Comprehensible and Accurate Cluster Labels in Text Clustering

Jerzy Stefanowski and Dawid Weiss

Institute of Computing Science, Poznan University of Technology
Piotrowo 2, 60–965 Poznań, Poland

{jerzy.stefanowski,dawid.weiss}@cs.put.poznan.pl

Abstract

The purpose of text clustering in information retrieval is to discover groups of semantically related documents. Accurate and comprehensible cluster descriptions (labels) let the user comprehend the collection's content faster and are essential for various document browsing interfaces. The task of creating descriptive, sensible cluster labels is difficult—typical text clustering algorithms focus on optimizing proximity between documents inside a cluster and rely on keyword representation for describing discovered clusters. In the approach called *Description Comes First* (DCF) cluster labels are as important as document groups—DCF promotes machine discovery of comprehensible candidate cluster labels later used to discover related document groups. In this paper we describe an application of DCF to the k-Means algorithm, including results of experiments performed on the 20-newsgroups document collection. Experimental evaluation showed that DCF does not decrease the metrics used to assess the quality of document assignment and offers good cluster labels in return. The algorithm utilizes search engine's data structures directly to scale to large document collections.

Introduction

Organizing unstructured collections of textual content into semantically related groups, from now on referred to as *text clustering* or *clustering*, provides unique ways of digesting large amounts of information. In the context of information retrieval and text mining, a general definition of clustering is the following: given a large set of documents, automatically discover diverse subsets of documents that share a similar topic. In typical applications input documents are first transformed into a mathematical model where each document is described by certain features. The most popular representation for text is the *vector space model* [Salton, 1989]. In the VSM, documents are expressed as rows in a matrix, where columns represent unique terms (features) and the intersection of a column and a row indicates the importance of a given word to the document.

A model such as the VSM helps in calculation of similarity between documents (angle between document vectors) and thus facilitates application of various known (or modified) numerical clustering algorithms. While this is sufficient for many applications, problems arise when one needs to construct some *representation* of the discovered groups of documents—a label, a symbolic description for each cluster, something to represent the information that makes

documents inside a cluster similar to each other and that would convey this information to the user. Cluster labeling problems are often present in modern text and Web mining applications with document browsing interfaces.

The process of returning from the mathematical model of clusters to comprehensible, explanatory labels is difficult because text representation used for clustering rarely preserves the inflection and syntax of the original text. Clustering algorithms presented in literature usually fall back to the simplest form of cluster representation—a list of cluster's *keywords* (most "central" terms in the cluster). Unfortunately, keywords are stripped from syntactical information and force the user to manually find the underlying concept which is often confusing.

Motivation and Related Works

The user of a retrieval system judges the clustering algorithm by what he sees in the output—clusters' descriptions, not the final model which is usually incomprehensible for humans. The experiences with the text clustering framework Carrot^2 (www.carrot2.org) resulted in posing a slightly different research problem (aligned with clustering but not exactly the same). We shifted the emphasis of a clustering method to providing comprehensible and accurate cluster labels in addition to discovery of document groups. We call this problem *descriptive clustering*: **discovery of diverse groups of semantically related documents associated with a meaningful, comprehensible and compact text labels**.

This definition obviously leaves a great deal of freedom for interpretation because terms such as *meaningful* or *accurate* are very vague. We narrowed the set of requirements of descriptive clustering to the following ones:

— *comprehensibility* understood as grammatical correctness (word order, inflection, agreement between words if applicable);
— *conciseness* of labels. Phrases selected for a cluster label should minimize its total length (without sacrificing its comprehensibility);
— *transparency* of the relationship between cluster label and cluster content, best explained by ability to answer questions as: "Why was this label selected for these documents?" and "Why is this document in a cluster labeled X?".

