Chapter 5

Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering is a method that forms clusters by merging the most similar existing clusters. The original similarity that this algorithm uses can be replaced by the hybrid similarity measures to include relational information. Experiments with both the original similarity and the hybrid similarities are done on the Cora dataset, a dataset with scientific papers. This should give a first indication whether using the hybrid similarities could lead to better results.

5.1 Introduction

In Chapter 2, it was explained that Knowledge Discovery in Databases (KDD) deals with the increasing demand of retrieving useful knowledge from rapidly growing databases. The most important step in the KDD-process is called Data Mining, and that is where the algorithm actually is doing its task. A data mining algorithm depends, amongst others, on the type of data it gets as input. One of the challenges of KDD is to address the problem that there are many different types of data to create algorithms for. It is even possible to combine different types of information in the same database, thereby creating a wide variety of different types of databases.

It is not feasible to create a different algorithm for every new combination of data types. Rather than doing that, it could be more beneficial to create a way to incorporate one type of information into a data mining algorithm that is created for another type of information. The method proposed in this thesis, the hybrid similarity measure, does exactly that.

Chapter 3 explained precisely the working of this method. The chapter showed that the hybrid similarity measure alters a similarity measure that is

based purely on the content of the elements in the database. It does so by regarding the neighborhood of the elements and the content-based similarities to them. The average of these similarities is treated as a new similarity and replaces the old similarity. Indirectly, the relational information is thus incorporated into a data mining algorithm that uses only content-based information.

One of the benefits of this hybrid similarity measure is that it is not necessary to invent new data mining algorithms for data that has both content-based and relational information. Other benefits were described in Chapter 4, where it was shown that there are always different types of noise or data variance in natural data sets, amongst others because of content variability. The hybrid similarity uses homophily and transitivity of similarity to make the data mining algorithm more robust to various types of noise.

In this chapter, the hybrid similarity measure will be implemented in an already existing data mining algorithm and used on real data to compare it with the original similarity. The data come from the Cora data set, which is presented in Section 5.2. The algorithm used is agglomerative hierarchical clustering, which is presented in Section 5.3. Section 5.4 presents the experimental setup in detail, after which Section 5.5 presents the results from the experiments. Finally, Section 5.6 presents the conclusions that can be drawn from these first results.

5.2 The Used Dataset

5.2.1 The Cora Dataset

The experiments are done on the Cora dataset, which was created by McCallum et al. [65]. They researched the possibility to use several machine learning techniques to automatically create a domain specific internet portal. Such a portal is an information gateway that often includes a search engine plus additional content. It enables the user to ask very specific, highly detailed queries on a certain subject.

Their process consisted of three steps. First, a web crawler searched for documents to add to the collection, using a reinforcement learning framework to decide which documents to add (see the survey by Kaelbling et al. [42]). After that, characteristic pieces of information were extracted from the documents using hidden Markov models (see the tutorial by Rabiner [78]). Finally, the documents were arranged into a hierarchical classification by means of an extended version of naive Bayes (see Lewis [58] or McCallum and Nigam [64]).

McCallum et al. demonstrated the working of these techniques by creating a portal for computer science research papers called "Cora." More than 50,000 papers are included in the database and available for keyword search. For 37,000 of these papers, the abstracts are also available. Besides that, all papers are placed in a computer science topic hierarchy with 70 classes at the leaves.

Finally, the citation links between these papers are available creating a big citation graph. This graph consists of many more papers, since it also includes all papers that are cited in the papers from the database, while the keywords or abstracts of these papers are not yet available.

The downside of this dataset is that it may not necessarily be correct. Since it has been created automatically, papers can be assigned to the wrong class. This makes it difficult to evaluate the result of a data mining algorithm. The fact that things have gone wrong, stems from the fact that some papers have more than one abstract or class assigned to it. From this, it can be assumed that more errors are made, but it is not possible to know how many, and, more importantly, which. For the purpose of evaluating data mining algorithms, this is not a big problem. The "ground truth" is defined as the information that is in the database. This is the reference point that algorithms are compared to.

5.2.2 The Created Subsets

The Cora dataset is created with a web crawler. Generally, a web crawler starts with one paper and from there tries to expand its scope by adding papers. Such a web crawler is sometimes called a spider, since it creates its web by starting in the center and then expanding it. When the crawler stops, the neighborhoods of elements in the central area are added to the dataset completely. In the same time, the neighborhoods of the elements from the areas on the edge (the relatively new papers) are not yet completely added to the dataset.

