Chapter 7

KNN-Classification

Classification aims to determine to which class a certain element in the dataset belongs. K-nearest neighbor classification does this by regarding the k most nearest neighbors. The hybrid similarities can be used to determine which neighbors are the nearest. In this way, relational information is inserted into this method. Experiments on real life data show that using the hybrid similarities can lead to better results.

7.1 Introduction

Previous chapters showed how the hybrid similarity measure can improve standard clustering algorithms like agglomerative hierarchical clustering (Chapter 5) and k-means and k-medoids (Chapter 6). Since there are other types of data mining tasks, it is interesting to see if the hybrid similarity measure can also improve any of those. A likely candidate would be classification. In this chapter it is shown how the hybrid similarity measure can be used to improve k-nearest neighbor classification, KNN-classification.

The rest of this chapter is organized as follows. Section 7.2 explains the working of KNN-classification in detail. Section 7.3 explains how the hybrid similarity measure can be used to improve KNN-classification. Section 7.4 explains the experimental setup and the results from those experiments are reported in Section 7.5, after which Section 7.6 concludes this chapter.

7.2 The Method

There are many different classification techniques. An overview of the most important of them can be found in Section 2.2.1. Amongst them is k nearest neighbor classification, or KNN-classification. It works according to a very simple

and basic principle: To determine the label of an element, choose the one that is most prevailing amongst the k most similar elements that are labeled. It was originally proposed by Fix and Hodges [25] and in the mean time, a number of applications have been proposed, for example by Cover and Hart [18] and Han et al. [31].

More formally, considering a set of elements V, the algorithm will predict the label for an element $v \in V$ of which this label is still to be determined as follows:

1. For every element $u \in V$ with $u \neq v$ and $\lambda(u)$ is known:
2. Calculate the similarity between v and u: $\mathcal{S}(v, u)$.
3. Create the set K of size k with the property that for all $u \in K$, $w \notin K$, we have: $\mathcal{S}(v, u) > \mathcal{S}(v, w)$.
4. For every label $\ell \in \mathcal{L}$:
5. Calculate the contribution of ℓ: $C(\ell) = \sum_{u \in K} 1$ where $\lambda(u) = \ell$.
6. Now, $\lambda(v) = \ell$ where $C(\ell)$ is maximal for all $\ell \in \mathcal{L}$.

To create a good classifier, it needs to be trained on data where the labels are known. To have some sort of estimation of how good the classifier is, it needs to be tested. One problem that can occur in the training fase is overfitting. This means that the classifier is specialised in classifying items from the training set, but it does not perform too well on other examples. It is therefore important that the test dataset and the training dataset are disjoint. In this way, the amount of overfitting can be measured. This process is called cross-validation, and there are several tactics used for the process.

With k-fold cross-validation, the dataset is partioned into k random, disjoint, subsets. This k is not to be confused with the k from k nearest neighbor classification. One subset is used as the test dataset, and the others are used as training dataset. This process is repeated for every subset. The best value to choose for k remains unclear, as can be read in the book by Geisser [28]. A special case of k-fold cross-validation is when k equals the number of elements in the dataset. This is also known as leave-one-out cross-validation. Here, the entire dataset, minus one element, is used to predict the value for this one element. This is done for every element. Other examples of cross-validation include repeated random sub-sampling validation where the dataset is repeatedly and randomly split into a training set and a test set.

