative hierarchical clustering*), which is implemented "from the bottom up". *At first, it treats all objects as independent clusters, then at each step it merges two of the existing (the two closest) clusters to form a larger one* (fig. 5). If it is not aborted, or there is no specified limit for cluster sizes or other clustering termination criteria, the process continues until all clusters are merged into one. There is also the opposite "top-down" approach (divisive clustering), where at the beginning all objects are in one common cluster and the algorithm starts iteratively

dividing it into subgroups. At each iteration, the largest cluster is split into two until a state is reached where each object is in its own cluster.

The algorithm for performing agglomerative hierarchical cluster analysis is presented in figure 5. The grouping is done as described above - from bottom to top. The algorithm uses the following more important data structures:

**`distMatrixClusters[clusterId1][clusterId2]`** – matrix of distances between clusters. Initialized with the matrix of distances between individual objects, because at the beginning each object forms its own cluster.

**`clusters[clusterId]`** – an array whose indices are the cluster identifiers and whose values are other arrays containing the individual documents belonging to the corresponding cluster.

**`calculateDistance()`** – function that calculates the distance between two clusters using one of the methods described below (nearest neighbor, farthest neighbor, or average distance).

```
----------------------------------------------------------------------
// distance matrix between clusters
// distMatrixClusters[clusterId1][clusterId2]
distMatrixClusters = distMatrix; // distMatrix - distance
// between individual documents

i=0;
for every document in the collection doc_i {
   clusters[i] = doc_i; i++;
}
nbClusters = number of documents for clustering;

do {
   // traverse all clusters (documents initially)
   // and look for the two closest ones to merge
   // minimum distance between clusters is initialized
   // with some large (random) value
   minDist = 1000;
   // closest_i and closest_j are the identifiers
   // of the two closest clusters
   closest_i = null;
   closest_j = null;
   // Finding the two closest clusters. For higher efficiency:
   // traverse only the upper right diagonal part
   for every cluster cluster_i from distMatrixClusters  {
      for every other cluster cluster_j, different than cluster_i {
         if (distMatrixClusters[cluster_i][cluster_j] < minDist)
            minDist = distMatrixClusters[cluster_i][cluster_j];
            closest_i = cluster_i;
            closest_j = cluster_j;
         }
      }
   }
```

```
   // after knowing which are the two closest clusters
   // closest_i and closest_j, they should merge. For that purpose
   // those with lower number absorbe the other with higher number.
   absorbing = min(closest_i, closest_j);
   absorbed = max(closest_i, closest_j);
   // merging
   clusters[absorbing] = clusters[absorbing]+clusters[absorbed];
   // deleting the absorbed cluster
   // from both the clusters[] and distMatrixClusters[] arrays
   delete clusters[absorbed];
   delete rows and columns in distMatrixClusters,
                              associated with absorbed;
   // after the absorbed cluster is deleted from distMatrixClusters[]
   // then the distance between the new larger (absorbing) cluster
   // and all the other clusters should be recalculated!
   for every cluster cluster_j in distMatrixClusters,
                                          different than absorbing {
      distMatrixClusters[absorbing][cluster_j]=calculateDistance(
              clusters[absorbing],clusters[cluster_j],distMatrix);
      distMatrixClusters[cluster_j][absorbing] =
                       distMatrixClusters[absorbing][cluster_j];
   }
   nbClusters--;
} while (nbClusters > 1)
-----------------------------------------------------------------
```

*Figure 5. An algorithm for implementation of agglomerative hierarchical clustering*

In this bottom-up movement, the algorithm generates a *connected graph* between individual clusters and the elements within them. Each edge in the graph is associated with a weight that indicates the *semantic similarity between the clusters it connects*. The generated graph is actually a binary tree, the graphical representation of which is called a *dendrogram* (fig. 2). It contains important information that can answer the questions: which object should be grouped with which other objects; in how many clusters the objects should be grouped in; how to arrange/order the individual clusters.

