Document Clustering with Committees

Patrick Pantel and Dekang Lin
University of Alberta
Department of Computing Science
Edmonton, Alberta T6H 2E1 Canada
{ppantel, lindek}@cs.ualberta.ca

ABSTRACT

Document clustering is useful in many information retrieval tasks: document browsing, organization and viewing of retrieval results, generation of Yahoo-like hierarchies of documents, etc. The general goal of clustering is to group data elements such that the intra-group similarities are high and the inter-group similarities are low. We present a clustering algorithm called CBC (Clustering By Committee) that is shown to produce higher quality clusters in document clustering tasks as compared to several well known clustering algorithms. It initially discovers a set of tight clusters (high intra-group similarity), called committees, that are well scattered in the similarity space (low inter-group similarity). The union of the committees is but a subset of all elements. The algorithm proceeds by assigning elements to their most similar committee. Evaluating cluster quality has always been a difficult task. We present a new evaluation methodology that is based on the editing distance between output clusters and manually constructed classes (the answer key). This evaluation measure is more intuitive and easier to interpret than previous evaluation measures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval---*Clustering*.

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Document clustering, evaluation methodology, machine learning, document representation.

1. INTRODUCTION

Document clustering was initially proposed for improving the precision and recall of information retrieval systems [18]. Because clustering is often too slow for large corpora and has indifferent performance [8], document clustering has been used more recently in document browsing [3], to improve the organization and viewing of retrieval results [6], to accelerate nearest-neighbor search [1] and to generate Yahoo-like hierarchies [12].

Common characteristics of document clustering include:

- there is a large number of documents to be clustered;
- the number of output clusters may be large;
- each document has a large number of features; e.g., the features may include all the terms in the document; and
- the feature space, the union of the features of all documents, is even larger.

In this paper, we propose a clustering algorithm, CBC (Clustering By Committee), which produces higher quality clusters in document clustering tasks as compared to several well known clustering algorithms. Many clustering algorithms represent a cluster by the centroid of all of its members (e.g., K-means) [13] or by a representative element (e.g., K-medoids) [10]. When averaging over all elements in a cluster, the centroid of a cluster may be unduly influenced by elements that only marginally belong to the cluster or by elements that also belong to other clusters. To illustrate this point, consider the task of clustering words. We can use the contexts of words as features and group together the words that tend to appear in similar contexts. For instance, U.S. state names can be clustered this way because they tend to appear in the following contexts:

(List A)	___ appellate court	campaign in ___
	___ capital	governor of ___
	___ driver's license	illegal in ___
	___ outlaws sth.	primary in ___
	___'s sales tax	senator for ___

If we create a centroid of all the state names, the centroid will also contain features such as:

(List B)	___'s airport	archbishop of ___
	___'s business district	fly to ___
	___'s mayor	mayor of ___
	___'s subway	outskirts of ___

because some of the state names (like *New York* and *Washington*) are also names of cities.

SIGIR'02, August 11-15, 2002, Tampere, Finland.

Using a single representative from a cluster may be problematic too because each individual element has its own idiosyncrasies that may not be shared by other members of the cluster.

CBC constructs the centroid of a cluster by averaging the feature vectors of a subset of the cluster members. The subset is viewed as a committee that determines which other elements belong to the cluster. By carefully choosing committee members, the features of the centroid tend to be the more typical features of the target class. For example, our system chose the following committee members to compute the centroid of the state cluster: Illinois, Michigan, Minnesota, Iowa, Wisconsin, Indiana, Nebraska and Vermont. As a result, the centroid contains only features like those in List A.

Evaluating clustering results is a very difficult task. We introduce a new methodology that is based on the editing distance between clustering results and manually constructed classes (the answer key). We argue that it is easier to interpret the results of this evaluation measure than the results of previous measures.

The remainder of this paper is organized as follows. In the next section, we review related clustering algorithms that are commonly used in document clustering. Section 3 describes our representational model and in Section 4 we present the CBC algorithm. The evaluation methodology and experimental results are presented in Sections 5 and 6. In Section 7, we show an example application of CBC. Finally, we conclude with a discussion and future work.