This means, of course, that the older regions of the dataset are better usable for data mining tasks than the newly added regions, which are yet to be fully processed. Unfortunately, it is difficult to determine which regions are complete and which are not yet. However, the connectivity of a particular region could show its completeness. For regions that have low connectivity, it could be assumed that not all the links that are there in the real world are already established for the dataset.

In an attempt to solve this problem, for this experiment, several subsets of the Cora dataset were created. All classes were sorted on connectivity. Here, not only the ratio between edges and vertices was taken into account, but also things like the amount of vertices that are connected to only one other vertex, and other comparable attributes are considered. Then, a first subset was created, named Cora-1, with the eight best connected classes. This dataset was extended four times by adding more classes. An overview of this can be seen in Table 5.1.

In this table, it can be seen that the average degree rises despite the fact that the connectivity of the classes that are added is supposed to be less than the classes that are already in a smaller subset. This can be explained by the edges that connect vertices from classes that were already in the subset with edges from classes that are newly added. In relation to the amount of vertices,

dataset	number of classes	number of papers	number of edges	average degree	graph density
CORA-1	8	720	2244	6.23	0.0087
CORA-2	17	1725	6093	7.06	0.0041
CORA-3	24	2523	10022	7.94	0.0032
CORA-4	31	4692	18634	7.94	0.0017
CORA-5	45	7917	39317	9.93	0.0013

Table 5.1: Overview of the five subsets from the Cora dataset and some of their characteristics. The 'average degree' is calculated as $\frac{2|E|}{|V|}$ and the 'graph density' is calculated with use of the definition by Coleman and Moré: $\frac{2 \cdot |E|}{|V| \cdot (|V|-1)}$ [14]

.

relatively more edges are added to a subset. However, the table shows that the overall graph density decreases. This becomes clear by using the definition of Coleman and Moré: $\frac{2 \cdot |E|}{|V| \cdot (|V|-1)}$ [14], which is basically the ratio between the edges that are in the graph and all the edges that could be in the graph (e.g. the amount of edges a complete graph with n vertices has).

5.3 The Method

One of the simplest ways of clustering elements is by using agglomerative hierarchical clustering. While the field of taxonomy already used similar techniques for a longer time, agglomerative hierarchical clustering was formalized by Johnson in 1967 [41]. For clustering n elements, it only needs a $n \times n$ similarity matrix S where an element s_{ij} denotes the similarity between elements v_i and v_j. Information about the elements is not necessary. It works as follows:

1. Assign every element to its own cluster.
2. Determine which clusters are most similar.
3. Merge these two clusters to a single cluster.
4. Update all similarities from this new cluster to the others.
5. Repeat steps 2 through 4 until all elements are in the same cluster.

The result of the algorithm is not a single, best clustering, but rather a hierarchy of found clusters. Normally, this hierarchy is represented in a *dendrogram*, which is a tree where all the leaves are the single elements and the root is the final cluster with all elements in it. The nodes in between represent one of the merges from step 3. Every level of this dendrogram can be considered a found clustering. An example of such a clustering is shown in Figure 5.1.