Little research has been done to address the requirements above. In the STC algorithm authors employed frequently recurring phrases as both document similarity feature and final cluster description [Zamir and Etzioni, 1999]. A follow-up work [Ferragina and Gulli, 2004] showed how to avoid certain STC limitations and use non-contiguous phrases (so-called approximate sentences). A different idea of 'label-driven' clustering appeared in *clustering with committees* algorithm [Pantel and Lin, 2002], where strongly associated terms related to unambiguous concepts were evaluated using semantic relationships from WordNet. We introduced the DCF approach in our previous work [Osiński and Weiss, 2005] and showed its feasibility using an algorithm called Lingo. Lingo used singular value decomposition of the term-document matrix to select good cluster labels among candidates extracted from the text (frequent phrases). The algorithm was designed to cluster results from Web search engines (short snippets and fragmented descriptions of original documents) and proved to provide diverse meaningful cluster labels.

Lingo's weak point is its limited scalability to full or even medium sized documents. In this

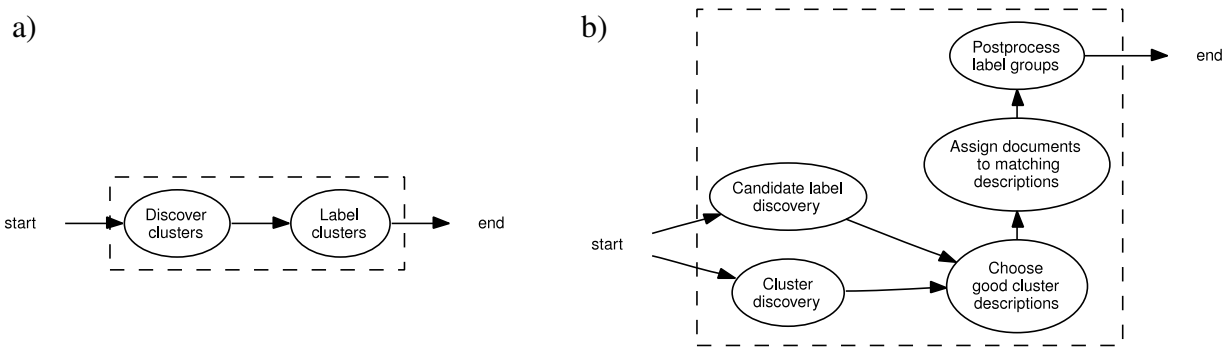

Figure 1: Steps of a clustering procedure in a typical clustering algorithm (a) and in the DCF approach (b). Note the possible parallelization in DCF.

paper we were interested in finding how to apply DCF to large document collections. The goals of this paper can be thus summarized by the following questions.

— Is it possible to apply DCF to cluster large document collections, yielding comprehensible and accurate cluster labels?
— Does the quality of clustering (measured as correct document assignments) change when DCF and traditional clustering are applied on the same test collection?

To answer these questions we experimented with a well known clustering algorithm k-Means and its DCF-modified version called Descriptive k-Means. We perform an experimental evaluation of document allocation quality using both algorithms on a popular test collection 20-newsgroups.

An Overview of Description Comes First

In a typical text clustering algorithm, cluster labeling follows cluster discovery, as illustrated in Figure 1. A mathematical model of document representation, such as the vector space model, facilitates calculating document proximity, but loses the information essential to construct human-friendly cluster descriptions: syntax. One solution to this problem is to use features that are immediately intuitive to the user, for example entire phrases appearing in the text. This idea was employed in the STC algorithm for example. But phrase-based clustering introduces its own difficulties: meaningless, but frequent phrases are difficult to identify and filter out, the notion of a phrase as an ordered sequence of words is not natural to many languages with less strict syntax (compared to English).

The description comes first approach changes the troublesome conventional order of a typical clustering algorithm by separating the processes of *candidate cluster label discovery* and *document clustering* (see Figure 1). In DCF these two phases are split and independent:

— *candidate label discovery* phase is responsible for collecting all phrases potentially useful for creating good cluster labels,
— *cluster discovery* provides a data model and information about document groups present in the input data.

Once candidate labels and document clusters are known, we proceed to the *matching step*. Out of all candidate labels we select only these which are „supported" (most similar) to cluster centroids discovered in parallel. The model of clusters is then discarded entirely and

documents are reassigned to each selected label by querying the input collection of documents looking for documents that match the label's phrase (approximate match is allowed).

