# Experiments in Document Clustering using Cluster Specific Term Weights

Christian Borgelt and Andreas Nürnberger

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
{borgelt,nuernb}@iws.cs.uni-magdeburg.de

**Abstract.** We study methods to initialize or bias different clustering methods using prior information about the "importance" of a keyword w.r.t. to the specific clusters. These studies give us hints on how to initialize clustering methods in order to improve the clustering performance if prior knowledge is available. This can be especially useful if a user-specific clustering of a document collection or web search result set is desired.

## 1 Introduction

The problem of finding descriptive weights for terms in document collections in order to improve retrieval performance has been studied extensively in the past (see, for instance, [12, 24, 23]). To achieve an improved classification or clustering performance for a given text collection, it is usually necessary to select a subset of all describing features (i.e. keywords) and/or to re-weight the features w.r.t. a specific classification or clustering goal. Consequently, several studies were conducted in this direction. For example, it was explored how to select keywords based on statistical and information theoretical measures [9, 21, 28] or how to combine clustering and keyword weighting techniques [10] in order to improve the clustering performance.

In prior work we studied different hard and fuzzy clustering methods with and without variances [5]. These experiments indicated that the use of variances—which can be considered as a method for cluster specific keyword weighting—can improve the clustering performance. Nevertheless, it is still unclear to what extent term re-weighting influences the clustering performance and whether initial—global or cluster specific—term re-weighting can be used to bias or improve the performance. Therefore, in the following, we compare clustering with and without term re-weighting techniques using different hard and fuzzy clustering methods.

This paper is organized as follows: In Section 2 we briefly review some basics of fuzzy clustering and a fuzzified version of learning vector quantization. In Section 3 we review pre-processing methods for documents and in particular the vector space model, which we use to represent documents. In Section 4 we

present our experimental results of clustering web page collections using different global and cluster-specific term weighting approaches and finally, in Section 5, we draw conclusions from our discussion.

## 2 Clustering

The best-known classical prototype based hard clustering methods are *c-means clustering* [7, 4] and *learning vector quantization* [17, 18]. In the following, we briefly describe their generalizations to fuzzy clustering and fuzzified learning vector quantization as we use it in our experiments. For a more detailed discussion and evaluation of these methods for document clustering see [5].

### 2.1 Fuzzy Clustering

While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for *degrees of membership*, to which a datum belongs to different clusters [1, 2, 14]. Most fuzzy clustering algorithms are objective function based: they determine an optimal (fuzzy) partition of a given data set $\mathbf{X} = \{\boldsymbol{x}_j \mid j = 1, \ldots, n\}$ into $c$ clusters by minimizing an objective function

$$J(\mathbf{X}, \mathbf{U}, \mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}^{2}$$

subject to the constraints

$$\forall i; 1 \leq i \leq c: \quad \sum_{j=1}^{n} u_{ij} > 0, \qquad \text{and} \qquad \forall j; 1 \leq j \leq n: \quad \sum_{i=1}^{c} u_{ij} = 1,$$

where $u_{ij} \in [0, 1]$ is the membership degree of datum $\boldsymbol{x}_j$ to cluster $i$ and $d_{ij}$ is the distance between datum $\boldsymbol{x}_j$ and cluster $i$. The $c \times n$ matrix $\mathbf{U} = (u_{ij})$ is called the *fuzzy partition matrix* and $\mathbf{C}$ describes the set of clusters by stating location parameters (i.e. the cluster center) and maybe size and shape parameters for each cluster. The parameter $w$, $w > 1$, is called the *fuzzifier* or *weighting exponent*. It determines the "fuzziness" of the classification: with higher values for $w$ the boundaries between the clusters become softer, with lower values they get harder. Usually $w = 2$ is chosen. Hard clustering results in the limit for $w \to 1$. However, a hard assignment may also be determined from a fuzzy result by assigning each data point to the cluster to which it has the highest degree of membership.

Since the objective function $J$ cannot be minimized directly, an iterative algorithm is used, which alternately optimizes the membership degrees and the cluster parameters [1, 2, 14]. That is, first the membership degrees are optimized for fixed cluster parameters, then the cluster parameters are optimized for fixed membership degrees. The main advantage of this scheme is that in each of the two steps the optimum can be computed directly. By iterating the two steps

the joint optimum is approached (although, of course, it cannot be guaranteed that the global optimum will be reached—the algorithm may get stuck in a local minimum of the objective function $J$). The update formulae are derived by simply setting the derivative of the objective function $J$ w.r.t. the parameters to optimize equal to zero (necessary condition for a minimum).

