# K-Means Based Approaches to Clustering Nodes in Annotated Graphs

Tijn Witsenburg[1] and Hendrik Blockeel[12]

[1] Leiden Institute of Advanced Computer Science, Universiteit Leiden
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
tijn@liacs.nl, blockeel@liacs.nl
[2] Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven, Belgium
hendrik.blockeel@cs.kuleuven.be

**Abstract.** The goal of clustering is to form groups of similar elements. Quality criteria for clusterings, as well as the notion of similarity, depend strongly on the application domain, which explains the existence of many different clustering algorithms and similarity measures. In this paper we focus on the problem of clustering annotated nodes in a graph, when the similarity between nodes depends on both their annotations and their context in the graph ("hybrid" similarity), using $k$-means-like clustering algorithms. We show that, for the similarity measure we focus on, $k$-means itself cannot trivially be applied. We propose three alternatives, and evaluate them empirically on the Cora dataset. We find that using these alternative clustering algorithms with the hybrid similarity can be advantageous over using standard $k$-means with a purely annotation-based similarity.

**Keywords:** clustering, graph mining, similarity measure, $k$-means

## 1 Introduction

### 1.1 Clustering

Clustering is a common data mining task. It can be defined as: given a set of data elements, partition it into subsets ("clusters") such that elements within a subset are highly similar to each other, and elements in different subsets are highly dissimilar. This is a very broad definition: first, one needs to specify the notion of similarity; second, even if this is clearly defined, there is the question of exactly how to measure the quality of the clustering. As a result, many different clustering methods have been proposed, each with their own strengths and weaknesses.

Within clustering, we can distinguish the clustering of elements of a set (where each element has its own independent description, and "similarity" refers to the similarity of these descriptions) and clustering of nodes in a graph (where "similarity" refers to their topological closeness or connectedness in the graph). We call the first setting **standard clustering**, the second **graph clustering**. (We use the latter term to be consistent with some of the literature, even if it could be misunderstood as clustering graphs rather than nodes in a graph - we do mean the latter.)

A setting in between these two is where each node in a graph has its own description (here called *annotation*), in addition to being connected to other nodes. We call this setting *annotated graph clustering*. While there has been much research on standard and graph clustering, this mixed setting has only recently started to receive attention, despite its obvious application potential in web mining, systems biology, etc. It is clear that neither standard clustering, nor graph clustering, is optimal in this setting, as each exploits only one part of the available information.

We will call any method that exploits both the information in the annotations and in the graph structure a **hybrid clustering** method. It has been shown before that hybrid methods can yield better clustering results [15,16].

In this paper we build on earlier work by Witsenburg and Blockeel [15], who proposed a hybrid similarity measure and showed that it can be used successfully for agglomerative clustering. We investigate to what extent the same similarity measure can be used in $k$-means-like clustering approaches. It turns out that $k$-means cannot be used *as is* with this measure, because no suitable center measure can be defined. We propose three alternatives: one is the use of $k$-medoids instead of $k$-means, the other two are new variants of $k$-means. An empirical evaluation shows that these variants can yield a better clustering than plain $k$-means with a standard similarity measure.

In the following section, we provide some more background and discuss related work. In Section 3, we discuss the hybrid similarity measure we will use, and the $k$-means algorithm. In Section 4, we discuss the problem with using the hybrid similarity with $k$-means, which boils down to the lack of a good center measure, and we present three ways in which this problem can be solved. In Section 5 we present experiments showing that the proposed algorithms, with the hybrid measure, can indeed yield better clusters. We conclude in Section 6.

## 2 Related Work

Standard clustering methods include bottom-up hierarchical clustering methods, the well-known $k$-means algorithm [9], and many more methods and variants; for an overview, see, e.g., Hartigan [5]. In graph clustering, too, a variety of methods exist (e.g., [1,14,3]); many of these handle weighted edges and construct minimal cuts [2], i.e., the quality criterion for a clustering is that the total weight of connections between clusters should be minimal.