Note that there are a few advantages of DCF. First, candidate cluster labels can be extracted from the input text based on frequency or some other method (identification of noun chunks, for example). Alternatively, candidate phrases can come from a completely different source or even a predefined ontology (to guarantee their meaning is comprehensible for end users). Second, cluster discovery and candidate phrase extraction are completely independent and can be easily parallelized. In fact, candidate phrase extraction can be performed incrementally as the documents are added to the system and cluster discovery can be performed on a sample of input collection's documents (as demonstrated in Descriptive k-Means). Third, note the relationship between cluster and label discovery: unclear (model-wise) clusters are unlikely to find a matching candidate label and should be discarded. A query constructed for each selected candidate cluster label must still collect enough documents to actually form a final cluster. We thus increase the likelihood that groups of documents in the output are really relevant and well described.

A data-mining way of looking at the DCF approach is to consider candidate cluster label discovery as a selection of meaningful *patterns*. These patterns are then used as seeds around which final clusters of documents exhibiting these patterns ar built. Note the freedom of choosing the actual algorithm used to discover the model of clusters (groups of documents)—because cluster model is used only internally to select good candidate labels, the difficult problem of proper cluster labeling never has a chance to surface.

Descriptive k-Means Algorithm

The *Descriptive k-Means* is algorithm is a DCF-adjusted modification of k-Means. We considered a few different "baseline" text clustering algorithms and finally selected k-Means for a few reasons.

— k-Means is widely recognized and used. It has a few limitations when applied to text clustering—unique document-to-cluster partitioning, spherical clusters and the requirement to specify the number of clusters in advance, but our intention and inspiration was to demonstrate that even this inconvenient algorithm can be adopted to the DCF approach and yield good results.
— We had very good experiences with SVD decomposition in Lingo (our first algorithm based on DCF). Reducing the dimensionality of the term-document matrix reveals the latent relationships between terms and, although this is difficult to verify, approximates "dominating concepts" present in the documents. It was previously shown that SVD decomposition can be approximated with cluster centroids discovered by k-Means [Dhillon and Modha, 2001] and we hoped to create a very scalable algorithm consistent with the concepts once proven in Lingo.
— Finally, k-Means is a baseline algorithm very often used for comparisons of clustering quality. Any definition of quality in text clustering is highly controversial and prone to subjective bias because no single ideal result exists [Manning and Schütze, 1999]. Our motivation was to evaluate DKM and k-Means in the same experiment conditions to see if DCF at least retains and hopefully improves the "quality" of the output document assignment. Comprehensibility of cluster labels is not taken into account in such evaluation and was not our primary objective (we know that cluster candidate selection procedure such as

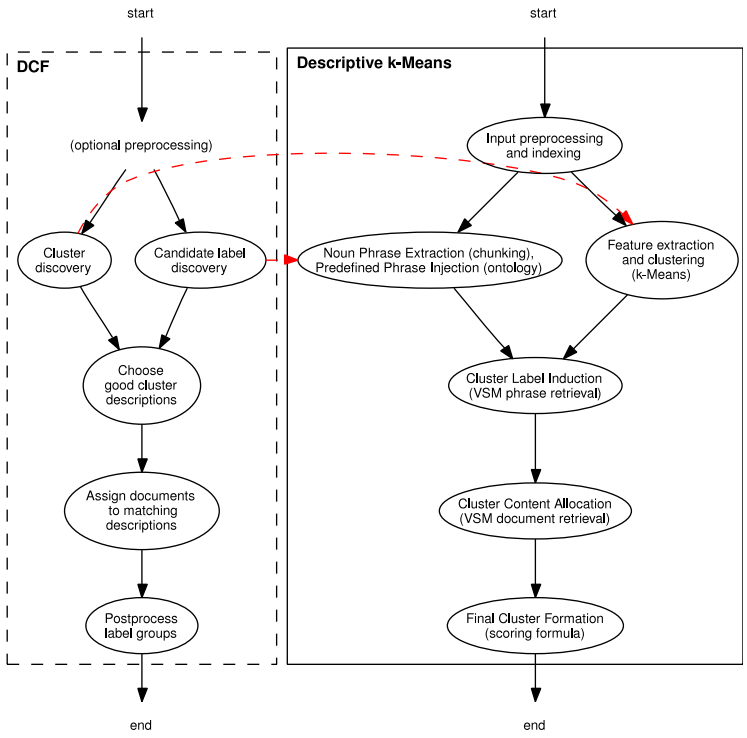

Figure 2: Elements of DKM matched against the generic DCF approach.

extraction of frequent phrases or noun phrases yields more comprehensible results than the keyword-based representation).