Cross-validation can thus be used to help determine the amount of nearest neighbors; the k in 'k nearest neighbors.' It can be seen easily that different values for k can lead to different results. This is illustrated in Figure 7.1. A low value for k makes it more susceptible to noise, since every incorrectly labeled element has a great influence on all elements it is close to. A high value for

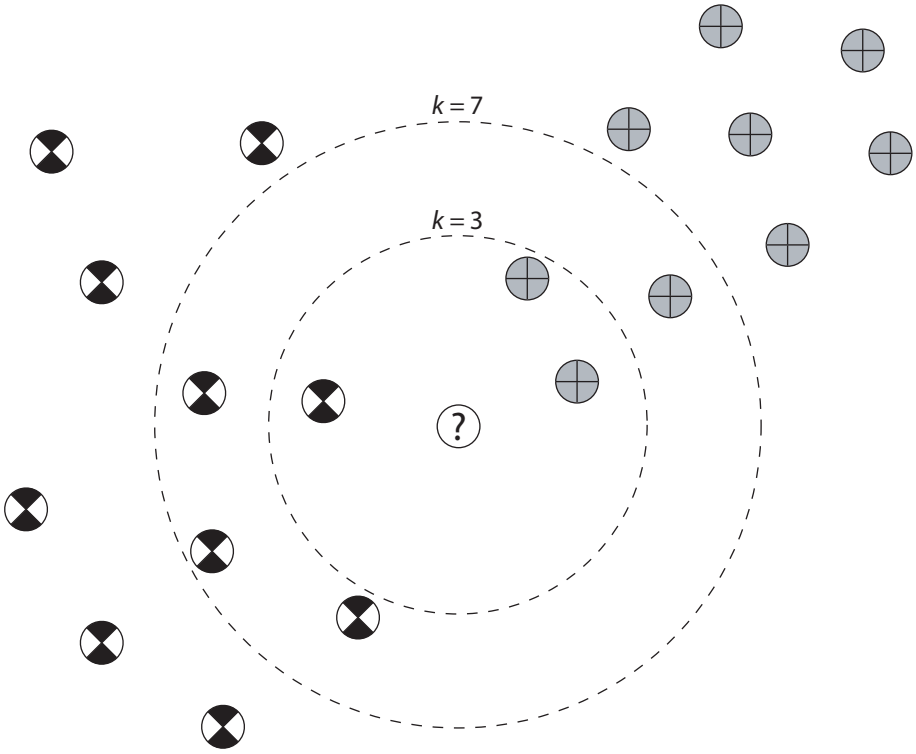

Figure 7.1: Illustration of the working of KNN-classification. The element with the question mark needs to be classified. The circles denote the areas covered for different values of k.

k makes it more difficult to draw distinctive boundaries between classes. In binary classification (where there are only two classes) it is best to choose k to be an odd number, since it will prevent ties.

When there are multiple classes, the user can choose between plurality voting or majority voting. In the first case, the class that appears most is chosen, while in the second case, a class is chosen when it appears more than half of the time amongst the k nearest neighbors. When using majority voting, there is a possibility that it is impossible to assign a class to an element. On the other hand, every class that will be assigned has a high confidence. This is not always the case with plurality voting, where it is possible that there are many different labels amongst the k nearest neighbors, and that the one that appears most still does not appear often.

It is also possible to assign a weight to the label of each neighbor according to their similarity to the element being classified. This would mean that neighbors

that are more similar have a higher influence on the label to choose. When the algorithm uses distances or dissimilarities, a monotonicly decreasing function can be used, like the inverse, one minus the distance, or the reciprocal of an exponential function.

A problem that can occur when using KNN-classification, is that classes that appear more frequently, tend to dominate the predictions, even more than their relative frequency would justify. Using weighted voting and keeping k low could prevent this from happening.

In case KNN-classification is used on a partially labeled dataset, it is possible to use the newly labeled elements to predict the label for another element. In this case it is good to keep track of the confidence of a predicted label. This can be used as a weight during the voting.

7.3 KNN-Classification and Hybrid Similarities

KNN-classification uses a similarity measure to calculate the similarity between elements. In this way, it is possible to determine which k elements are closest. Normally, the data mining algorithm would use a content-based similarity for this. When there is also relational information in the dataset, the hybrid similarities could be used to replace the content-based similarities and thus include the relational information in the algorithm.