Hybrid methods have not received much attention. One thread of approaches can be found in inductive logic programming, where methods for relational clustering have been proposed [13]. These methods usually consider objects that are assumed to be independent but have an internal structure that is relational. They are typically not set in the context of clustering nodes in a graph. Neville et al. [12] were among the first to explicitly discuss the need for incorporating node content information into graph clustering. More specifically, they consider graph clustering algorithms that can work with weighted edges, and define the weights according to the similarity of the nodes connected by the edge. Thus, they map the hybrid clustering problem to a graph clustering problem, after which any graph clustering method can be used.

More recently, Zhou et al. [16] proposed a method that inserts nodes in the graph for every attribute value in the annotations. Then, edges connect the original nodes with these *attribute nodes* when this attribute value is in the annotation of this original node. This approach is somewhat more flexible with respect to trading off the different goals of standard clustering and graph clustering similarity; for instance, two nodes that originally did not have a path between them could still be in the same cluster, since they can be connected through one or more attribute nodes. This is not the case for most graph clustering methods.

While the above mentioned approaches reduce the hybrid clustering problem to a graph clustering problem, Witsenburg and Blockeel [15] did the opposite: they incorporated graph information into a standard clustering approach. They proposed a similarity measure that combines similarity of annotations with context information from the graph, and showed that bottom-up hierarchical clustering methods can be made to produce better results by using this new similarity measure, instead of a pure annotation-based one. One advantage of translating to standard clustering is that a plethora of clustering methods become available, more than for the translation to graph clustering.

However, not all of those may be as readily available as it may seem. For instance, $k$-means clustering does not only require a similarity measure, but also a center measure (sometimes referred to as centroid or prototype definition) that is compatible with it. For Witsenburg and Blockeel's hybrid similarity measure, it is not obvious what that center measure should be. Hence, the hybrid similarity cannot simply be plugged in into $k$-means. In this paper we discuss and compare three ways around this problem.

## 3 Background Definitions

We here recall Witsenburg and Blockeel's similarity measures [15] and the $k$-means algorithm [9].

**Content-based, Contextual and Combined Similarity** Consider the data set that needs to be clustered to be an annotated graph, then this data set $D$ can be defined as $D = (V, E, \lambda)$ 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, and $\lambda : V \to \mathcal{A}$ a function that assigns to any element $v$ of $V$ an "annotation"; this annotation $\lambda(v)$ is considered to be the *content* of vertex $v$. The graph is undirected and an edge cannot loop back to the same vertex, so with two elements $v, w \in V$ this means that $(v, w) \in E \Leftrightarrow (w, v) \in E$ and $(v, v) \notin E$.

The space of possible annotations is left open; it can be a set of symbols from, or strings over, a finite alphabet; the set of reals; an $n$-dimensional Euclidean space; a powerset of one of the sets just mentioned; etc. The only constraint on $\mathcal{A}$ is that it must be possible to define a similarity measure $\mathcal{S}_{content} : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ as a function that assigns a value to any pair of annotations expressing the similarity between these annotations. Since this similarity is entirely based on the content of the vertices, it will be called the *content-based similarity.* Normally the value of this similarity is in the range $[0, 1]$ where 0 stands for no similarity at all and 1 means that they are considered to be identical.

Now let $\phi : V \times V \to \{0, 1\}$ be a function that assigns a value to a pair of elements in the data set such that $\phi(v, w) = 1$ if $(v, w) \in E$ and 0 otherwise. We define the *neighbor similarity* $\mathcal{S}_{neighbor} : V \times V \to \mathbb{R}$ between two elements $v$ and $w$ from $V$ as the average content-based similarity between $v$ and all neighbors of $w$:

$$\mathcal{S}_{neighbor}(v, w) = \frac{\sum_{u \in V} \mathcal{S}_{content}(\lambda(v), \lambda(u)) \cdot \phi(u, w)}{\sum_{u \in V} \phi(u, w)} \tag{1}$$