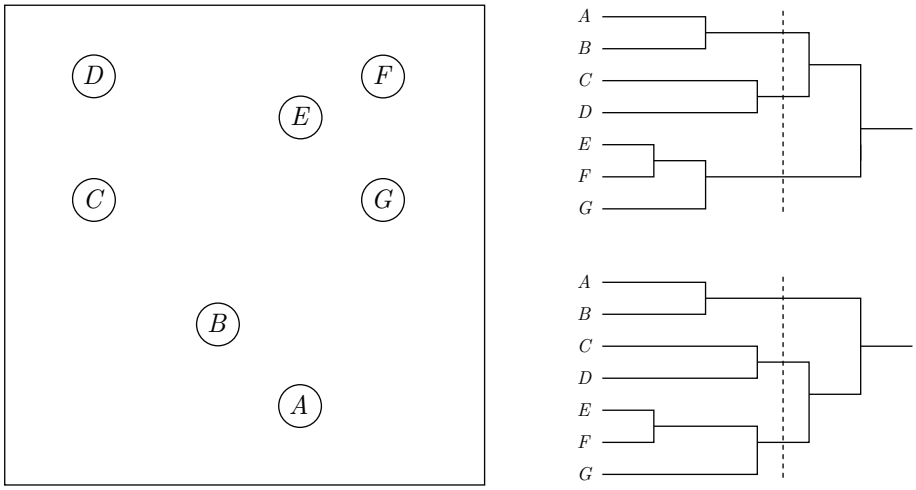

Figure 5.1: On the left are some elements that will be clustered with agglomerative hierarchical clustering. Here, instead of a similarity a distance measure is used. The distance between two elements is the physical distance in the drawing. On the right, two possible dendrograms are drawn. The top one is created by using single linkage and the bottom one is created with complete linkage. Both are created from left to right and the horizontal position shows when two clusters are merged during the process. At every position, one can cut the dendrogram to see a possible clustering. The dotted line is an example of such a cut. In both cases, it is of the clustering $\{\{A, B\}, \{C, D\}, \{E, F, G\}\}$. Before this dotted line, both methods produced the same clusters, but not necessarily in the same order. After the dotted line, the results start to differ. With single linkage, $\{A, B, C, D\}$ is formed, while with complete linkage, $\{C, D, E, F, G\}$ is formed. Both clusters do not appear anywhere in the other process.

The precise working of the algorithm, and the dendrograms resulting from that, depend for a big part on the way that the similarities are updated in step 4. There are several ways of doing that, each with their own benefits and downsides. The three most used methods are single linkage, complete linkage and average linkage.

Single linkage was originally defined by Florek et al. [26], and later reinvented by McQuitty [66] and Sneath [86]. In *single linkage*, the similarity between two clusters is the similarity of the two most similar of their elements that each come from a different cluster. More formally, given elements a and b, and clusters A and B, with $a \in A$ and $b \in B$, then the similarity between A and B, $\mathcal{S}(A, B)$, is defined by $max\{\mathcal{S}(a, b)\}$.

Complete linkage originally was defined by Sørensen [89]. It can be considered the opposite of single linkage. With *complete linkage*, the similarity between two cluster is the similarity of the two least similar of their elements that each come from a different cluster. Formalized with the same elements as before, $\mathcal{S}(A, B) = min\{\mathcal{S}(a, b)\}$.

Trying to strike the golden mean between single linkage and complete linkage, Sokal and Michener [88] came up with *average linkage*, where the similarity between two clusters is defined as the average of all similarities between an element from one cluster and an element from the other. It is formalized with the same elements as previously used: $\mathcal{S}(A, B) = \frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} \mathcal{S}(a, b)$.

The clustering hierarchies found can be very different when a different linkage method is used. Figure 5.1 shows an example of different results depending on the linkage method. Each method has its own good and bad sides. In an extensive study, Milligan compared several linkage methods [67]. The problem of single linkage is that there is a great chance of chaining. This means that elements are added to a cluster one at a time, because they are similar to the most recently added element. This is done without taking into account the similarity compared to the other elements in that cluster. The result can be long chains of elements that are all in the same cluster, but are spread throughout the entire domain. This problem is known as *chaining*.

The problem of chaining is solved by using complete linkage. Complete linkage has the tendency to form clusters of roughly the same diameter, and thus, the problem of chaining does not appear with complete linkage. Unfortunately, it can be heavily distorted by outliers. Since the distance is determined by the pair of least similar elements, it can be expected that at some point during the execution of the algorithm, the outliers often will be the elements determining the similarity between clusters.

Average clustering suffers much less from chaining than single linkage, and it is less sensitive to outliers than complete linkage. It is therefore understandable that, from these three linkage methods, it is the most popular. Still, there can be many situations where single or complete linkage can lead to better results than average linkage. There are also other linkage methods, like Ward's minimum-

variance method (that tries to minimize the variance between elements in a cluster [98]), and the centroid method by Sokal and Michener [88] (that uses the centroids of the clusters to calculate the similarities between them).

5.4 Experimental Setup

We briefly recall the formal definitons from Section 3.3. All experiments are done on a dataset $D = (V, E, \alpha, \lambda)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices or elements, $E \subseteq V \times V$ is the set of edges, $\alpha : V \to \mathcal{A}$ a function that assigns to any vertex v of V an "annotation," and $\lambda : V \to \mathcal{L}$ a function that assigns to any vertex v of V a "label." Also, the neighborhood of a vertex v is defined as $\mathcal{N}(v)$, where $\mathcal{N}(v)$ is the set of vertices that are connected to v with an edge. So with two vertices $v, w \in V$ this means that $w \in \mathcal{N}(v) \Leftrightarrow (v, w) \in E$.