Depending on the distance measure used, several different fuzzy clustering algorithms can be distinguished. Classical fuzzy $c$-means clustering employs the *Euclidean distance*, while Gustafson-Kessel algorithm [13] uses the *Mahalanobis distance* and the fuzzy maximum likelihood estimation (FMLE) algorithm [11] is based on the assumption that the data was sampled from a mixture of $c$ multivariate normal distributions as in the statistical approach of mixture models [8, 3]. It is worth noting that of both the Gustafson-Kessel as well as the FMLE algorithm there exist so-called *axes-parallel* versions, which restrict the covariance matrices to diagonal matrices and thus allow only axes-parallel ellipsoids [15]. These variants have certain advantages w.r.t. robustness and execution time.

### 2.2 Learning Vector Quantization

Learning vector quantization [17, 18], in its classical form, is a competitive learning algorithm that has been developed in the area of artificial neural networks and that can be applied to classified as well as unclassified data. Here we confine ourselves to unclassified data, where the algorithm consists in iteratively updating a set of $c$ so-called *reference vectors* $\boldsymbol{\mu}_i$, $i = 1, \ldots, c$, each of which is represented by a neuron. For each data point $\boldsymbol{x}_j$, $j = 1, \ldots, n$, the closest reference vector (the so-called "winner neuron") is determined and then this reference vector (and only this vector) is updated according to

$$\boldsymbol{\mu}_i^{(\text{new})} = \boldsymbol{\mu}_i^{(\text{old})} + \eta_1 \left( \boldsymbol{x}_j - \boldsymbol{\mu}_i^{(\text{old})} \right), \tag{1}$$

where $\eta_1$ is a learning rate. This learning rate usually decreases with time in order to avoid oscillations and to enforce the convergence of the algorithm.

Membership degrees can be introduced into this basic algorithm in two different ways. In the first place, one may employ an activation function for the neurons, for which a radial function like the

$$\text{Cauchy function} \quad f(r) = \frac{1}{1+r^2} \qquad \text{or the} \quad \text{Gaussian function} \quad f(r) = e^{-\frac{1}{2}r^2}$$

may be chosen, where $r$ is the (radial) distance from the reference vector. In this case all reference vectors are updated for each data point, with the update being weighted with the value of the activation function. However, this scheme, which is closely related to *possibilistic fuzzy clustering* [19], usually leads to unsatisfactory results, since there is no dependence between the clusters, so that they tend to end up at the center of gravity of all data points. This corresponds to the fact that in possibilistic fuzzy clustering the objective function is truly minimized only if all cluster centers are identical [27]. Useful results are obtained only if the method gets stuck in a local minimum, which is an undesirable situation.

An alternative is to rely on a normalization scheme as in *probabilistic fuzzy clustering*, that is, to compute the weight for the update of a reference vector as the relative inverse (squared) distance from this vector, or as the *relative* activation of a neuron. This is the approach we employ here.

Furthermore we associate with each neuron not only a reference vector $\boldsymbol{\mu}_i$, but also a covariance matrix $\boldsymbol{\Sigma}_i$, which describes the shape and (if we do not require it to be normalized to determinant 1) the size of the represented cluster. A derivation of the update rule for this covariance matrix can be found in [5]. It should be noted that versions of this algorithm that require the covariance matrix to be normalized to determinant 1 or restrict the covariance matrix to a diagonal matrix may be considered, too. Such constraints can improve the robustness or the execution time of the algorithm. Finally it should be noted that the updates may be executed in batch mode, aggregating the changes resulting from the data points and actually updating the reference vectors and covariance matrices only at the end of an epoch.

## 3 Clustering Document Collections

To be able to cluster text document collections with the methods discussed above, we have to map the text files to numerical feature vectors. Therefore, we first applied standard preprocessing methods, i.e., stopword filtering and stemming (using the Porter Stemmer [22]), encoded each document using the vector space model [23] and finally selected a subset of terms as features for the clustering process as briefly described in the following.