We eventually decided to use the following version of k-Means algorithm.

— The initial maximum number of clusters k must be given in advance (we did not try to estimate it). Note that the final number of clusters may be smaller, depending on how many matching label candidates we can find for the discovered k clusters. The fact that k must be given in advance is of course a weak point and target for future work.
— The initial state of cluster centroid vectors is computed by taking a random subset of input documents and finding k most diverse elements in this subset. This method of bootstrapping k-Means is reportedly stable and leads to good cluster diversity [Dhillon et al., 2001].
— The convergence criterion is an alternative of two conditions: either there are no more reassignments of documents to clusters, or the global gradient criterion (sum of distances to cluster centroids) no longer significantly changes (this condition is denoted as r_{min} and τ_{min} in line 32, Figure 3).

Algorithm Outline and Implementation

Major steps of the algorithm (further denoted DKM) and their relation to DCF is shown in Figure 2, verbose pseudocode of the algorithm can be found in Figure 3. We discuss each step and its implementation details in the following paragraph.

We adopted plain k-Means to DCF by splitting label candidate discovery and cluster discovery into two independent steps. In the first step, we extract candidate cluster labels from the input collection and store them in a disk-based data structure with an associated inverted index allowing fast lookups [Baeza-Yates and Ribeiro-Neto, 1999] (we use Lucene library for

this purpose, lucene.apache.org). In our experiments we used two different candidate label extraction methods: frequent phrase extraction (implemented with suffix trees), and simple linguistic processing (noun phrase extraction with a trained statistical chunker).

A similar inverted index is built for input documents. In our implementation cluster label candidate extraction and document indexing run in parallel at the time when documents are added to the system. Note that both are fairly independent and can be easily distributed over a large number of processing units.

In the next step, we build a model of clusters using plain k-Means, taking advantage of the fact that good approximation of cluster centroids can be discovered by using just a subsample of the original collection [Goswami et al., 2004]. Each document in a sample is preprocessed to the standard term-vector representation. We used various feature weighting schemes—pointwise mutual information, *tf-idf* and their modifications [Baeza-Yates and Ribeiro-Neto, 1999]—the results were very similar regardless of the method used.

In the merging step, we search for candidate phrases "closest" to cluster centroids. To handle this step efficiently, we build a boolean query with terms and their weights taken from the cluster centroid's representation and execute such query against the index of label candidates. The result is a similarity-ordered list of best matching labels from which we select a few top scoring phrases to become potential cluster labels (we call them *pattern phrases*). We adjust the similarity score of each label with a function of the label phrase's length to give preference to more concise labels (line 39 in Figure 3).

Finally, for each pattern phrase we attempt to find its matching documents. We reuse the inverted index structure again and build queries which select documents that contain all words in the pattern phrase, preferably retaining their exact order, but allowing minor word reorderings and injections of other words inside the phrase. We thus allow documents where the pattern phrase is slightly distorted, for example for "Hillary Clinton", we allow documents containing: "Hillary Rodham Clinton". When not enough documents can be found for a pattern phrase, it is discarded. Note that the relationship between a cluster's label and its contents in DKM is very explicit—the label must be present in each document inside it. We believe this *transparency* of relationship between cluster labels and their content is in general one of the most desirable features in document browsing interfaces.