In this case, the dataset is any of the five subsets taken from the Cora dataset, as described in Section 5.2.2. The vertices or elements in the dataset are the papers, and the edges are the citations. (i.e. the edge $(v, w) \in E$ if and only if v cites w or w cites v). The labels are the classes that were assigned to the papers by the creators of the Cora dataset.

The Annotations

The space of annotations was left open in Section 3.3, but now it needs to be well defined. The annotations are derived from the words that appear in the abstracts of the papers. For this, first, it needs to be determined what all the useful words in all the abstracts are. This is the set $P = \{p_1, p_2, \ldots, p_m\}$, and is created by taking every word that appears in at least two different abstracts. From this set P, a set of known stop words is removed. Stop words are words that are so general that they do not have any power to make a distinction between different texts. Examples of stop words are 'the', 'is, 'would', and so on. For Cora-1, this set P contains 2924 words. As the subsets grow bigger, so do the corresponding sets of P up to 9082 words for Cora-5.

Using this set of useful words P, the annotation a_i of a vertex v_i can now be defined as a vector $\vec{a_i} = (a_{i0}, a_{i1}, \ldots, a_{im})$ where a_{ij} is 1 if p_j appears in the abstract of v_i, and 0 otherwise. These vectors can be considered the content of the elements and are used to calculate the content-based similarity.

The Similarity Measures

The content-based similarity is defined solely on the annotations of the elements. Since these annotations are vectors that state what words appear in the abstract, a good similarity should return a high value for a similarity between two elements when they have many words in common. From a vector perspective, when two abstracts have relatively many words in common, their

annotation vectors will point roughly in the same direction, and thus, the angle between them is relatively small. Therefore, the cosine of the angle between two annotation vectors can be used very well as a similarity measure, as it gives 1 when they point in the exact same direction and 0 when they are perpendicular. Since, in this case, the elements of the vectors cannot have a negative value, the cosine cannot be lower than 0.

More formally, given two elements $v_i, v_j \in V$ and their annotations defined by the function α, so that $\alpha(v_i) = \vec{a_i}$ and $\alpha(v_j) = \vec{a_j}$, then the content-based similarity between v_i and v_j is defined by:

$$\mathcal{S}_{content}(v_i, v_j) = \cos \angle(\vec{a_i}, \vec{a_j}) = \frac{\vec{a_i} \cdot \vec{a_j}}{|\vec{a_i}| \cdot |\vec{a_j}|} \tag{5.1}$$

With the content-based similarity defined, the contextual similarity and the combined similarity can be calculated easily. Following the definitions from Section 3.4.2, the contextual similarity is defined by:

$$\mathcal{S}_{contextual}(v_i, v_j) = \frac{1}{2} \cdot \left(\frac{\sum_{u \in \mathcal{N}(w)} \mathcal{S}_{content}(v, u)}{|\mathcal{N}(w)|} + \frac{\sum_{u \in \mathcal{N}(v)} \mathcal{S}_{content}(w, u)}{|\mathcal{N}(v)|} \right) \tag{5.2}$$

Finally, the combined similarity is defined following the definition from Section 3.4.3:

$$\mathcal{S}_{combined}(v, w) = c \cdot \mathcal{S}_{content}(v, w) + (1 - c) \cdot \mathcal{S}_{context}(v, w) \tag{5.3}$$

with $0 \leq c \leq 1$. c determines the weight of the content-based similarity in the combined similarity. Since no significant difference was found in performance, c was chosen as $\frac{1}{2}$.

The Linkage Method

With these similarities, the elements in the dataset can be clustered. When doing so, a linkage method needs to be chosen. In this case that is average linkage. For the experiments in this chapter, the exact linkage method is not important. The goal of the experiments is not to find the best clustering, but to see whether the original method can be improved.

The Evaluation Method

To compare different clustering methods, an evaluation method is needed. For this, Larsen and Aone's F-score [56] was used. It is a method to compare one clustering to another. A higher F-score means that those two clusterings are more similar to each other than two clusterings that have a lower F-score. A score of 1 means they are completely identical. For the elements in the

Cora dataset, all the labels are known and also can be seen as a clustering. This clustering is, of course, the ultimate goal for a clustering algorithm. The higher the F-score between the found clustering and the clustering based on the labels, the better the algorithm works. Therefore, to facilitate the distinguishing between the two, a cluster in the clustering based on the labels will be called a label class.