### 3.1 The Vector Space Model

The vector space model represents text documents as vectors in an $m$-dimensional space, i.e., each document $j$ is described by a numerical feature vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$. Each element of the vector represents a word of the document collection, i.e., the size of the vector is defined by the number of words of the complete document collection.

For a given document $j$ the so-called weight $x_{jk}$ defines the importance of the word $k$ in this document with respect to the given document collection $C$. Large weights are assigned to terms that are frequent in relevant documents but rare in the whole document collection [24]. Thus a weight $x_{jk}$ for a term $k$ in document $j$ is computed as the term frequency $\mathrm{tf}_{jk}$ times the inverse document frequency $\mathrm{idf}_k$, which describes the term specificity within the document collection.

In [25] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency (defined as $\mathrm{idf}_k = \log(n/n_k)$), a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$x_{jk} = \frac{\mathrm{tf}_{jk} \log \frac{n}{n_k}}{\sqrt{\sum_{l=1}^{m} \left( \mathrm{tf}_{jl} \log \frac{n}{n_l} \right)^2}}, \tag{2}$$

where $n$ is the size of the document collection $C$, $n_k$ the number of documents in $C$ that contain term $k$, and $m$ the number of terms that are considered.

Based on a weighting scheme a document $j$ is described by an $m$-dimensional vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$ of term weights and the similarity $S$ of two documents (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors (by which—if we assume normalized vectors—the cosine between the two document vectors is computed), i.e.

$$S(\boldsymbol{x}_j, \boldsymbol{x}_k) = \sum_{l=1}^{m} x_{jl} \cdot x_{kl}. \tag{3}$$

For a more detailed discussion of the vector space model and weighting schemes see, for instance, [12, 24, 23].

Note that for normalized vectors the scalar product is not much different in behavior from the Euclidean distance, since for two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ it is

$$\cos\varphi = \frac{\boldsymbol{xy}}{|\boldsymbol{x}| \cdot |\boldsymbol{y}|} = 1 - \frac{1}{2}\, d^2\!\left(\frac{\boldsymbol{x}}{|\boldsymbol{x}|}, \frac{\boldsymbol{y}}{|\boldsymbol{y}|}\right).$$

Although the scalar product is faster to compute, it enforces spherical clusters. Therefore we rely on the Mahalanobis distance in our approach.

### 3.2 Index Term Selection

To reduce the number of words in the vector description we applied a simple method for keyword selection by extracting keywords based on their entropy. In the approach discussed in [16], for each word $k$ in the vocabulary the entropy as defined by [20] was computed:

$$W_k = 1 + \frac{1}{\log_2 n} \sum_{j=1}^{n} p_{jk} \log_2 p_{jk} \qquad \text{with} \quad p_{jk} = \frac{\mathrm{tf}_{jk}}{\sum_{l=1}^{n} \mathrm{tf}_{lk}}, \tag{4}$$

where $\mathrm{tf}_{jk}$ is the frequency of word $k$ in document $j$, and $n$ is the number of documents in the collection. Here the entropy gives a measure how well a word is suited to separate documents by keyword search. For instance, words that occur in many documents will have low entropy. The entropy can be seen as a measure of the importance of a word in the given domain context. As index words a number of words that have a high entropy relative to their overall frequency have been chosen, i.e. of words occurring equally often those with the higher entropy can be preferred. Empirically this procedure has been found to yield a set of relevant words that are suited to serve as index terms [16].

However, in order to obtain a fixed number of index terms that appropriately cover the documents, we applied a greedy strategy: from the first document in the collection select the term with the highest relative entropy as an index term. Then mark this document and all other documents containing this term. From the first of the remaining unmarked documents select again the term with the

| Label | Dataset Category | Associated Theme |
|---|---|---|
| A | Commercial Banks | Banking & Finance |
| B | Building Societies | Banking & Finance |
| C | Insurance Agencies | Banking & Finance |
| D | Java | Programming Lang. |
| E | C / C++ | Programming Lang. |
| F | Visual Basic | Programming Lang. |
| G | Astronomy | Science |
| H | Biology | Science |
| I | Soccer | Sport |
| J | Motor Racing | Sport |
| K | Sport | Sport |

**Table 1.** Categories and Themes of the used benchmark data set of web pages.

highest relative entropy as an index term. Then mark again this document and all other documents containing this term. Repeat this process until all documents are marked, then unmark them all and start again. The process can be terminated when the desired number of index terms have been selected.