The hierarchical cluster analysis, unlike the k-means algorithm, does not work directly with the objects, but with the matrix of distances between them. In order to form it, it is necessary to calculate the similarities between all the objects using one of the previously discussed similarity measures. This will generate a similarity matrix which is easily converted to a distance matrix by subtracting from 1.

When calculating the distance between two clusters, different linkage methods (criteria) can be used. The most suitable are: the nearest neighbor method (*single linkage clustering*) - fig. 6; the furthest neighbor method, also known as *complete linkage clustering* - fig. 7; and the *average linkage clustering* method - fig. 8, where the distance is calculated as the average distance of all pairs of objects in the two clusters. Of course, one object must be from the first cluster and the other object from the second. The latter is also known as the between groups linkage method.

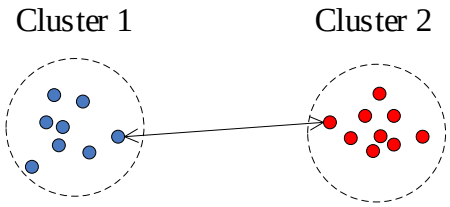



*Figure 6. Nearest neighbor method (single linkage clustering) for calculating the distance between two clusters*

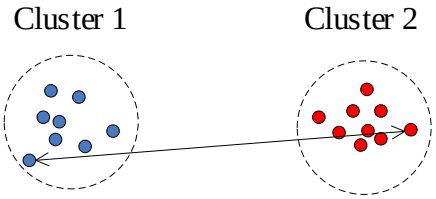



*Figure 7. Furthest neighbor method (complete linkage clustering) for calculating the distance between two clusters*

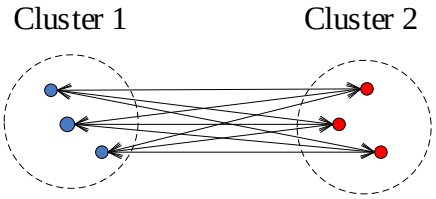



*Figure 8. Average linkage clustering method for calculating the distance between two clusters*

Which one of the three linkage methods is most suitable in a particular situation can be easily assessed by the *cophenetic correlation coefficient*. As can be seen from the dendrogram in fig. 2, any two objects, or clusters, are connected to each other by a *link having a specified height that indicates the distance between them*. This height is called a *cophenetic distance*. If the clustering is adequate, the links between the clusters in the binary tree should correspond to the distances in the matrix. So, to evaluate the accuracy of the chosen linking method, the correlation between the height of links (*cophenetic distances*) between individual objects and the distances between them in the input matrix must be calculated. Pearson's linear correlation coefficient is well suited for the purpose. The closer the correlation coefficient is to 1, the more accurate the clustering is, i.e. more closely matches the natural partitioning of the data. My own experimental studies, when grouping scientific papers in thematic fields, show that the average linkage leads to the best results and the highest cophenetic correlation.

The algorithm terminates when all clusters merge into one or when the cophenetic distance between clusters becomes larger than a predefined threshold. The user does not need to set the number of clusters in advance, and the algorithm is given the freedom to follow the natural partition of the data. Therefore, the clustering is more accurate, compared to flat clustering methods. The number of clusters can be determined at a later stage after analyzing the generated dendrogram or by setting a threshold distance when merging the clusters.

The agglomerative hierarchical clustering is used to group the same 42 scientific papers from the CompSysTech 2016 conference that were also clustered by the k-means algorithm in the previous section. This allows an objective comparison to be made between the two approaches. When merging clusters, the average linkage method was used to calculate the distance between clusters. Since the documents (papers) are described by a set of keywords selected from a taxonomy, the initial distances between them are calculated according to the proposed formula for calculation similarity between two sets of concepts in a common taxonomy. It actually calculates the similarity, but the distance is easily obtained as the 1-similarity. Grouping is done by a proprietary implementation of the algorithm (in php language). The results are presented in a colored dendrogram (fig. 9), which was generated by Matlab based on the distances between the clusters.