This similarity is not symmetric, but we can easily symmetrize it, leading to the *contextual similarity* $\mathcal{S}_{context} : V \times V \to \mathbb{R}$:

$$\mathcal{S}_{context}(v, w) = \frac{\mathcal{S}_{neighbor}(v, w) + \mathcal{S}_{neighbor}(w, v)}{2} \tag{2}$$

The motivation behind defining this similarity measure is that, if similar nodes tend to be linked together, then the neighbors of $w$ in general are similar to $w$. Hence, if similarity is transitive, a high similarity between $v$ and many neighbors of $w$ increases the reasons to believe that $v$ is similar to $w$, even if there is little evidence of such similarity when comparing $v$ and $w$ directly (for instance, due to noise or missing information in the annotation of $w$).

This contextual similarity measure is complementary to the content-based one; it does not use the content-based similarity between $v$ and $w$ at all. Since in practical settings it may be good not to ignore this similarity entirely, Witsenburg and Blockeel also introduced the *combined similarity* $\mathcal{S}_{combined} : V \times V \rightarrow \mathbb{R}$:

$$\mathcal{S}_{combined}(v, w) = c \cdot \mathcal{S}_{content}(v, w) + (1 - c) \cdot \mathcal{S}_{context}(v, w) \qquad (3)$$

with $0 \leq c \leq 1$. $c$ determines the weight of the content-based similarity in the combined similarity. As Witsenburg and Blockeel [15] found no strong effect of using different values for $c$, from now on we consider only the combined similarity with $c = \frac{1}{2}$.

Both the contextual and the combined similarity measures are hybrid similarity measures, since they use both content-based and graph information. The contextual similarity measure between two nodes $v$ and $w$ does take the contents of $v$ and $w$ into account, it just does not use the direct similarity between these contents.

Note that any standard clustering method that can cluster nodes based on (only) their content similarity, can also cluster nodes based on the contextual or combined similarity, and in the latter case it implicitly takes the graph structure into account; the method itself need not know that these data come from a graph.

**The *k*-means algorithm** $k$-means [9] is a clustering algorithm that works as follows. Data set elements are assumed to be vectors in an $n$-dimensional space, and similarity is expressed by Euclidean distance (the smaller the distance, the greater the similarity). The number of clusters $k$ is a parameter of the algorithm. $k$-means proceeds as follows:

1. Choose randomly $k$ different points $M_i$ ($i = 1, \ldots, k$) in the data space; each $M_i$ will serve as a prototype for a cluster $C_i$.
2. Assign each data element to the cluster with the closest prototype.
3. Recalculate each $M_i$ as the mean of all elements of $C_i$.
4. Repeat steps 2 and 3 until the $M_i$ and $C_i$ no longer change.

Here step 2 is known as the *assignment step* and step 3 is known as the *update step.*

$k$-means always converges to a (possibly local) optimum. The proof of this (see, for instance [10]) involves the fact that the sum of all distances from one element to its cluster's prototype can only decrease in each update and assignment step, and only a finite number of such decrements is possible.

## 4 K-Means with the Hybrid Similarity Measure

$K$-means can in principle be used with all sorts of similarity measures; however, it also needs a center measure (e.g., the mean), and to guarantee convergence this center measure must be *compatible* with the similarity measure, that is, reassigning elements to clusters must lead to a monotonic increase (or decrease) of some aggregate function of the similarities beween elements and centers (e.g., increasing sum of similarities). We will call this aggregate function the *overall similarity.*

In our setting, the data elements are annotated vertices in a graph. This raises the question how to calculate the "prototypical vertex" of a subset of vertices from an annotated graph. If annotations are vectors, we can easily define the annotation of prototype $M_i$ as the mean of all annotations $\lambda(v)$ where $v \in C_i$. But our hybrid similarity measures are based on $\mathcal{S}_{neighbor} : V \times V \rightarrow \mathbb{R}$, which also needs the prototype to be connected to nodes in the graph. Since this is not the case, we cannot compute the contextual similarity between the prototype and a data element.