## 4 Experiments

For our experimental studies we chose the collection of web page documents used in [26].[1] The data set consists of 11,000 web pages classified into 11 equally-sized categories each containing 1,000 web documents. To each category one of four distinct themes, namely Banking and Finance, Programming Languages, Science, and Sport was assigned as shown in Table 1.

In the following we present results we obtained using the preprocessing strategies described above. After stemming and stop word filtering we obtained 163,860 words. This set was further reduced by removing terms that are shorter than 4 characters and that occur less then 15 times or more than $11,000/12 \approx 917$ times in the whole collection. In this way we made sure that no words that perfectly separate one class from another are used in the describing vectors. From the remaining 10626 words we selected 400 words by applying the greedy index-term selection approach described in Section 3.2. For our clustering experiments we selected finally subsets of the 50, 100, 150, ..., 350, 400 most frequent words in the subset to be clustered. Based on these words we determined vector space descriptions for each document (see Section 3.1, Equation (2)) that we used in our clustering experiments. All vectors were normalized to unit length (*after* the subset selection).

To assess the clustering performance using term re-weighting techniques, we computed the performance on the same data sets used in our previous experi-

[1] This collection is available for download at
http://www.pedal.rdg.ac.uk/banksearchdataset

ments [5], i.e., we clustered the union of the dissimilar data sets A and I, and the semantically more similar data sets B and C. In a third experiment we used all classes and tried to find clusters describing the four main themes, i.e., banking, programming languages, science, and sport.

For our experiments we used $c$-means, fuzzy clustering and learning vector quantization methods. The learning vector quantization algorithm updated the cluster parameters once for every 100 documents. [2]

A detailed discussion of the performance of these methods with and without cluster centers normalized to unit length, with and without variances (i.e., spherical clusters and axes-parallel ellipsoids—diagonal covariance matrices—of equal size), and with the inverse squared distance or the Gaussian function for the activation / membership degrees can be found in [5]. Here, however, we focus on term re-weighting aspects.

### 4.1 Clustering using Variances

Our prior experiments in document clustering [5] indicated that the use of variances—which can be seen as a method for cluster specific keyword weighting —can sometimes improve the clustering performance and stability. However, in our first studies we restricted ourselves to analyze the performance using mean performances and variances. As a consequence, the causes for the differences in the performance remained somewhat unclear. Therefore we repeated several of the experiments and present in Figures 2 to 3 the results obtained with cluster centers normalized to length 1 with and without variances for hard $c$-means, fuzzy $c$-means and (fuzzified) learning vector quantization. All results represent the values of ten runs, which differed in the initial cluster positions and the order in which documents were processed. For the experiments with variances we restricted the maximum ratio of the variances to $1.2^2 : 1 = 1.44 : 1$, which seemed to yield the best results over all three clustering experiments.

The dotted lines show the default accuracy (obtained if all documents are assigned to the majority class). The grey horizontal lines in each block, which are also marked by diamonds to make them more easily visible, show the average classification accuracy (computed from a confusion matrix by permuting the columns so that the minimum number of errors results) in percent (left axis), while the black crosses indicate the performance single experiments. The grey dots and lines close to the bottom show the average execution times in seconds (right axis), while the smaller grey dots and lines at the top of each diagram show the performance of a Naïve Bayes Classifier trained with the corresponding subset of words. The Naïve Bayes Classifier can be considered as an upper limit, while the default accuracy is a lower baseline.

For all data sets the clustering process for fuzzy $c$-means and (fuzzified) learning vector quantization is much more stable than $c$-means. However, all

[2] All experiments were carried out with a program written in C and compiled with gcc 3.3.3 on a Pentium 4C 2.6GHz system with 1GB of main memory running S.u.S.E. Linux 9.1. The program and its sources can be downloaded free of charge at http://fuzzy.cs.uni-magdeburg.de/~borgelt/cluster.html.

methods seem to switch between two strong local minima for the semantically similar data sets B and C.

The introduction of variances increases the performance of fuzzy $c$-means in all cases. However, the performance for $c$-means is only improved for the two class problem with data sets A and I and the four class problem, while the performance of (fuzzified) learning vector quantization is improved for the semantically more similar data sets B and C and the four class problem.