To compare the found clustering with the classes, two aspects of F-score are important: precision and recall. Considering a cluster C from a clustering $\mathcal{C}$ and a label class L from the set of all labels $\mathcal{L}$, the *precision*, $P(C, L)$, is the amount of elements from label class L in cluster C, divided by the total amount of elements in C. The *recall*, $R(C, L)$, is the amount of elements from label class L in cluster C, divided by the total amount of elements in L. Therefore, every class-cluster combination has its own precision and recall. Now, the F-score can be calculated as:

$$F\text{-score}(C, L) = \frac{2 \cdot P(C, L) \cdot R(C, L)}{P(C, L) + R(C, L)} \tag{5.4}$$

This does not yet give the F-score for the entire clustering in comparison to the complete labeling. To do so, for every label class, the best fitting cluster needs to be found. This is the cluster that has the best F-score for that class. Then, of all these F-scores, the weighted average (with regard to the size of the label class) is taken. Thus, the F-score for a clustering $\mathcal{C}$ with regard to a labelling $\mathcal{L}$ can be calculated with:

$$F\text{-score}(\mathcal{C}, \mathcal{L}) = \frac{1}{n} \cdot \sum_{L \in \mathcal{L}} (|L| \cdot \max\{F\text{-score}(C, L) \; \forall C \in \mathcal{C}\}) \tag{5.5}$$

The agglomerative hierarchical clustering starts with a clustering that consists of n clusters containing one element, and then stepwise merges these clusters until only one cluster remains. Every step can be seen as a clustering and thus at every step, the found clustering can be compared with the classes. During these experiments, the F-score was calculated every 10 steps to monitor its development during the procedure. The results, when using the original, content-based similarity, can be compared with the new hybrid similarities using these F-scores.

5.5 Results

The main goal for these experiments is to compare the original, content-based similarity with the new hybrid similarities: the contextual similarity and the combined similarity. This can be done best by comparing the F-scores throughout the clustering process. They can be seen in Figures 5.2 and 5.3 for all the five subsets.