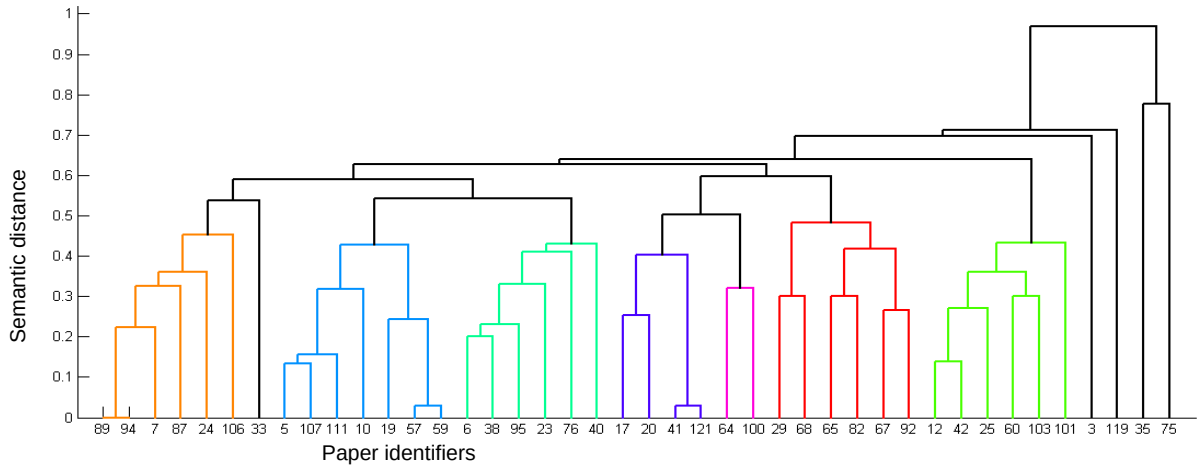



*Figure 9. Color dendrogram, showing the proposal of the agglomerative hierarchical clustering for grouping papers in CompSysTech 2016 conference (without the e-Learning section) in thematic fields and working sessions*

When comparing the dendrogram with the clusters resulting from the k-means algorithm, it seems that the clustering is indeed similar, but not identical. It could not be quite identical because agglomerative cluster analysis works directly with the distance matrix calculated by the proposed similarity measure, while the k-means algorithm additionally calculates a Manhattan distance on its row-vectors. Here again, it should be noted that, in general, the k-means algorithm does not work with the distances between documents, but with their feature vectors.

In the comparison, it is noticed that the orange group of papers (89, 94, …) in the dendrogram is almost identical to Group 1 obtained by the k-means algorithm. According to the latter, paper 10 should also fall into this group, although according to the dendrogram it is fine where it is. Similarly, the light blue group is also almost identical to Group 4 of the flat clustering. According to the k-means algorithm, paper 68 should also be included in this group, but it is quite far in the dendrogram. The combination of the first green group (6, 38, 95, …) as well as the purple one (17, 20, …) make up Group 5 obtained by the k-means algorithm. Probably, if papers were grouped not in 6, but in more groups, these two clusters could have been separate. However, the silhouette graphs show that with 7 and more groups, the misclassified papers are more. In both clustering approaches, the result grouping is similar, but agglomerative hierarchical cluster analysis provides a very important additional piece of information – the semantic distance

between all clusters and between documents within them. Based on this information, subsequently, human experts could further split or merge clusters, move and rearrange both the clusters and the documents within them. Thanks to all this, with the help of agglomerative cluster analysis, a more accurate and reliable grouping of documents could be obtained than with flat clustering methods and in particular the k-means algorithm. However, this statement applies mostly to cases where the number of grouped objects is relatively small and human experts would look at and modify the grouping. If the number of documents is very large, then the benefits of hierarchical clustering are devalued, because hardly anyone will do additional analysis and manual restructuring of the clustering.