2. RELATED WORK

Generally, clustering algorithms can be categorized as hierarchical and partitional. In hierarchical agglomerative algorithms, clusters are constructed by iteratively merging the most similar clusters. These algorithms differ in how they compute cluster similarity. In single-link clustering [16], the similarity between two clusters is the similarity between their most similar members while complete-link clustering [11] uses the similarity between their least similar members. Average-link clustering [5] computes this similarity as the average similarity between all pairs of elements across clusters. The complexity of these algorithms is $O(n^2\log n)$, where n is the number of elements to be clustered [7]. These algorithms are too inefficient for document clustering tasks that deal with large numbers of documents. In our experiments, one of the corpora we used is small enough (2745 documents) to allow us to compare CBC with these hierarchical algorithms.

Chameleon is a hierarchical algorithm that employs dynamic modeling to improve clustering quality [9]. When merging two clusters, one might consider the sum of the similarities between pairs of elements across the clusters (e.g. average-link clustering). A drawback of this approach is that the existence of a single pair of very similar elements might unduly cause the merger of two clusters. An alternative considers the number of pairs of elements whose similarity exceeds a certain threshold [4]. However, this may cause undesirable mergers when there are a large number of pairs whose similarities barely exceed the threshold. Chameleon clustering combines the two approaches.

Most often, document clustering employs *K*-means clustering since its complexity is linear in n, the number of elements to be clustered. *K*-means is a family of partitional clustering algorithms. The following steps outline the basic algorithm for generating a set of K clusters:

1. randomly select K elements as the initial centroids of the clusters;
2. assign each element to a cluster according to the centroid closest to it;
3. recompute the centroid of each cluster as the average of the cluster's elements;
4. repeat Steps 2-3 for T iterations or until the centroids do not change, where T is a predetermined constant.

K-means has complexity $O(K\times T\times n)$ and is efficient for many document clustering tasks. Because of the random selection of initial centroids, the resulting clusters vary in quality. Some sets of initial centroids lead to poor convergence rates or poor cluster quality.

Bisecting *K*-means [17], a variation of *K*-means, begins with a set containing one large cluster consisting of every element and iteratively picks the largest cluster in the set, splits it into two clusters and replaces it by the split clusters. Splitting a cluster consists of applying the basic *K*-means algorithm α times with K=2 and keeping the split that has the highest average element-centroid similarity.

Hybrid clustering algorithms combine hierarchical and partitional algorithms in an attempt to have the high quality of hierarchical algorithms with the efficiency of partitional algorithms. Buckshot [3] addresses the problem of randomly selecting initial centroids in *K*-means by combining it with average-link clustering. Cutting et al. claim its clusters are comparable in quality to hierarchical algorithms but with a lower complexity. Buckshot first applies average-link to a random sample of $\sqrt{n}$ elements to generate K clusters. It then uses the centroids of the clusters as the initial K centroids of *K*-means clustering. The complexity of Buckshot is $O(K\times T\times n + n\log n)$. The parameters K and T are usually considered to be small numbers. Since we are dealing with a large number of clusters, Buckshot and *K*-means become inefficient in practice. Furthermore, Buckshot is not always suitable. Suppose one wishes to cluster 100,000 documents into 1000 newsgroup topics. Buckshot could generate at most 316 initial centroids.

3. REPRESENTATION

CBC represents elements as feature vectors. The features of a document are the terms (usually stemmed words) that occur within it and the value of a feature is a statistic of the term. For example, the statistic can simply be the term's frequency, *tf*, within the document. In order to discount terms with low discriminating power, *tf* is usually combined with the term's inverse document frequency, *idf*, which is the inverse of the percentage of documents in which the term occurs. This measure is referred to as *tf-idf* [15]:

$$\textit{tf-idf} = \textit{tf} \times \log \textit{idf}$$

We use the mutual information [2] between an element (document) and its features (terms).