### 4.2 Keyword Weighting by Information Gain

*Information gain* (also known as *mutual (Shannon) information* or *(Shannon) cross entropy*), which is frequently used in decision tree learning, measures the average or expected entropy reduction resulting from finding out the value of a specific attribute. In text categorization information gain can be used to measure how well a term can be used to categorize a document, i.e., it measures the entropy reduction based on this specific term.

The information gain of a term $t_k$ for a given set of $r$ classes $c_i$ is defined as:

$$\begin{aligned} I_{\text{gain}}(t_k) = & -\sum_{i=1}^{r} P(c_i) \log_2 P(c_i) \\ & + P(t_k) \sum_{i=1}^{r} P(c_i|t_k) \log_2 P(c_i|t_k) \\ & + P(\bar{t}_k) \sum_{i=1}^{r} P(c_i|\bar{t}_k) \log_2 P(c_i|\bar{t}_k) \end{aligned} \tag{5}$$

The information gain values are then either used to re-weight the terms of each document or to initialize the cluster-specific variances (see below).

### 4.3 Re-Scaling the Document Space

In order to study the effects of keyword weighting, we computed the "importance" of each keyword for the classification of a document based on the information gain (see above). These "importance" values are then used to re-weight the terms in each document by computing

$$x^*_{jk} = x_{jk} \cdot (I_{\text{gain}}(t_k) + o) \tag{6}$$

and then re-normalizing the document vectors to unit length, resulting in a re-scaling of the document space with respect to the importance of a keyword.

The offset $o$ in the above formula was computed as

$$o = \frac{\max_{t_k \in T} I_{\text{gain}}(t_k) - r \cdot \min_{t_k \in T} I_{\text{gain}}(t_k)}{r - 1},$$

where $r$ is a user-specified maximum ratio of the scaling factors for different terms and $T$ is the current set of index terms. From several experiments we

conducted it seems that values of $r$ must be small (close to 1) in order not to spoil the performance completely. Here we chose $r = 1.5$.

The results of these experiments are shown in the top rows of Figures 4 to 6. As can be seen, no gains result in any of the cases. The accuracy rather deteriorates slightly, an effect that gets stronger with higher values of $r$ as we observed in other experiments. Hence we can conclude that re-scaling the document space in the way described does not lead to an improved performance.

### 4.4 Cluster Specific Keyword Weights

Instead of using the information gain to re-scale the document space one may also add shape parameters (i.e., (co)variances) to the cluster prototypes, which are initialized according to the "importance" of a term. This has the advantage that term weights can be cluster specific, since each cluster may use a different set of variances.

To evaluate this approach, we proceeded as follows: in a first step we clustered the documents with randomly chosen starting points and without variances. Afterwards, the best matching classes are automatically assigned by evaluating the confusion matrix of the classification result obtained with the learned clusters and the correct document classes.

Then the cluster prototypes were enhanced with cluster-specific variances computed as the product of the term frequency in the class and the information gain of the term w.r.t. a separation of the class assigned to the cluster from all other classes. In order to keep the cluster shapes close to spherical, we restricted the maximum ratio of the variances to $1.2^2 : 1 = 1.44 : 1$ (cf. Section 4.1. Other values for this maximum ratio (higher as well as lower) led to worse results. Especially larger values considerably worsened the performance.

Finally, in a second clustering run, these enhanced cluster prototypes were optimized without changing the variances (only the cluster centers were adapted).

The results of these experiments are shown in the bottom rows of Figures 4 to 6. As can be seen, the cluster-specific variances stabilize the results for the four cluster problem and—though only very slightly—improve the performance for the two cluster problems. Thus we can conclude that cluster-specific variance may provide some means for term weighting. However, the approach seems to be very sensitive to the parameter settings. Furthermore, the computational costs are very high.

### 4.5 Choosing Initial Cluster Centers

As we mentioned in Section 4.1 all clustering methods seem to switch between local minima depending on the initial cluster centers choosen—which is in fact a well known clustering problem, especially for the less robust $c$-means algorithm, which is prone to get stuck in local optima easily. Therefore we studied a quite simple initialization approach: for each class we sorted the index terms w.r.t. the product of the term frequency in the class and the information gain of the term