In order to correctly use the hybrid similarities, it needs to be ascertained that they can be implemented without causing any malfunctions. As we have seen in Chapter 6, implementing the hybrid similarities in k-means was not straightforward. The similarity measure of this algorithm was not limited to similarities between elements only. It also uses the similarities between elements and some prototype that is not in the dataset. If KNN-classification only uses similarities between elements, and no other similarities, the hybrid similarities can be implemented without any further adaptions to the original algorithm.

This is indeed the case. The algorithm has two main steps: first, determine which are the k nearest neighbors, and second, determine which is the label that needs to be assigned to it. When determining which neigbors are the nearest, the algorithm computes the similarities between the element that needs to be labeled, and all other elements. After that, the k most similar elements are chosen. For this, only similarities between elements are needed, and so the hybrid similarities can be used instead of the content-based similarities.

When determining which class is the most common among these k elements, different tactics are possible. When the labels are counted unweighted, the similarities are not used, and thus, for this step, it does not matter whether the hybrid similarity is used or not. When the most common label is decided with votes that are weighted with regards to their similarity, once again, only similarities between elements are used. Therefore, here, it is also possible to use the hybrid similarities as weights to decide the label.

All in all, there does not seem to be a problem to implement the hybrid similarities. This would mean that it should be able to use them and in this way insert relational information in an algorithm that only used content-based information so far.

7.4 Experimental Setup

The experiments are done on subsets of the Cora data set by McCallum et al. [65], as described in Section 5.2. For these subsets, the datasets are defined following the definitions from Section 3.3, as described in Section 5.4.

Now, the contextual similarity can be calculated using (3.3), and the combined similarity can be calculated using (3.4). The results will be compared with the results from using the original, content-based similarity. In this way, it is possible to investigate whether the results of KNN-classification can be improved with the hybrid similarities.

The experiments are done with k ranging from 1 to $n-1$. Choosing a label for an element is done with plurality voting. The votes are weighted according to their relative similarity with regards to the element of which the label is to be determined. The experiments are done with the 'leave one out'-principle. In this case all labels are known and the label for one particular element is predicted by regarding the labels of the k closest elements and compared with the original value of the label of that element. Repeating this procedure for all elements will come to a percentage of correctly predicted labels for that combination of data set, similarity measure and k. These results are all compared.

7.5 Results

In Section 7.3, we hypothesized that the hybrid similarities could improve KNN-classification. In order to see if this is the case, experiments were done on five subsets of Cora using 'leave one out'-cross validation. During these experiments, k ranged from '1' to '$n-1$.' Figure 7.2 shows the ratio of correctly predicted labels for the subset CORA-1. The top picture shows all results, and the bottom picture zoomed in on the range of k from 1 to 40, to show this part of the graph in more detail. It is clear that the hybrid similarities outperform the original, content-based, similarity. Also, the contextual similarity outperforms the combined similarity for k greater than 4.

The experiments on the other subsets show similar results. Figure 7.3 shows these results for CORA-2 through CORA-5, zoomed in on k from 1 to 40. Here, it shows that the hybrid similarities outperform the original similarity. It also shows that for bigger k, the contextual similarity outperforms the combined similarity. Table 7.1 shows the best score for any similarity and any dataset, and for which k this score was acquired. Here, it shows that the best scores