We will discuss three ways around this problem. The first one is to use *$k$-medoids* instead of $k$-means; $k$-medoids always uses existing data elements as centers, which solves the above problem. An alternative is to define the center as an annotation, and define similarity between a node and an annotation, rather than between two nodes. Our third method will be a more efficient approximation of the second. We next discuss these three methods.

### 4.1 $K$-Medoids

$K$-medoids [7] is similar to $k$-means, but differs from it in that the prototype of a cluster (in this case known as the medoid) is always an element from the data set: in the beginning (step 1) it is a random data element; and during an update step (step 3), $M_i$ becomes the element of $C_i$ for which the overall similarity to all others elements of $C_i$ is maximal.

It is easy to see that the hybrid similarity measure can be applied here without problems: since the prototype is always an element in the data set (i.e., a node in the graph), the similarity with other elements can be calculated. To compute the new prototype, one only needs to compute for each element the sum of the similarities to all other elements in that cluster, to determine which is the largest.

$K$-medoids can be a good alternative for $k$-means. It is known to be better than $k$-means when it comes to handling outliers [4], but more limited in its choice of prototypes [8], and less efficient when handling big data sets [4]. The latter is due to the fact that to calculate the new prototype of a cluster in $k$-medoids one needs to calculate the distance from every node in that cluster to every other node in that cluster, while for $k$-means one only needs to calculate the mean of all nodes in it.

### 4.2 $K$-Means-NAM: $K$-Means with neighbor annotation means

The second solution we explore, is to define the center as an annotation instead of a node. Recall that the contextual similarity $S_{context}$ is a symmetrized version of $S_{neighbor}$. The definition of the latter (see (1)) uses for the first element ($v$) only its annotation $\lambda(v)$, not its location in the graph. Thus, the neighbor similarity $\mathcal{S}_{neighbor}$ can be rewritten as a function $\mathcal{S}'_{neighbor} : \mathcal{A} \times V \to \mathbb{R}$ that is defined by:

$$\mathcal{S}'_{neighbor}(M, v) = \frac{\sum_{w \in V} \mathcal{S}_{content}(M, \lambda(w)) \cdot \phi(w, v)}{\sum_{w \in V} \phi(w, v)} \tag{4}$$

We can use this asymmetric neighbor similarity instead of the contextual similarity as described in (2). Also the combined similarity can then be rewritten as a function $\mathcal{S}'_{combined} : \mathcal{A} \times V \to \mathbb{R}$ that is defined by:

$$\mathcal{S}'_{combined}(M, v) = c_1 \cdot \mathcal{S}_{content}(M, v) + c_2 \cdot \mathcal{S}'_{neighbor}(M, v) \tag{5}$$

In this case the new mean of a cluster can be calculated as the average of the annotations of all elements in that cluster, and it is still possible to calculate the similarity between a mean and an element from the data set.

This approach causes a new problem though: the proposed center measure and similarity measure are not compatible, and as a result, $k$-means may no longer converge. In the update step, the new prototype is the mean of the cluster element's annotations, which increases the average *content* similarity between $M$ and the nodes, but not necessarily the *neighbor* similarity between $M$ and these nodes. Consider an element $v$ from the data set whose annotations of the neighbors differ a lot from its own annotation. When using contextual similarity, $v$ will be placed in a cluster with a mean that is close to the annotations of the *neighbors* of $v$, but when updating the new mean for this cluster, this will be done using the annotation of $v$; this will pull the mean towards $v$'s own annotation, and away from the annotation of its neighbors. The effect could be that the new mean will be far from the annotations of $v$'s neighbors, so the monotonic increase of the overall similarity is no longer guaranteed, and the algorithm may not converge.

To ensure convergence, we need to redefine the center measure to be compatible with the similarity measure. As described in Section 3, $\lambda : V \to \mathcal{A}$ assigns an annotation to a vertex. Now let $\lambda' : V \to \mathcal{A}$ be a new function that assigns another annotation to a vertex:

$$\lambda'(v) = \frac{\sum_{w \in V} \lambda(w) \cdot \phi(v, w)}{\sum_{w \in V} \phi(v, w)} \tag{6}$$

$\lambda'(v)$ is the mean annotation of all neighbors of $v$. It is easily seen that calculating the center as the average of these new annotations is compatible with the proposed similarity measure.