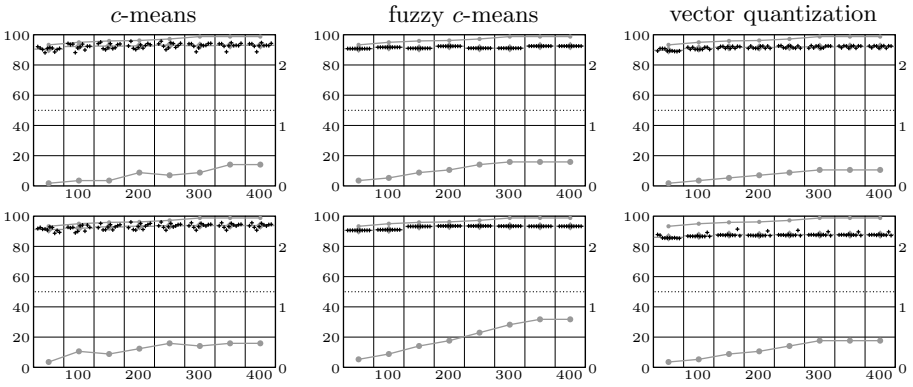


**Fig. 1.** Classification accuracy over number of keywords on commercial banks versus soccer (top row: standard, bottom row: with adaptable variances).

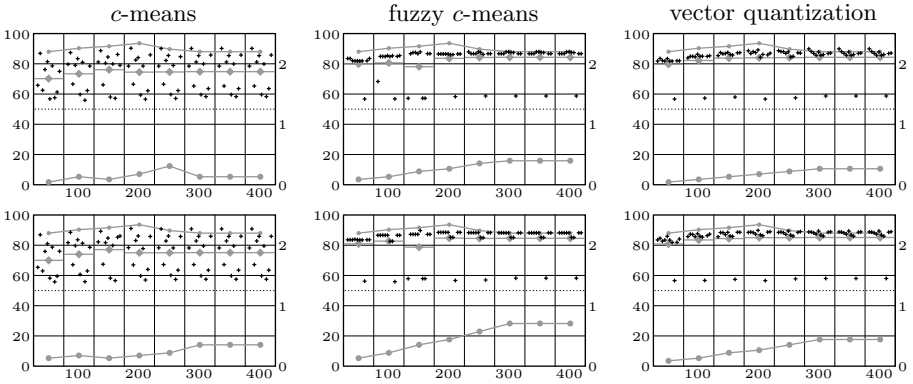


**Fig. 2.** Classification accuracy on building companies versus insurance agencies (top row: standard, bottom row: with adaptable variances).

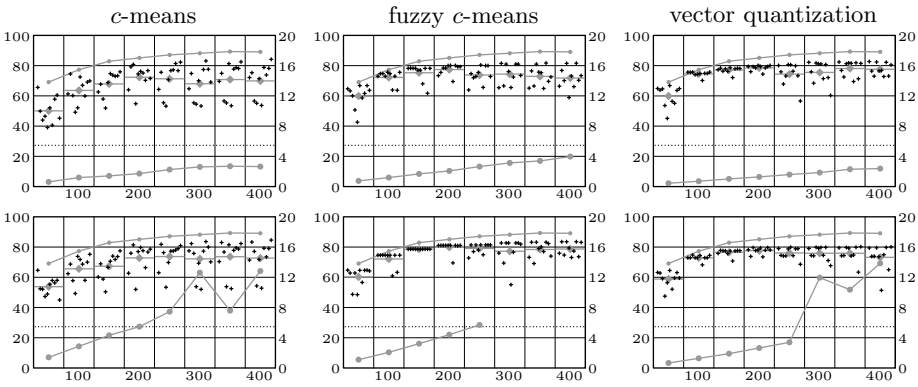


**Fig. 3.** Classification accuracy on major themes (four clusters; top row: standard, bottom row: with adaptable variances).

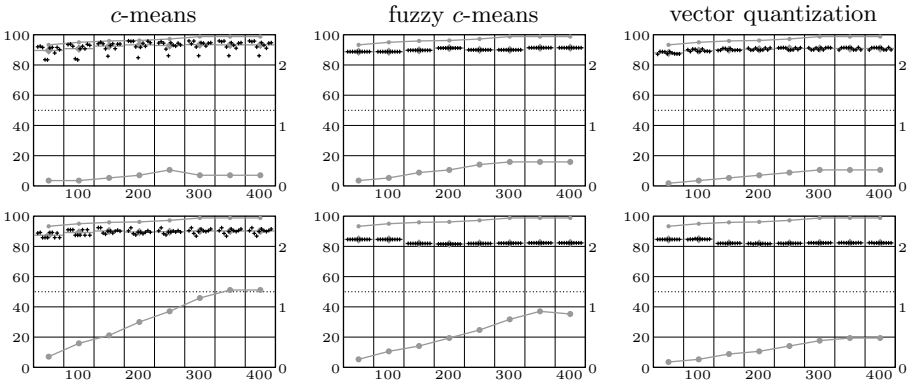


**Fig. 4.** Classification accuracy on commercial banks versus soccer (top row: document space re-scaled, bottom row: fixed cluster specific variances).