In our algorithm, for each element e, we construct a **frequency count vector** $C(e) = (c_{e1}, c_{e2}, \ldots, c_{em})$, where m is the total number of features and c_{ef} is the frequency count of feature f occurring in element e. In document clustering, e is a document and c_{ef} is the term frequency of f in e. We construct a **mutual information vector** $MI(e) = (mi_{e1}, mi_{e2}, \ldots, mi_{em})$, where mi_{ef} is the mutual information between element e and feature f, which is defined as:

$$mi_{ef} = log \frac{\frac{c_{ef}}{N}}{\frac{\sum_i c_{if}}{N} \times \frac{\sum_j c_{ej}}{N}}$$

where $N = \sum_i \sum_j c_{ij}$ is the total frequency count of all features of all elements.

We compute the similarity between two elements e_i and e_j using the *cosine coefficient* [15] of their mutual information vectors:

$$sim(e_i, e_j) = \frac{\sum_f mi_{e_i f} \times mi_{e_j f}}{\sqrt{\sum_f mi_{e_i f}^{\,2} \times \sum_f mi_{e_j f}^{\,2}}}$$

4. ALGORITHM

CBC consists of three phases. In Phase I, we compute each element's top-k similar elements. In our experiments, we used k = 20. In Phase II, we construct a collection of tight clusters, where the elements of each cluster form a **committee**. The algorithm tries to form as many committees as possible on the condition that each newly formed committee is not very similar to any existing committee. If the condition is violated, the committee is simply discarded. In the final phase of the algorithm, each element is assigned to its most similar cluster.

4.1 Phase I: Find top-similar elements

Computing the complete similarity matrix between pairs of elements is obviously quadratic. However, one can dramatically reduce the running time by taking advantage of the fact that the feature vector is sparse. By indexing the features, one can retrieve the set of elements that have a given feature. To compute the top similar elements of an element e, we first sort the mutual information vector $MI(e)$ and then only consider a subset of the features with highest mutual information. Finally, we compute the pairwise similarity between e and the elements that share a feature from this subset. Since high mutual information features tend not to occur in many elements, we only need to compute a fraction of the possible pairwise combinations. With 18,828 elements, Phase I completes in 38 minutes. Using this heuristic, similar words that share only low mutual information features will be missed by our algorithm. However, in our experiments, this had no visible impact on cluster quality.

4.2 Phase II: Find committees

The second phase of the clustering algorithm recursively finds tight clusters scattered in the similarity space. In each recursive step, the algorithm finds a set of tight clusters, called committees, and identifies residue elements that are not covered by any committee. We say a committee **covers** an element if the element's similarity to the centroid of the committee exceeds some high similarity threshold. The algorithm then recursively attempts to find more committees among the residue elements. The output of the algorithm is the union of all committees found in each recursive step. The details of Phase II are presented in Figure 1.

```
Input:   A list of elements $E$ to be clustered, a similarity
         database $S$ from Phase I, thresholds $\theta_1$ and $\theta_2$.

Step 1:  For each element $e \in E$
           Cluster the top similar elements of $e$ from $S$ using
              average-link clustering.
           For each cluster discovered $c$ compute the following
              score: $|c| \times \text{avgsim}(c)$, where $|c|$ is the number of
              elements in $c$ and avgsim($c$) is the average
              pairwise similarity between elements in $c$.
           Store the highest-scoring cluster in a list $L$.

Step 2:  Sort the clusters in $L$ in descending order of their
         scores.

Step 3:  Let $C$ be a list of committees, initially empty.
         For each cluster $c \in L$ in sorted order
           Compute the centroid of $c$ by averaging the
              frequency vectors of its elements and computing
              the mutual information vector of the centroid in
              the same way as we did for individual elements.
           If $c$'s similarity to the centroid of each committee
              previously added to $C$ is below a threshold $\theta_1$, add
              $c$ to $C$.

Step 4:  If $C$ is empty, we are done and return $C$.

Step 5:  For each element $e \in E$
           If $e$'s similarity to every committee in $C$ is below
              threshold $\theta_2$, add $e$ to a list of residues $R$.

Step 6:  If $R$ is empty, we are done and return $C$.
         Otherwise, return the union of $C$ and the output of a
            recursive call to Phase II using the same input
            except replacing $E$ with $R$.

Output:  a list of committees.
```

Figure 1. Phase II of CBC.