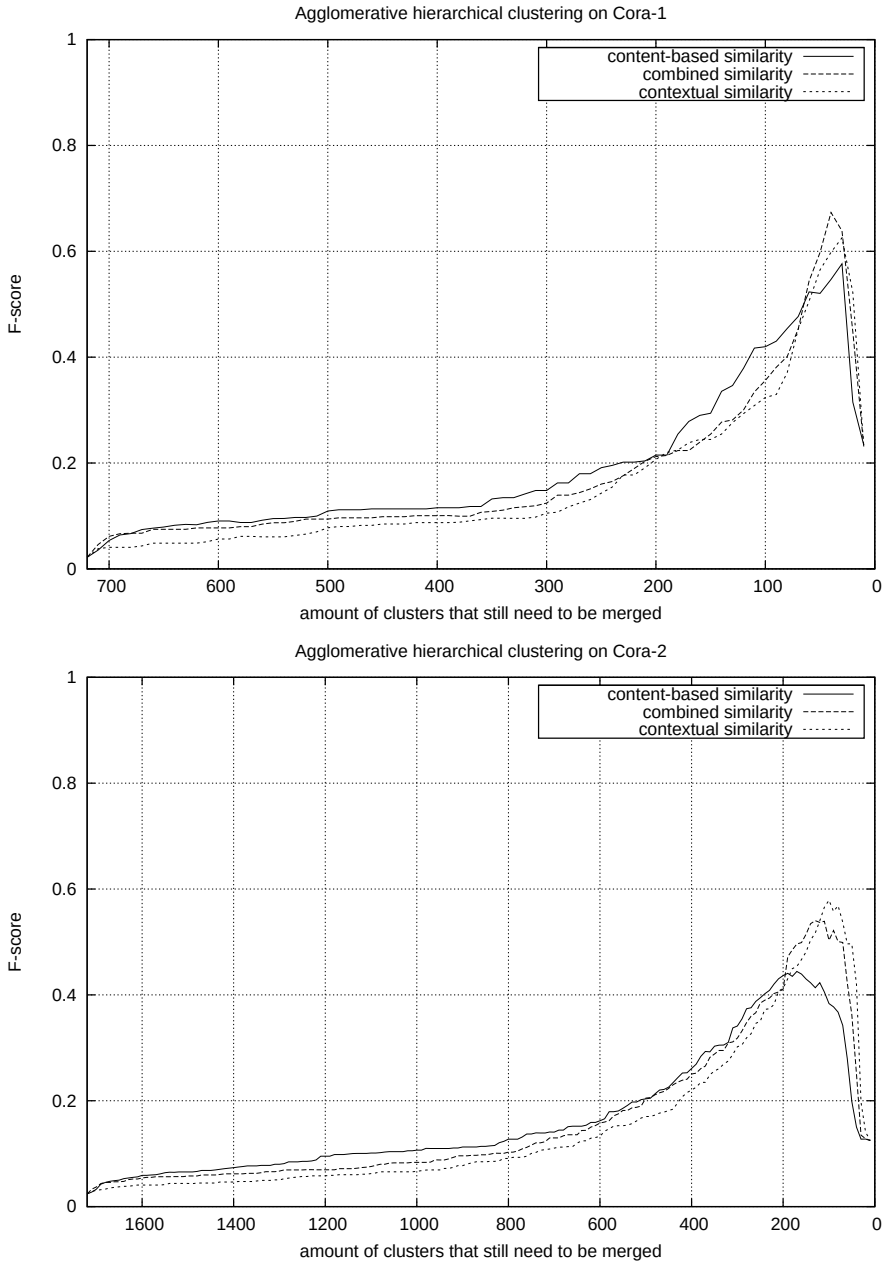

Figure 5.2: F-score for all the found clusterings during the process of agglomerative hierarchical clustering with average linkage on CORA-1 (top) and CORA-2 (bottom).

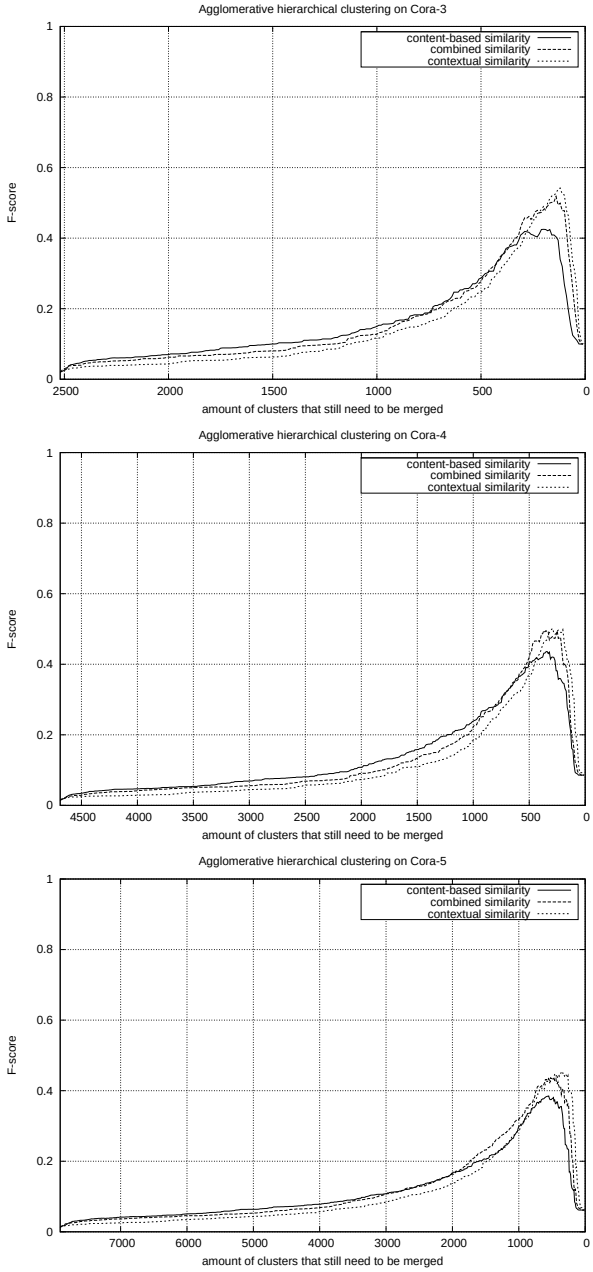

Figure 5.3: F-score for all the found clusterings during the process of agglomerative hierarchical clustering with average linkage on respectively CORA-3 (top), CORA-4 (middle), and CORA-5 (bottom).

Every line in these graphs shows the same characteristics; they start very low, rising slightly. As the clustering continues, the F-scores rise faster and faster, until a maximum is reached and the line drops rapidly. Also in all the graphs, the three similarities behave more or less in the same way with respect to each other. The content-based similarity performs better than the two hybrid similarities at first. Then, it reaches its maximum before the hybrid similarities do, and the F-score of the content-based similarity starts to drop. While this happens, the F-scores of the contextual and combined similarities continue to rise and they reach a significantly better score.