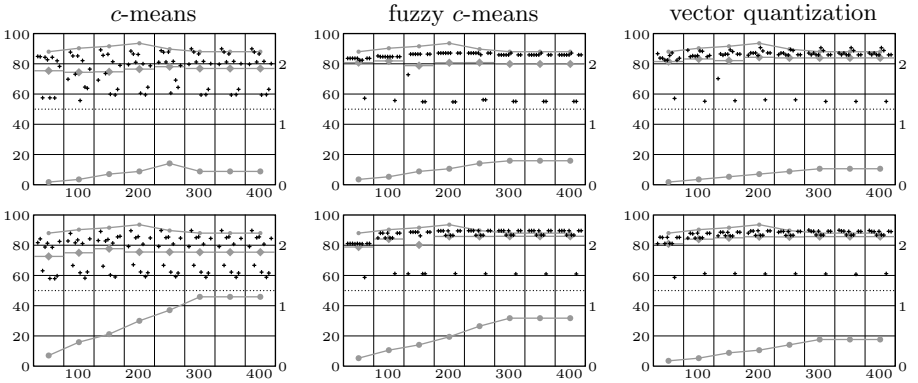


**Fig. 5.** Classification accuracy on building companies versus insurance agencies (top row: document space re-scaled, bottom row: fixed cluster specific variances).

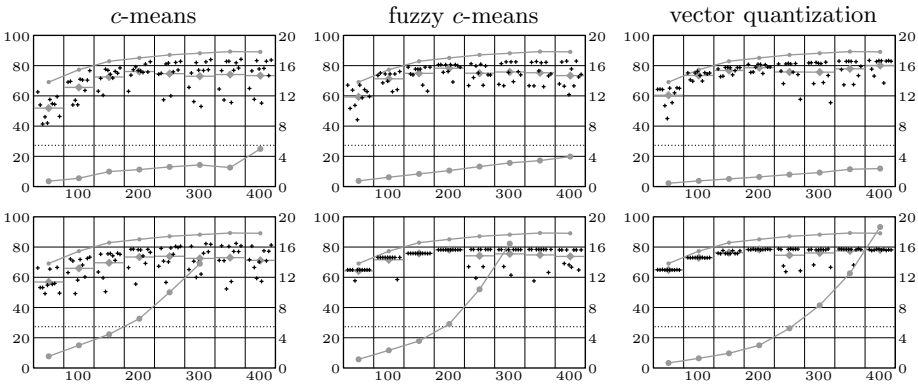


**Fig. 6.** Classification accuracy on major themes (top row: document space re-scaled, bottom row: fixed cluster specific variances).

w.r.t. a separation of the class from all other classes (cf. Section 4.4). Then we selected the first $k$ words in these lists and initialized the cluster center using the same value for each selected word and zero for all others, finally normalizing the vector to unit length. Even for fairly small values of $k$ (i.e. few selected words), this initialization results in a very stable clustering performance. Thus—similar to the idea of weight initialization in order to bias the clustering process—known describing keywords can be used in order to initialize the clustering process. In this way unwanted local minima may be avoided and the results may be stabilized.

## 5 Conclusions

Our experiments show that including prior information about the "importance" or "goodness" of a keyword for a desired class or cluster can, in principle, improve the clustering performance. However, it is fairly difficult to find a good way of scaling the documents or enhancing the cluster prototypes in an appropriate way. Scaling the document space does not yield any improvement at all. On the other hand, cluster-specific variances derived from the "importance" of index terms can slightly improve and stabilize the clustering performance. However, the gains are marginal and the approaches seem to be fairly sensitive to parameter settings.