In Step 1, the score reflects a preference for bigger and tighter clusters. Step 2 gives preference to higher quality clusters in Step 3, where a cluster is only kept if its similarity to all previously kept clusters is below a fixed threshold. In our experiments, we set $\theta_1 = 0.35$. Step 4 terminates the recursion if no committee is found in the previous step. The residue elements are identified in Step 5 and if no residues are found, the algorithm terminates; otherwise, we recursively apply the algorithm to the residue elements.

Each committee that is discovered in this phase defines one of the final output clusters of the algorithm.

4.3 Phase III: Assign elements to clusters

In Phase III, every element is assigned to the cluster containing the committee to which it is most similar. This phase resembles *K*-means in that every element is assigned to its closest centroid. Unlike *K*-means, the number of clusters is not fixed and the centroids do not change (i.e. when an element is added to a cluster, it is not added to the committee of the cluster).

5. EVALUATION METHODOLOGY

Many cluster evaluation schemes have been proposed. They generally fall under two categories:

- comparing cluster outputs with manually generated answer keys (hereon referred to as **classes**); and
- embedding the clusters in an application (e.g. information retrieval) and using its evaluation measure.

One approach considers the average entropy of the clusters, which measures the purity of the clusters [17]. However, maximum purity is trivially achieved when each element forms its own cluster.

Given a partitioned set of *n* elements, there are *n* × (*n* – 1) / 2 pairs of elements that are either in the same partition or not. The partition implies *n* × (*n* – 1) / 2 decisions. Another way to evaluate clusters is to compute the percentage of the decisions that are in agreement between the clusters and the classes [19]. This measure sometimes gives unintuitive results. Suppose the answer key consists of 20 equally sized classes with 1000 elements in each. Treating each element as its own cluster gets a misleadingly high score of 95%.

The evaluation of document clustering algorithms in information retrieval often uses the embedded approach [6]. Suppose we cluster the documents returned by a search engine. Assuming the user is able to pick the most relevant cluster, the performance of the clustering algorithm can be measured by the average precision of the chosen cluster. Under this scheme, only the best cluster matters.

The entropy and pairwise decision schemes each measure a specific property of clusters. However, these properties are not directly related to application-level goals of clustering. The information retrieval scheme is goal-oriented, however it measures only the quality of the best cluster. We propose an evaluation methodology that strikes a balance between generality and goal-orientation.

Like the entropy and pairwise decision schemes, we assume that there is an answer key that defines how the elements are supposed to be clustered. Let *C* be a set of clusters and *A* be the answer key. We define the editing distance, *dist*(*C*, *A*), as the number of operations required to transform *C* into *A*. We allow three editing operations:

- merge two clusters;
- move an element from one cluster to another; and
- copy an element from one cluster to another.

Let *B* be the baseline clustering where each element is its own cluster. We define the quality of cluster *C* as follows:

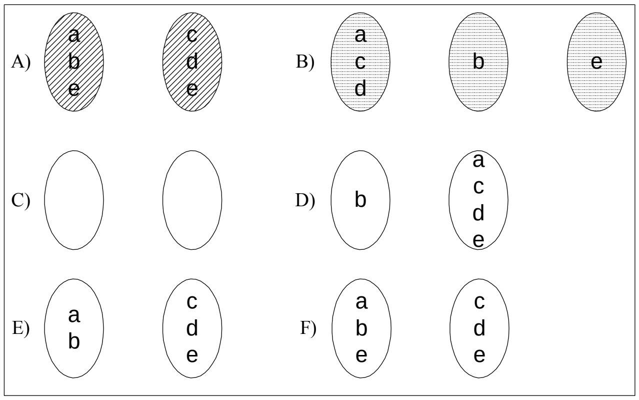

Figure 2. An example of applying the transformation rules to three clusters. A) The classes in the answer key; B) the clusters to be transformed; C) the sets used to reconstruct the classes (Rule 1); D) the sets after three merge operations (Step 2); E) the sets after one move operation (Step 3); F) the sets after one copy operation (Step 4).

$$1 - \frac{dist(C, A)}{dist(B, A)}$$

This measure can be interpreted as the percentage of savings from using the clustering result to construct the answer key versus constructing it from scratch (i.e. the baseline).