Another aspect that can be regarded is the found dendrogram. Figures 5.4 through 5.6 show these for respectively the content-based similarity, the combined similarity, and the contextual similarity. Due to lack of space, the complete dendrograms cannot be shown. The numbers on the leafs represent the amount of elements that are still on that branch. The main thing that can be noticed is that at the end of the clustering process, there are still many singletons that need to be merged. These are probably outliers that are difficult to cluster. Still, it looks like the hybrid similarities have a slightly better spread dendrogram than the content-based similarity. This could explain the results for the F-score.

5.6 Conclusion

In a relational context, one may want to cluster objects based on both their content and the relationships between them, with the latter being indicated by a graph. We have proposed a method for adapting the distance or similarity measure in such a way that relational information is inserted. Experiments on the Cora data set show that this more informed similarity measure leads to better clustering results when it is plugged into a standard clustering procedure.

It is difficult to say anything about the found F-scores in general. Since the used subsets are custom-made for these experiments, the results cannot be compared with results from other experiments. What can be said about the results however, is that the found dendrograms are unbalanced. As this is probably due to outliers, a better handling of these outliers might improve the F-scores even more.

However, the used similarities during these experiments can still be compared easily. A first conclusion can be that it seems that adding relational information does enhance the performance of this clustering procedure on these data sets. The content-based similarity outperforms the others from the start of the clustering for a long time, but the hybrid similarities overtake it when it matters most: when the amount of clusters starts to approach the amount of classes. This better performance can also be seen to some extent in the dendrograms for the hybrid similarities that appear to be slighty less unbalanced than that of the content-based similarity.

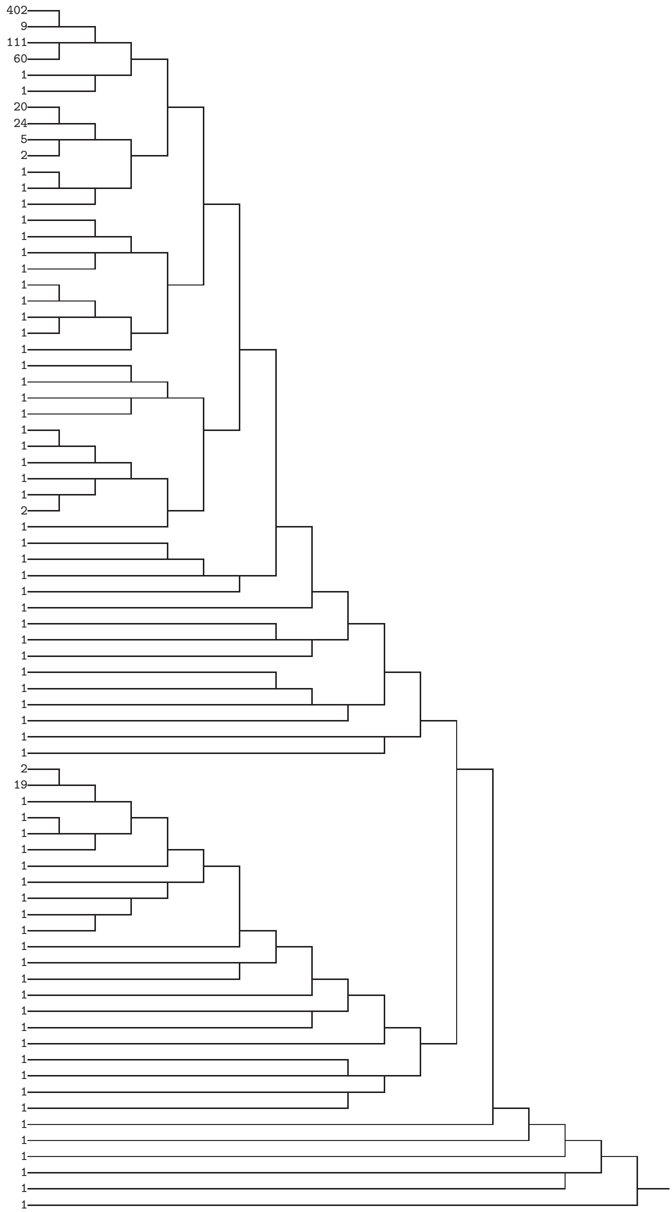

Figure 5.4: Dendrogram for the result on CORA-1 with the content-based similarity.