### 3. Comparing alternative clusterings/groupings

In some situations, two or more groupings need to be compared with each other. Not only when using alternative settings (number of clusters, different distance metrics, etc.), but mostly for evaluation of the quality of the obtained groupings. Quality assessment can be done through *internal evaluation* and *external evaluation*. The internal evaluation takes into account how closely the clustering matches the input data and distances between documents, but completely ignores any subjective factors. Examples of internal evaluation are the *silhouette diagrams*, which report the quality of grouping in flat clustering methods, and the *cophenetic correlation coefficient*, in hierarchical clustering. With their help, one can determine the optimal number of clusters in flat clustering or the most suitable linkage method in hierarchical clustering, but both indicators cannot tell what the quality of the resulting clustering is, compared to subjective human judgment. In addition, the distances from each object to the center of the neighboring clusters, or the distances between individual clusters, are needed to calculate these indicators. Very often such data simply does not exist. They are available during the operation of the clustering algorithm, but afterwards (especially in flat clustering methods) such information is not stored.

External evaluation, unlike internal evaluation, *relies entirely on human experts to determine the accuracy of the resulting clustering* from a given automated method. Typically, *experts provide their own clustering, which is taken as a benchmark (reference)* and the resulting clustering is compared to it. Obviously, it is not possible to have additional information, such as distances between clusters and/or between documents with the benchmark, therefore internal evaluation indicators such as silhouette diagrams and cophenetic correlation coefficient are completely inapplicable.

There are metrics that allow evaluating *how well a grouping corresponds to another grouping*. They check whether each of the objects belongs to the same cluster in both groupings or not. As clusters have no default names, they are identified by the elements within them, i.e. metrics consider the relationships between all pairs of objects, and check whether these pairs of objects belong to the same or to different clusters in one and the other grouping. Yes, it leads to complication, but there is no other way. In this context, the four main indicators (TP, TN, FP, FN) have the following meaning:

TP (*true positives*) – number of pairs of objects that belong to the same cluster, in both the generated grouping A and reference grouping B.

TN (*true negatives*) – number of pairs of objects that belong to different clusters, in both the generated grouping A and reference grouping B.

FP (*false positives*) – number of object pairs that belong to the same cluster in the generated grouping A, but in different clusters in the reference grouping B.

FN (false negatives) – number of object pairs that belong to different clusters in the generated grouping A, but in the same cluster in the reference grouping B.

Подобно на метриките за оценка на върнатите резултати при търсенето на документи, комбинацията от посочените 4 показателя може да се използва за оценка на съответствието/подобието между две групирания (клъстеризации). За целта могат да се приложат: индексът на Rand (1), коефициентът на Jaccard (2) и индексът на Fowlkes–Mallows (3).

Similar to the metrics for evaluation of search results, a combination of the four mentioned indicators can be used to evaluate the correspondence/similarity between two groupings (clusterings). For this purpose, the following can be applied: the Rand's index (1), the Jaccard's index (2) and the Fowlkes–Mallows's index (3).

Rand's index:

$$RI = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Jaccard's index:

$$JI = \frac{TP}{TP + FP + FN} \tag{2}$$

Fowlkes–Mallows's index:

$$FMI = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \tag{3}$$

Rand's index is calculated in the same way as the accuracy of the search results. Since it counts the number of true negatives (TN) in its numerator, its value will always be very high, making it not sufficiently precise and informative. At the other end is the Jaccard's index, which completely ignores true negative relationships. This gives it high precision, but makes it super sensitive. If one large cluster in the generated clustering is split into two smaller ones in the reference one, the Jaccard's index would have an unduly low value, given that the documents are still clustered together, just that one large cluster is split into two smaller ones, due to some reason. The Fowlkes–Mallows's index represents the geometric mean between precision and recall, and gives more balanced results.

The *percentage of correctly classified documents* can be added to the evaluation metrics as well. A document is considered to be correctly classified, if it is assigned to the same cluster in the generated grouping, to which it belongs in the reference grouping.