We make the assumption that each element belongs to exactly one cluster. The transformation procedure is as follows:

1. Suppose there are *m* classes in the answer key. We start with a list of *m* empty sets, each of which is labeled with a class in the answer key.
2. For each cluster, merge it with the set whose class has the largest number of elements in the cluster (a tie is broken arbitrarily).
3. If an element is in a set whose class is not the same as one of the element’s classes, move the element to a set where it belongs.
4. If an element belongs to more than one target class, copy the element to all sets corresponding to the target classes (except the one to which it already belongs).

dist(*C*, *A*) is the number of operations performed using the above transformation rules on *C*.

Figure 2 shows an example. In *D*) the cluster containing *e* could have been merged with either set (we arbitrarily chose the second). The total number of operations is 5.

6. EXPERIMENTAL RESULTS

In this section, we describe our test data and present an evaluation of our system. We compare CBC to the clustering algorithms presented in Section 2 and we provide a detailed analysis of *K*-means and Buckshot. We proceed by studying the effect of different clustering parameters on CBC.

Table 1. The number of classes in each test data set and the number of elements in their largest and smallest classes.

DATA SET	TOTAL DOCS	TOTAL CLASSES	LARGEST CLASS	SMALLEST CLASS
Reuters	2745	92	1045	1
20-news	18828	20	999	628

Table 2. Cluster quality (%) of several algorithms on the Reuters and 20-news data sets.

	REUTERS	20-NEWS
CBC	65.00	74.18
K-means	62.38	70.04
Buckshot	62.03	65.96
Bisecting *K*-means	60.80	58.52
Chameleon	58.67	n/a
Average-link	63.00	70.43
Complete-link	46.22	64.23
Single-link	31.53	5.30

6.1 Test Data

We conducted document-clustering experiments with two data sets: Reuters-21578 V1.2[1] and 20news-18828[2] (see Table 1). For the Reuters corpus, we selected documents that:

1. are assigned one or more topics;
2. have the attribute LEWISSPLIT="TEST"; and
3. have <BODY> and </BODY> tags.

There are 2745 such documents. The 20news-18828 data set contains 18828 newsgroup articles partitioned (nearly) evenly across 20 different newsgroups.

6.2 Cluster Evaluation

We clustered the data sets using CBC and the clustering algorithms of Section 2 and applied the evaluation methodology from the previous section. Table 2 shows the results. The columns are our editing distance based evaluation measure. CBC outperforms *K*-means with *K*=1000 by 4.14%. On the 20-news data set, our implementation of Chameleon was unable to complete in reasonable time. For the 20-news corpus, CBC spends the vast majority of the time finding the top similar documents (38 minutes) and computing the similarity between documents and committee centroids (119 minutes). The rest of the computation, which includes clustering the top-20 similar documents for every one of the 18828 documents and sorting the clusters, took less than 5 minutes. We used a Pentium III 750MHz processor and 1GB of memory.

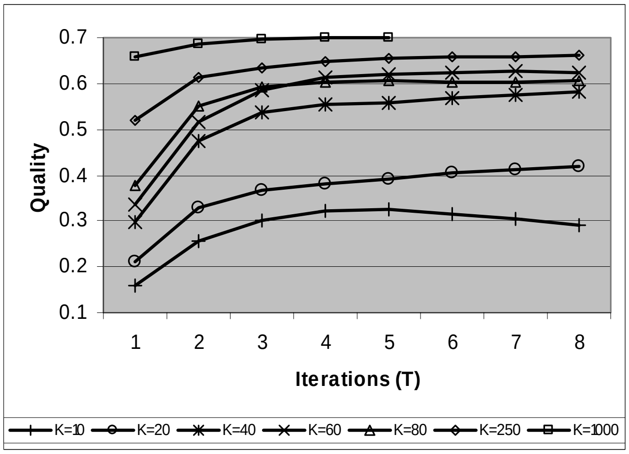

Figure 3. *K*-means cluster quality on the 20-news data set for different values of *K* plotted of over eight iterations.