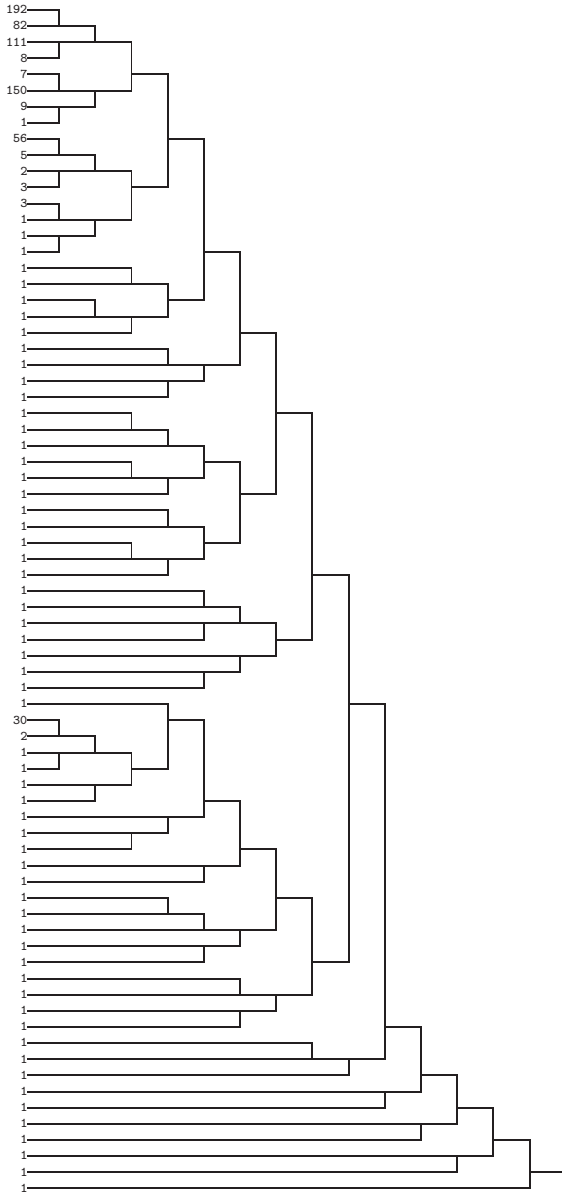

Figure 5.5: Dendrogram for the result on CORA-1 with the combined similarity.

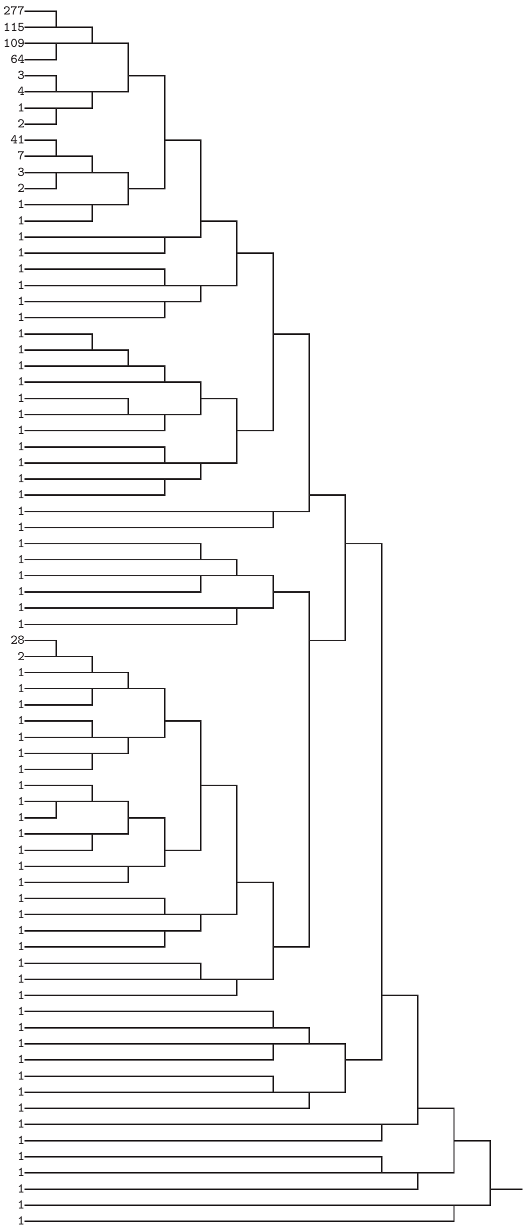

Figure 5.6: Dendrogram for the result on CORA-1 with the contextual similarity.