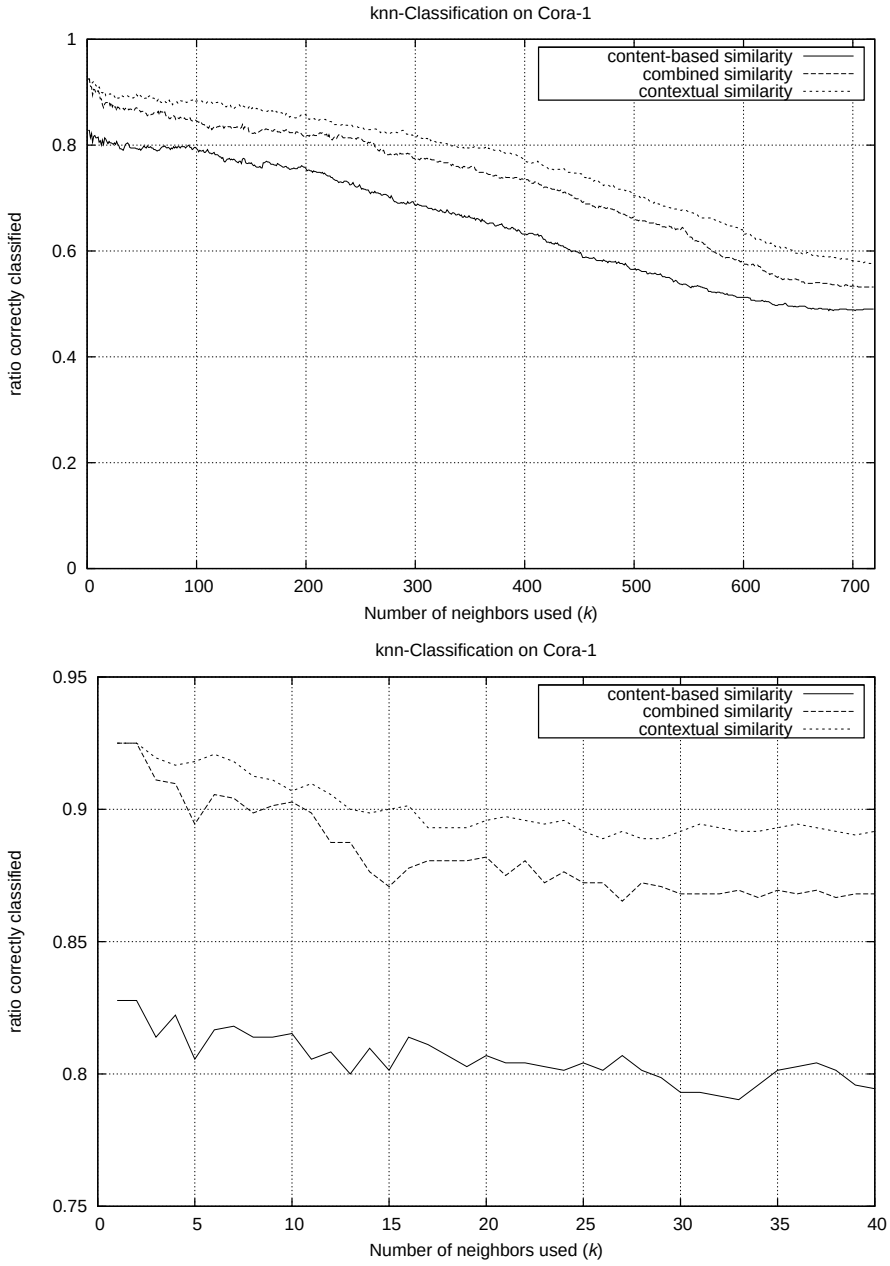

Figure 7.2: Ratio of correctly predicted labels for Cora-1. The upper picture shows the results for k is 1 to $n-1$ (where $n = 720$), and the bottom picture is zoomed in on k from 1 to 40.

Dataset	Content-based		Contextual		Combined	
	Ratio	k	Ratio	k	Ratio	k
Cora-1	0.828	2	0.925	2	0.925	2
Cora-2	0.747	10	0.881	3	0.882	3
Cora-3	0.680	5	0.823	11	0.822	3
Cora-4	0.685	12	0.815	7	0.817	4
Cora-5	0.655	14	0.773	7	0.779	5

Table 7.1: Best found scores for the three similarities on the five subsets of Cora. Also the k for which this score was acquired is given.

found for the two hybrid similarities do not differ much, but the improvement of performance with regards to the content-based similarity is significant. It is more difficult to draw any valid conclusions about the value of k for which the similarities perform best. It could be argued that the content-based similarity reaches its best performance at higher values for k than the higher similarities. Nonetheless, there are only fifteen cases and with several exceptions amongst them, this conclusion might be a bit too farfetched.