Discussion of Algorithm's Complexity

Estimating computational complexity of DKM depends on the method used to extract candidate cluster labels. Linguistic preprocessing methods rarely specify their computational complexity and are often iterative heuristics. Our experiments show that noun phrase extraction was by far the most time-consuming element of the entire process. An alternative label candidate extraction method—frequent phrase extraction—was much faster; we used suffix trees whose construction is the order of $O(n)$, where n is the number of input symbols. Another element of the algorithm is clustering. According to Arthur and Vassilvitskii [Arthur and Vassilvitskii, 2006] k-Means is the order of $O\left(n^{2+2/d}(\frac{D}{\delta})^2 2^{2n/d}\right)$, where δ is the smoothness factor, D is the diameter of the pointset, n is the number of documents and d is the number of dimensions of the feature space. While in theory k-Means may be costly, it is known that encountering pessimistic data scenarios in practice is very uncommon. Additionally, the algorithm can be (and usually is) interrupted after it converges and additional iterations are not changing significantly affecting the result. To further decrease the cost of

```
1: D ← a set of input documents
2: k ← number of "topics" expected in the input (used for k-Means).

   /* Preprocessing */
3: I_D ← empty inverted index of documents;
4: I_P ← empty inverted index of candidate cluster labels;
5: for all d ∈ D do
6:    I_D ←index— d; /* add d to index I_D */
7:    if extract candidate labels then
8:       T = a set of noun phrases and/or frequent phrases extracted from d;
9:       for all t ∈ T do
10:         I_P ←index— t; /* add t to index I_P */
11:      end for
12:   end if
13: end for
14: if predefined labels available then
15:   for all t ∈ (a set of predefined labels) do
16:      I_P ←index— t; /* add t to index I_P */
17:   end for
18: end if

   /* Topic Detection (k-Means) */
19: for all d ∈ D do
20:   f_d = extract_features(d); /* Feature extraction; mutual information, tfidf... */
21: end for
22: C = initialize(D); /* Initialize cluster centroids by finding most dissimilar documents. */
23: repeat
24:   τ = 0;
25:   for all d ∈ D do
26:      c_d = reassign_to_closest_centroid(f_d, C);
27:      τ = τ + sim(f_d, c_d);
28:   end for
29:   for all c ∈ C do
30:      update_centroid(c);
31:   end for
32: until reassignments > r_min and τ > τ_min

   /* Select pattern phrases */
33: P ← empty set of candidate labels;
34: F ← empty set of final pairs (description, documents);
35: for all c ∈ C do
36:   q_vsm = a boolean query for terms in c, terms boosted with the centroid's weights f_c;
37:   P_c = search(I_P, q_vsm); /* Execute the query against the index of labels. */
38:   for all p ∈ P_c do
39:      s = length_penalty(p) × hit_score(p); /* Penalize the score with a length function. */
40:      if s > s_min then
41:         P = P ∪ p; /* Add to pattern phrases. */
42:      end if
43:   end for
44: end for

   /* Assign documents to pattern phrases */
45: for all p ∈ P do
46:   q_vsm = construct a phrase query for pattern phrase p;
47:   R_p = search(I_D, q_vsm); /* Execute the query against documents. */
48:   if R_p contains more than minimum documents then
49:      F = F ∪ (p, R_p); /* Add a new cluster with label p and documents R_p. */
50:   end if
51: end for
52: Return the final set of clusters F.
```

Figure 3: Pseudocode of the Descriptive k-Means algorithm.

clustering we cluster only a representative subset of the entire documents collection—a subset big enough to be able to select good pottern phrases out of the candidate set. We found the performance of the entire process to be satisfactory in practice—clustering took from a few seconds to a few minutes, depending on the number of sampled documents, length of document vectors and number of centroids (k).

Experimental Evaluation

Experiment Goals and Assumptions

The goals of descriptive clustering slightly differ from these of pure clustering—we try to find coherent, *properly described* groups of documents. The experiment concentrates on *finding if DCF-based algorithms are not degrading clustering performance of their baseline counterparts*. The advantage of DCF is in providing better cluster labels, but we are quite sceptical about objective measurement of this aspect (especially using user-based surveys). We believe improved cluster labels are an outcome of the candidate cluster label extraction phase, complemented with cluster label selection and matching. Obviously whether this claim is true should be validated in the future. The experiment's goals can be therefore summarized by the following questions.

— Description Comes First has two elements that may degrade clustering quality: approximation of abstract topics inside discovered clusters with pattern phrases and document assignment to pattern phrases. Does DCF indeed improve, degrade or retain clustering quality?
— Given two preprocessing methods, one extracting frequent phrases, the second extracting noun phrases, is the quality of clustering different and how does it change?
— Does document subsampling used in k-Means change the clustering quality?