## References


1. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY, USA 1981
2. J.C. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Dordrecht, Netherlands 1999
3. J. Bilmes. A Gentle Tutorial on the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. University of Berkeley, Tech. Rep. ICSI-TR-97-021, 1997
4. H.H. Bock. *Automatische Klassifikation*. Vandenhoeck & Ruprecht, Göttingen, Germany 1974
5. C. Borgelt and A. Nürnberger. Fast Fuzzy Clustering of Web Page Collections. *Proc. PKDD Workshop on Statistical Approaches for Web Mining (SAWM, Pisa, Italy)*. 2004 (to appear).
6. A.P. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society (Series B)* 39:1–38. Blackwell, Oxford, United Kingdom 1977
7. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. J. Wiley & Sons, New York, NY, USA 1973
8. B.S. Everitt and D.J. Hand. *Finite Mixture Distributions*. Chapman & Hall, London, UK 1981
9. G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research* 3:1289-1305, 2003
10. H. Frigui and O. Nasraoui. Simultaneaous Clustering and Dynamic Keyword Weighting for Text Douments. M.W. Berry, ed. *Survey of Text Mining*, 45–72. Springer, New York, NY USA 2003

11. I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. Pattern Analysis & Machine Intelligence* 11:773–781. IEEE Press, Piscataway, NJ, USA, 1989
12. W.R. Greiff. A Theory of Term Weighting Based on Exploratory Data Analysis. *Proc. 21st Ann. Int. Conf. on Research and Development in Information Retrieval (Sydney, Australia)*, 17–19. ACM Press, New York, NY, USA 1998
13. E.E. Gustafson and W.C. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. *Proc. 18th IEEE Conference on Decision and Control (IEEE CDC, San Diego, CA)*, 761–766, IEEE Press, Piscataway, NJ, USA 1979
14. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis.* J. Wiley & Sons, Chichester, England 1999
15. F. Klawonn and R. Kruse. Constructing a Fuzzy Controller from Data. *Fuzzy Sets and Systems* 85:177-193. North-Holland, Amsterdam, Netherlands 1997
16. A. Klose, A. Nürnberger, R. Kruse, G.K. Hartmann, and M. Richards. Interactive Text Retrieval Based on Document Similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* 25:649–654. Elsevier, Amsterdam, Netherlands 2000
17. T. Kohonen. *Learning Vector Quantization for Pattern Recognition.* Technical Report TKK-F-A601. Helsinki University of Technology, Finland 1986
18. T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Heidelberg, Germany 1995 (3rd ext. edition 2001)
19. R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering, *IEEE Transactions on Fuzzy Systems*, 1:98-110. IEEE Press, Piscataway, NJ, USA 1993
20. K.E. Lochbaum and L.A. Streeter. Combining and Comparing the Effectiveness of Latent Semantic Indexing and the Ordinary Vector Space Model for Information Retrieval. *Information Processing and Management* 25:665–676. Elsevier, Amsterdam, Netherlands 1989
21. D. Mladenic. Using Text Learning to help Web browsing. *Proc. 9th Int. Conf. on Human-Computer Interaction.* New Orleans, LA, USA 2001
22. M. Porter. An Algorithm for Suffix Stripping. *Program: Electronic Library & Information Systems* 14(3):130–137. Emerald, Bradford, United Kingdom 1980
23. G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM* 18:613–620 ACM Press, New York, NY, USA 1975
24. G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24:513–523. Elsevier, Amsterdam, Netherlands 1988
25. G. Salton, J. Allan, and C. Buckley. Automatic Structuring and Retrieval of Large Text Files. *Communications of the ACM* 37:97–108. ACM Press, New York, NY, USA 1994
26. M.P. Sinka, and D.W. Corne. A Large Benchmark Dataset for Web Document Clustering. *A. Abraham, J. Ruiz-del-Solar, and M. Köppen (eds.), Soft Computing Systems: Design, Management and Applications*, 881–890. IOS Press, Amsterdam, The Netherlands 2002
27. H. Timm, C. Borgelt, and R. Kruse. A Modification to Improve Possibilistic Cluster Analysis. *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2002, Honolulu, Hawaii).* IEEE Press, Piscataway, NJ, USA 2002
28. Y. Yang and J.O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. *Proc. 14th Int. Conf. on Machine Learning (ICML'97, Nashville, TN)*, 412–420. Morgan Kaufman, San Mateo, CA, USA 1997