Following the same reasoning, when using the combined similarity instead of the contextual one, the annotation function $\lambda'' : V \to \mathcal{A}$ should be used:

$$\lambda''(v) = \frac{\lambda(v) + \lambda'(v)}{2} \tag{7}$$

This setup, which we call *$k$-means-NAM* ($k$-means with neighbor annotation means) is the second solution proposed to enable the use of a $k$-means-like algorithm with a hybrid similarity measure.

### 4.3 K-Means-NAMA: *k*-means-NAM efficiently approximated

The solution proposed in Section 4.2 is less efficient than the original $k$-means algorithm. To calculate the similarity between a mean and an element $v$, $k$-means only needs to calculate one content similarity. $k$-means-NAM, on the other hand, needs to calculate the content similarity between the prototype and all neighbors of $v$ (in order to compute their average), which makes it a number of times slower.

A more efficient alternative to this is to average out the neighbor annotations themselves, instead of averaging out the similarities. That is, with $v_1, \ldots, v_n$ the neighbors of $v$, instead of computing $\sum_i \mathcal{S}_{content}(M, \lambda(v_i))/n$, we could compute $\mathcal{S}_{content}(M, \sum_i \lambda(v_i)/n)) = \mathcal{S}_{content}(M, \lambda'(v))$. These two are mathematically different, and generally do not give the same outcome, but they approximate each other well when the $v_i$ are "in the same direction" from $a$. The advantage of this additional approximation is that for each $v$, $\lambda'(v)$ can be computed once in advance, and substituted for $\lambda(v)$, after which the standard $k$-means algorithm can be run. In the same way that using $\lambda'$ instead of $\lambda$ allows us to approximate $k$-means-NAM with contextual distance, using $\lambda''$ approximates $k$-means-NAM with the combined distance. We call this approximation $k$-means-NAMA.

## 5 Experimental Results

To evaluate the usefulness of the proposed methods, a few questions need to be answered experimentally. $K$-medoids can be used both with contents-based or hybrid similarities; does the use of a hybrid similarity yield better results? For $k$-means, we have to choose between standard $k$-means with the content-based similarity, or an approximative variant that takes contextual information into account; does the use of contextual information compensate for a possible quality loss due to having to use an approximate method? Finally, how do the three contextual methods compare?

| DATA SET | CLASSES | PAPERS | EDGES | DENSITY |
|---|---|---|---|---|
| $D_1$ | 8 | 752 | 2526 | 3.36 |
| $D_2$ | 17 | 1779 | 6658 | 3.74 |
| $D_3$ | 24 | 2585 | 10754 | 4.16 |
| $D_4$ | 31 | 4779 | 19866 | 4.17 |
| $D_5$ | 45 | 8025 | 41376 | 5.16 |

**Table 1.** Characteristics of the five subsets created from the Cora data set.

## 5.1 Experimental Setup

We evaluate the methods on the Cora dataset [11], an often-used benchmark. This set contains scientific papers divided into 70 categories. A paper can have multiple classes. For 37,000 of these papers, abstracts are available for keyword extraction, and the citations between papers are also available. In our context, papers are nodes in a graph, the abstracts are their annotations, and the citations between papers form the edges. We cluster the papers based on their annotations and citations. The quality measure for the clustering will relate to how well papers in the same cluster belong to the same classes (note that the classes themselves are not part of the annotation, only the abstracts are).

From Cora, five different subsets have been created. The first subset contains papers from 8 different classes. For every next subset, papers from several additional classes have been added. Only papers that have an abstract and are connected to (i.e., cite or are cited by) at least one other paper in the subset are regarded. Table 1 shows some of the characteristics of these subsets.