Data Set and Evaluation Methodology

To have a cross-comparison with previous research we decided to use a widely known *20-newsgroups* data set consisting of approximately 20000 mailing list messages partitioned nearly evenly into 20 different groups. Each group corresponds to a different topic, although some of them are closely related. We chose a subset of the original data set with removed redundancies and empty documents called a "bydate split" and consisting of 18941 documents.

Three combinations of algorithms undergo the evaluation: DKM with English noun phrases extractor, DKM with frequent phrases extractor and pure k-Means. Descriptive k-Means is in two variants because we wanted to see if different candidate label selection affects clustering quality. The "baseline" k-Means implementation was the same as the one used internally in DKM.

Most evaluation measures assume complete partitioning of the input data set into k disjoint subsets, while DCF-based algorithms are designed to produce partial partitioning into an unknown number of pattern phrases. We modified DKM slightly to adjust it to these requirements: pattern phrases selected by each cluster centroid form one merged cluster with a union of documents assigned for each pattern phrase. Any remaining unassigned documents are excluded from the result.

The experiment is quantitative rather than qualitative. For every run of the experiment we compared the clustered result against the ground truth partitioning, calculating the following

Threshold	k-Means	DKM
maximum reassignments $r_{\min}$	20	20
minimal global objective function increase τ	0.001	0.001
minimum documents allocated to a pattern phrase	n/a	10
minimum documents in a final cluster	n/a	5

Table 1: Valus of thresholds used in the experiment.

quality indicators: average cluster purity, average normalized entropy, average F-measure and average contamination measure. Entropy and F-measure are widely used and their details are available for example in [Cheng et al., 2005, Dhillon et al., 2001]. Cluster purity gives the average ratio of a dominating class in each cluster to the cluster size. Contamination measure [Weiss, 2006] for cluster k_i is defined as the number of pairs of objects found in the same cluster k_i, but not coexisting in any of the original partitions, divided by the worst case scenario—maximum number of such bad pairs in k_i; for pure clusters, cluster contamination measure equals 0, for an even mix of documents from all original partitions the measure equals 1.

Thresholds and Variables

Manually tuned values of thresholds were used for k-Means and DKM, their precise values are presented in Table 1. The number of expected clusters k was set to 20 for both algorithms (the "true" number of groups in the input). We clustered the ground truth set of documents 5 times for each possible combination of the following variable elements:

— *sample size*—size of the sample of documents from the original data set,
— *feature type*—type of feature weighting formulas used for feature selection; we used mutual information, discounted mutual information, plain *tf-idf* and its version downplaying the count of terms in the document (with a square root of the *tf* component).
— *feature vector length*—length of the document's feature vector, set to 30, 50, 70 or 100 elements.

Results

First of all, we should probably say that we expected Descriptive k-Means to be slightly *worse* at clustering documents from a predefined collection. After all, the DCF approach has a slightly different goal compared to traditional clustering and will not try to group documents for which there are no sensible descriptions. We were quite surprised when it turned out that the experiment's results show an *increase* in quality in most aspects of the analysis.

Figure 4 shows an average contamination for each feature selection type over all experiment runs. Surprisingly, descriptive clustering in both variants is much less contaminated than the baseline k-Means. This is confirmed by two other quality metrics—average entropy and average purity. Because of the paper size limits, we have to skip precise figures (see [Weiss, 2006] for full results); in these metrics, however, k-Means is slightly better when pointwise mutual information is used for feature weighting. This phenomenon can be explained—pointwise mutual information is known to boost low frequency elements [Manning and Schütze, 1999], if such elements are selected for the cluster centroid it is difficult to find a candidate label with such terms.