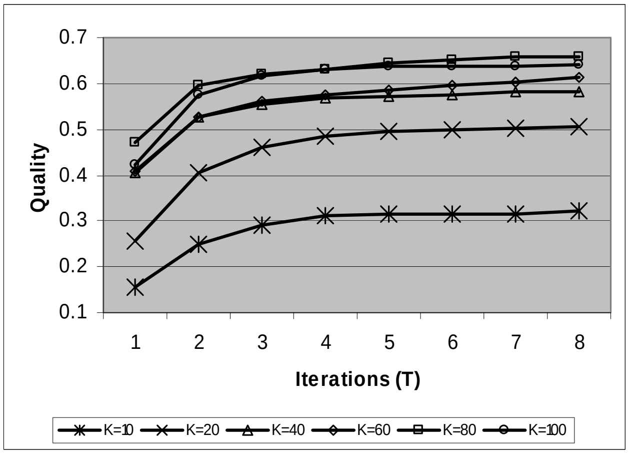

Figure 4. Buckshot cluster quality on the 20-news data set for different values of *K* plotted of over eight iterations.

6.3 *K*-means and Buckshot

Figure 3 and Figure 4 show the cluster quality of different *K*'s on the 20-news data set plotted over eight iterations of the *K*-means and Buckshot algorithms respectively. The cluster quality for *K*-means clearly increases as *K* reaches 1000 although the increase in quality slows down between *K*=60 and *K*=1000.

Buckshot has similar performance to *K*-means on the Reuters corpus; however it performs much worse on the 20-news corpus. This is because *K*-means performs well on this data set when *K* is large (e.g. *K*=1000) whereas Buckshot cannot have *K* higher than $\sqrt{18828} = 137$. On the Reuters corpus, the best clusters for *K*-means were obtained with *K* = 50, and Buckshot can have *K* as large as $\sqrt{2745} = 52$. However, as *K* approaches 52, Buckshot degenerates to the *K*-means algorithm, which explains why Buckshot has similar performance to *K*-means. Figure 5 compares the cluster quality between *K*-means and Buckshot for different values of *K* on the 20-news data set.

[1] http://www.research.att.com/~lewis/reuters21578.html

[2] http://www.ai.mit.edu/people/jrennie/20_newsgroups/

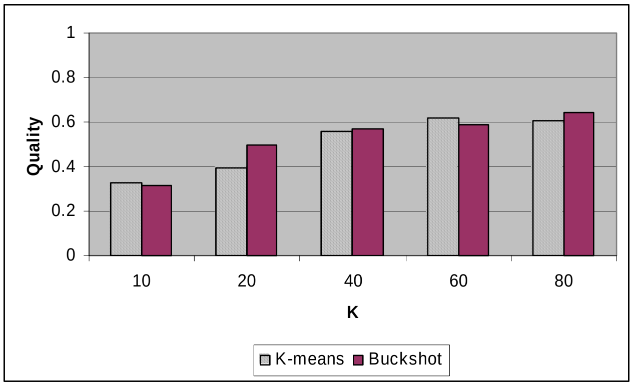

Figure 5. Comparison of cluster quality between *K*-means and Buckshot for different *K* on the 20-news data set.