7.6 Conclusion

When a data mining tool is used on a dataset that has both content-based and relational information, it is preferred that this data mining tool could explore the entire data space. While many data mining tools are designed for only one such type of data, the hybrid similarities could be used to insert relational information into existing data mining tools that were designed only for content-based information. It has been shown that for various clustering algorithms, these hybrid similarities indeed lead to better results.

Despite its simplicity, KNN-classification is used widely as a classification technique. The core of the algorithm is the use of a similarity measure to determine which k elements are most similar to the one that is to be classified. Normally, it uses a content-based similarity, but instead of that, the hybrid similarity also can be used. In this chapter we have presented experiments that showed that using the hybrid similarities on KNN-classification can lead to better results than using a similarity measure that is based on the content of the elements alone.

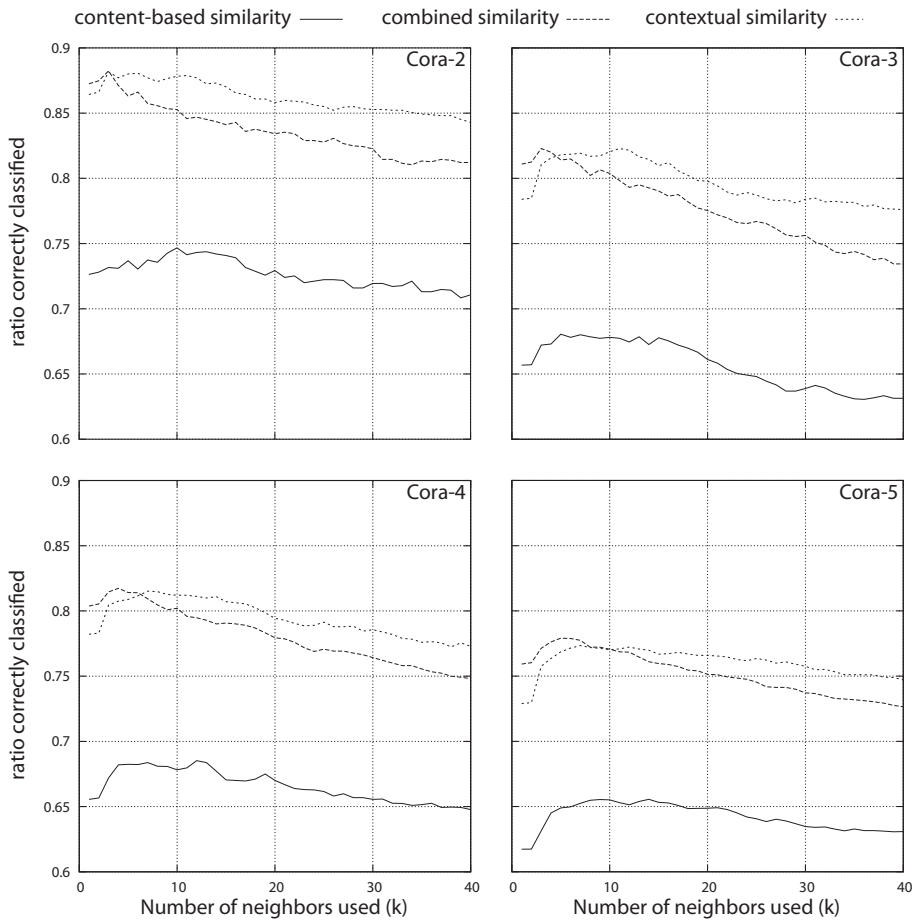

Figure 7.3: Ratio of correctly predicted labels for Cora-2, Cora-3, Cora-4, and Cora-5. They are all zoomed in on k from 1 to 40.