For every data set $D_1$ through $D_5$, $V$ is defined by all papers in that data set. For all $v, w \in V$, $(v, w) \in E$ when $v$ cites $w$ or vice versa. Let $W$ be the set of all keywords that can be found in the abstracts of all elements of all data sets $D_i$, and $m$ the total number of keywords; $\mathcal{A} \subseteq \mathbb{R}^m$ and $\lambda(v)$ is the set of keywords that are in the abstract of $v$. For the content-based similarity between two elements $v, w \in V$ we use the Jaccard index [6]:

$$\mathcal{S}_{content} = \frac{|\lambda(v) \cap \lambda(w)|}{|\lambda(v) \cup \lambda(w)|}. \tag{8}$$

The three solutions as proposed in Section 4, have been used to cluster the elements in the 5 data sets as described in this section. For every combination this has been done for all three similarities (content-based, contextual and combined). The value for $k$ was varied from 100 to 2. Keep in mind that $k$-means-NAM(A), used with the content-based similarity, is actually the regular $k$-means algorithm.

The found clusterings were evaluated by looking at every pair of elements in a cluster and calculating the Jaccard index of their categories. In the Cora data set, papers can have multiple categories, hence, the Jaccard index is used to resolve partial matching. The average of all these Jaccard indices is the quality of the clustering.

The experiments have been done for $k = 100, 95, 90, \ldots, 20, 18, 16, \ldots, 2$. A set of experiments where a clustering is found once for every $k$ is called a *series*. As $k$-means and $k$-medoids depend on the random initial choice of prototypes, every "series" has been done 100 times and 100 experiments with the same $k$ on the same data set is called a *run*. Of these runs the average results are reported.

## 5.2 *K*-Means-NAM vs *K*-Means-NAMA

In Section 4.3 we hypothesized that $k$-means-NAMA would get similar results as $k$-means-NAM, but faster. This can be tested by regarding the percentual difference in

|  | CONTEXTUAL SIMILARITY |  | COMBINED SIMILARITY |  |
|---|---|---|---|---|
| DATA SET | ABSOLUTE DIFFERENCE | AVERAGE DIFFERENCE | ABSOLUTE DIFFERENCE | AVERAGE DIFFERENCE |
| $D_1$ | 2.4% | -1.2% | 2.7% | -1.3% |
| $D_2$ | 2.4% | +1.0% | 1.9% | -0.2% |
| $D_3$ | 1.6% | -0.7% | 1.8% | +0.7% |
| $D_4$ | 0.9% | +0.2% | 1.3% | -0.8% |
| $D_5$ | 1.3% | +0.4% | 2.0% | +1.2% |

**Table 2.** Percentual difference in quality of the found clusters by $k$-means-NAMA compared to the found clusters by $k$-means-NAMA.

score between the results of the runs with $k$-means-NAMA and the same runs with $k$-means-NAM. A positive percentage means that $k$-means-NAMA outperformed $k$-means-NAM, and a negative percentage the opposite. Also the absolute value of these percentages are taken into concern. Table 2 shows these results. First, the averages of all runs in a data set for the absolute value of these percentages are small, indicating there is not a lot of difference in performance. Second, when the sign of the percentual differences are also taken into account, the differences are even smaller, indicating that one is not overall better than the other. These conclusions hold for both the contextual and the combined similarity.

Table 3 shows the average computation time each method needed to finish one series of clusterings. With the content-based similarity, there is not much difference. This is as expected, since here, $k$-means-NAM boils down to regular $k$-means. For the contextual and combined similarities, however, there is a big difference: the time that it takes $k$-means-NAMA to complete a series remains about the same, while $k$-means-NAM needs about 4 times as long for the contextual similarity and about 5 times as long for the combined similarity.