Figure 5 illustrates an average size of a cluster depending on the size of the document's

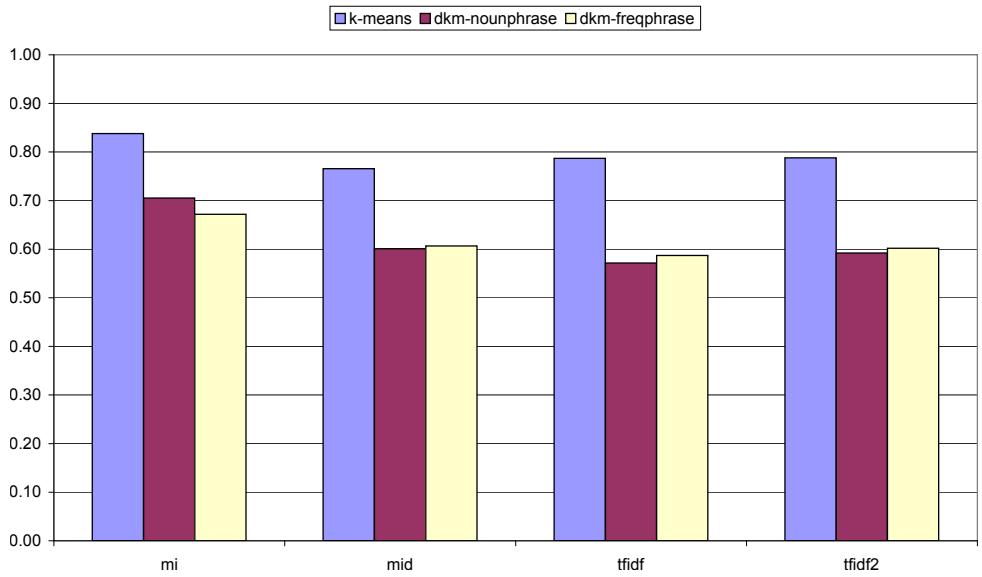

Figure 4: Average cluster contamination depending on the feature type (higher values indicate more contaminated clusters). See *Thresholds and Variables* section on page 9 for an explanation of acronyms.

representation (grouped by sample size). Descriptive k-Means produces significantly smaller groups of documents—only about 37–38% of the original number of documents were placed in clusters, the rest was left unassigned. Increasing the size of the sample affects the average cluster size gradually and in a minor way. We can speculate that DCF tends to produce smaller, but more accurate clusters. This hypothesis was also confirmed by looking at contingency matrices created for the results.

The difference in clustering quality between the two DKM versions—one using noun phrases and the other frequent phrases—was in most cases neglectable, although a preference towards noun phrases seemed to exist. This was a bit surprising (we hoped for noun phrases to be much better pattern phrases). Manual inspection of candidate cluster labels showed that many noun phrases were in fact incorrect due to a low-quality NP-chunker we used. Perhaps if we prepared noun phrases offline the quality would increase to a significant difference.

Clustering a sample of the input to retrieve cluster centroids proved to be very effective. Only the smallest sample of 2000 documents had different quality characteristic. Samples of 5000, 7000 and 9000 documents were very much alike (results for the entire data set were very similar). This confirms earlier reports that sampling can be successfully used to decrease the computational effort needed to cluster large numbers of documents using k-Means.

Although it was not among the experiment's objectives, we inspected cluster labels of several clustered instances. For example, topmost labels from a cluster most likely corresponding to a group called soc.religion.christian were: *Lord Jesus Christ*, *salvation through our Lord Jesus Christ*, *God does not exist*, *existence of God*, *grace of God*. Another sample showed cluster labels concerned with the Muslim/ Jew communities: *Palestinians, Israel*, *Israel Gaza*, *Serbs, Croats and Muslims*, *Israeli Jews*, *Bosnian Serbs and Bosnian Muslims*. In general, the quality of cluster labels was, as we suspected, very satisfactory. Again, note that we relied on the candidate cluster label extraction method to provide the best candidates possible, it was not our intention to measure if frequent phrases or noun phrases were truly comprehensive—such