Since there is no real difference in quality between $k$-means-NAM and $k$-means-NAMA, from now on we only consider the results for $k$-means-NAMA.

|  | CONTENT-BASED |  | CONTEXTUAL |  | COMBINED |  |
|---|---|---|---|---|---|---|
| DATA SET | $k$-MEANS-NAM | $k$-MEANS-NAMA | $k$-MEANS-NAM | $k$-MEANS-NAMA | $k$-MEANS-NAM | $k$-MEANS-NAMA |
| $D_1$ | $7.2 \cdot 10^1$ | $8.6 \cdot 10^1$ | $3.0 \cdot 10^2$ | $1.1 \cdot 10^2$ | $4.0 \cdot 10^2$ | $1.1 \cdot 10^2$ |
| $D_2$ | $4.8 \cdot 10^2$ | $5.9 \cdot 10^2$ | $1.5 \cdot 10^3$ | $6.0 \cdot 10^2$ | $2.1 \cdot 10^3$ | $5.8 \cdot 10^2$ |
| $D_3$ | $9.7 \cdot 10^2$ | $1.2 \cdot 10^3$ | $4.0 \cdot 10^3$ | $1.3 \cdot 10^3$ | $4.9 \cdot 10^3$ | $1.3 \cdot 10^3$ |
| $D_4$ | $3.5 \cdot 10^3$ | $4.2 \cdot 10^3$ | $1.3 \cdot 10^4$ | $4.7 \cdot 10^3$ | $1.7 \cdot 10^4$ | $4.4 \cdot 10^3$ |
| $D_5$ | $1.1 \cdot 10^4$ | $1.3 \cdot 10^4$ | $4.5 \cdot 10^4$ | $1.6 \cdot 10^4$ | $5.4 \cdot 10^4$ | $1.4 \cdot 10^4$ |

**Table 3.** CPU time, in seconds, for $k$-means-NAM and $k$-means-NAMA to do one series as defined in Section 5.1 (on a Intel® Quad Core™2, 2.4GHz with 4 Gb memory), for the content-based, contextual, and combined similarities.

## 5.3 Quality Improvement due to the Hybrid Similarity Measures

Figure 1 shows the results for clustering the subsets $D_1$ and $D_4$. The other subsets show similar characteristics. Table 4 shows the average results for all data sets for $k$-medoids and $k$-means-NAMA. On Cora, using the hybrid similarity measure indeed improves the quality of the found clustering, as compared to using the content-

| *k*-MEDOIDS | | | | | |
|---|---|---|---|---|---|
| | CONTENT-BASED | CONTEXTUAL | | COMBINED | |
| DATA SET | QUALITY | QUALITY | IMPROVEMENT | QUALITY | IMPROVEMENT |
| $D_1$ | 0.44 | 0.57 | +29% | 0.58 | +32% |
| $D_2$ | 0.25 | 0.36 | +41% | 0.39 | +55% |
| $D_3$ | 0.19 | 0.29 | +50% | 0.33 | +71% |
| $D_4$ | 0.19 | 0.25 | +33% | 0.30 | +59% |
| $D_5$ | 0.14 | 0.19 | +35% | 0.24 | +72% |
| *k*-MEANS-NAMA | | | | | |
| | CONTENT-BASED | CONTEXTUAL | | COMBINED | |
| DATA SET | QUALITY | QUALITY | IMPROVEMENT | QUALITY | IMPROVEMENT |
| $D_1$ | 0.32 | 0.52 | +60% | 0.49 | +51% |
| $D_2$ | 0.23 | 0.42 | +79% | 0.40 | +71% |
| $D_3$ | 0.21 | 0.37 | +77% | 0.35 | +70% |
| $D_4$ | 0.22 | 0.37 | +68% | 0.34 | +55% |
| $D_5$ | 0.19 | 0.36 | +74% | 0.31 | +62% |

**Table 4.** Average quality found and percentual improvement with regards to the content-based similarity, for *k*-medoids (upper half) and *k*-means-NAMA (lower half).

based similarity, for both *k*-medoids and *k*-means. There is no conclusive evidence, however, which one is best among *k*-medoids and (approximative) *k*-means.

## 6 Conclusion

We have discussed how a hybrid similarity for nodes in a graph (taking into account both contents and context) can be used with *k*-means-like clustering methods. *K*-means cannot be employed in a straightforward way because the concept of a "mean node" cannot be defined; however, it can be approximated by *k*-medoids and by two newly proposed methods. These two methods boil down to using approximate similarity or center measures so that *k*-means becomes applicable. The main conclusions from this work are that: (1) *k*-means clustering can indeed work with hybrid similarities, if adapted appropriately; (2) the use of a hybrid similarity with (adapted) *k*-means or *k*-medoids does yield better clusters, compared to content-based similarities; (3) the adapted *k*-means approaches sometimes work better than *k*-medoids, so they are a valuable alternative to it but do not make it redundant.

**Acknowledgements** The authors thank Walter Kosters for multiple suggestions w.r.t. formulation and writing style. This research is funded by the Dutch Science Foundation (NWO) through a VIDI grant.

## References

1. D.J. Cook and L.B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.
2. G.W. Flake, R.E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet mathematics*, 1:385–408, 2004.
3. M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
4. J. Han, M. Kamber, and A. Tung. Spatial clustering methods in data mining: A survey. In H. Miller and J. Han, editors, *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis, 2001.

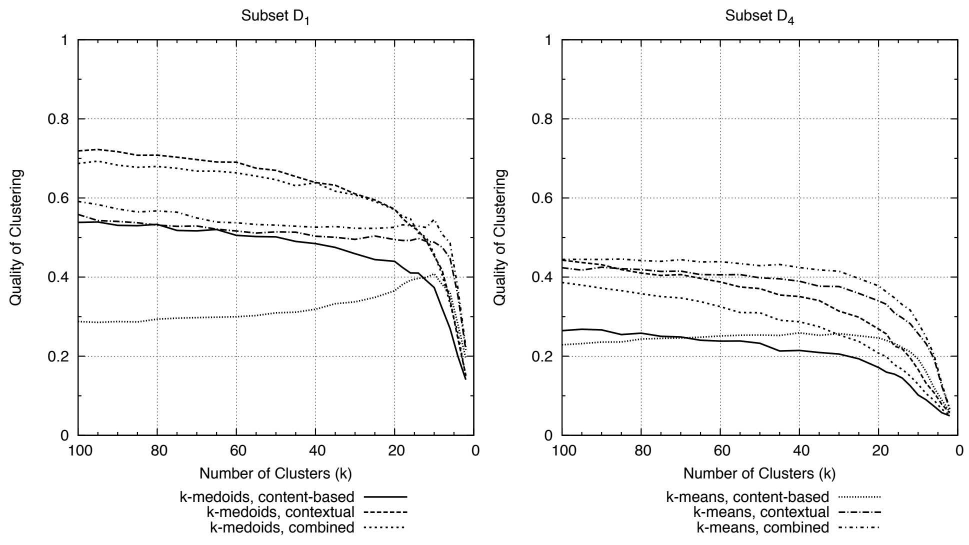

**Fig. 1.** Clustering quality for $D_1$ (left) and $D_4$ (right) for $k$-medoids and $k$-means-NAMA.

5. J. Hartigan. *Clustering algorithms*. Wiley, 1975.
6. P. Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Soci'etè Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
7. L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis Based on the $L_1$ Norm and Related Methods*, pages 405–416. Elsevier Science, 1987.
8. M. Kirsten and S. Wrobel. Extending k-means clustering to first-order representations. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866, pages 112–129. Springer Berlin / Heidelberg, 2000.
9. J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.
10. C. D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press., 2008.
11. A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.
12. J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence*, 2003.
13. J. Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, K.U.Leuven, Dept. of Computer Science, 2002.
14. P.H.A. Sneath and R.R. Sokal. *Numerical Taxonomy - The Principles and Practice of Numerical Classification*. W.H. Freeman, San Francisco, CA, 1973.
15. T. Witsenburg and H. Blockeel. A method to extend existing document clustering procedures in order to include relational information. In S. Kaski, S. Vishwanathan, and S. Wrobel, editors, *Proceedings of the sixth International Workshop on Mining and Learning with Graphs*, Helsinki, Finland, 2008.
16. Y. Zhou, H. Cheng, and J.X. Yu. Graph clustering based on structural/attribute similarities. In *Proceedings of the VLDB Endowment*, pages 718–729, Lyon, France, 2009. VLDB Endowment.