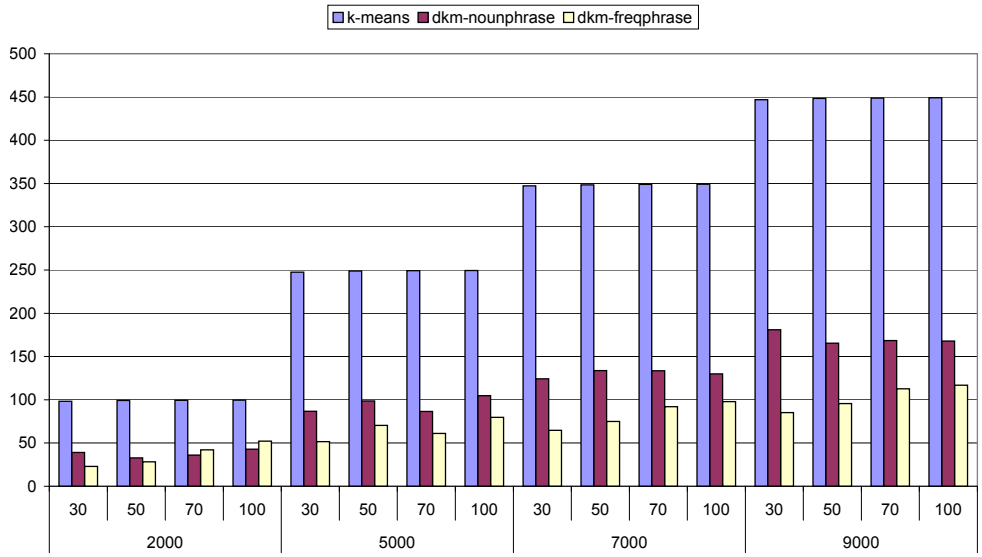

Figure 5: Average size of a cluster depending on the feature vector length and sample size.

an experiment would require a wholly different approach.

Summary and Conclusions

In this paper we considered the problem of clustering large collections of documents, while keeping cluster labels comprehensive to humans. We shortly outlined the description comes first approach in which cluster construction and potential cluster label discovery are split into concurrent phases and merged in the end. Then we described how to modify the classic k-Means algorithm so that it conforms to the DCF approach. Finally, we presented the results of a computational experiment in which we quantitatively compared the clustering quality of both the modified and the baseline k-Means algorithm.

The experimental results showed that Descriptive k-Means slightly increases clustering quality, but most of all, provides a possibility of describing the output clusters in a human-accessible way (using frequent or noun phrases).

Acknowledgement. This research has been supported by an internal grant from Poznan University of Technology.

References

[Arthur and Vassilvitskii, 2006] Arthur, D. and Vassilvitskii, S. (2006). On the worst case complexity of the k-means method. 22nd Annual ACM Symposium on Computational Geometry.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

[Cheng et al., 2005] Cheng, D., Vempala, S., Kannan, R., and Wang, G. (2005). A divide-and-merge methodology for clustering. In Li, C., editor, *Proceedings of the Twenty-fourth*

ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pages 196–205. ACM.

[Dhillon et al., 2001] Dhillon, I. S., Fan, J., and Guan, Y. (2001). Efficient clustering of very large document collections. In R. Grossman, C. Kamath, V. K. and Namburu, R., editors, *Data Mining for Scientific and Engineering Applications*. Kluwer Academic Publishers.

[Dhillon and Modha, 2001] Dhillon, I. S. and Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1–2):143–175.

[Ferragina and Gulli, 2004] Ferragina, P. and Gulli, A. (2004). The anatomy of snaket: A hierarchical clustering engine for web-page snippets. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *PKDD*, volume 3202 of *Lecture Notes in Computer Science*, pages 506–508. Springer.

[Goswami et al., 2004] Goswami, A., Jin, R., and Agrawal, G. (2004). Fast and exact out-of-core k-means clustering. In *ICDM*, pages 83–90. IEEE Computer Society.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[Osiński and Weiss, 2005] Osiński, S. and Weiss, D. (2005). A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54.

[Pantel and Lin, 2002] Pantel, P. and Lin, D. (2002). Document clustering with committees. In *Proceedings of SIGIR'02*, Tampere, Finland.

[Salton, 1989] Salton, G. (1989). *Automatic Text Processing — The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley.

[Weiss, 2006] Weiss, D. (2006). *Descriptive Clustering as a Method for Exploring Text Collections*. PhD thesis, Poznań University of Technology, Poznań, Poland.

[Zamir and Etzioni, 1999] Zamir, O. and Etzioni, O. (1999). Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374.