Buckshot first applies average-link clustering to a random sample of $\sqrt{n}$ elements, where *n* is the number of elements to be clustered. The sample size counterbalances the quadratic running time of average-link to make Buckshot linear. We experimented with larger sample sizes to see if Buckshot performs better. For the 20-news data set, clustering 137 elements using average-link is very fast so we can afford to cluster a larger sample. Figure 6 illustrates the results for *K*=150 on the 20-news data set where *F* indicates the forced sample size and *F*=SQRT is the original Buckshot algorithm described in Section 2. Since $K > 137$, *F=SQRT* is just the *K*-means algorithm (we always sample at least *K* elements). Buckshot has better performance than *K*-means as long as the sample size is significantly bigger than *K*. All values of $F \geq 500$ converged after only two iterations while *F*=SQRT took four iterations to converge.

6.4 Clustering Parameters

We experimented with different clustering parameters. Below, we describe each parameter and their possible values:

1. **Vector Space Model** (described in Section 3):
   a) *MI* : the mutual-information model;
   b) *TF* : the term-frequency model;
   c) *TFIDF1* : the *tf-idf* model;
   d) *TFIDF2* : the *tf-idf* model using the following formula: $tf\text{-}idf = \sqrt{tf} \times \log idf$.

2. **Stemming**:
   a) `S-` : terms are not stemmed;
   b) `S+` : terms are stemmed using Porter's stemmer [14].

3. **Stop Words**:
   a) `W-` : no stop words are used as features;
   b) `W+` : all terms are used.

4. **Filtering**:
   a) `F-` : no term filtering is performed;
   b) `F+` : terms with *MI*<0.5 are deleted.

When filtering is on, the feature vectors become smaller and the similarity computations become much faster.

We refer to an experiment using a string where the first position corresponds to the *Stemming* parameter, the second position corresponds to the *Stop Words* parameter and the third position corresponds to the *Filtering* parameter. For example, experiment `S+W-F+` means that terms are stemmed, stop words are ignored, and filtering is performed. The *Vector Space Model* parameter will always be explicitly given.

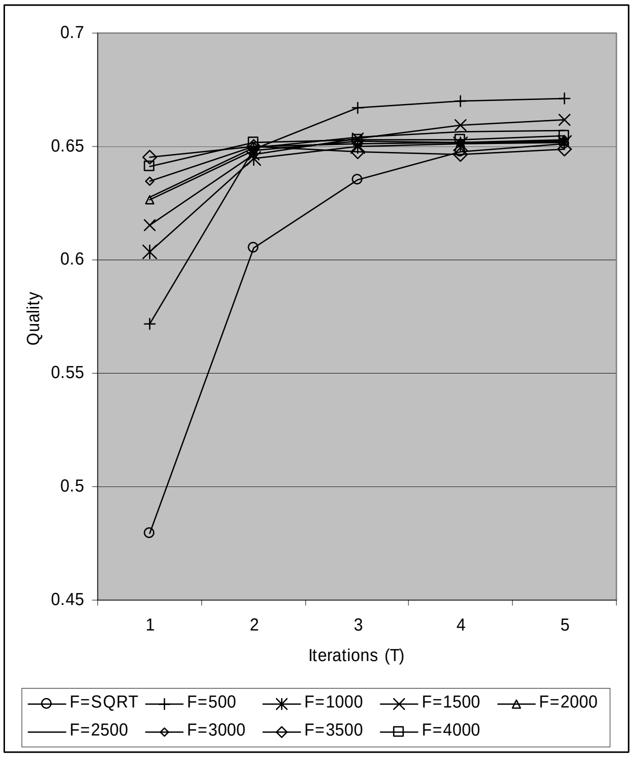

Figure 6. Buckshot cluster quality with *K*=150 for varying sample size (F) on the 20-news data set plotted over five iterations.

Figure 7 illustrates the quality of clusters generated by CBC on the Reuters corpus while varying the clustering parameters. Most document clustering systems use *TFIDF1* as their vector space model; however, the *MI* model outperforms each model including *TFIDF1*. Furthermore, varying the other parameters on the *MI* model makes no significant difference on cluster quality making *MI* more robust. *TF* performs the worse since terms with low discriminating power (e.g. *the*, *furthermore*) are not discounted. Although *TFIDF2* slightly outperforms *TFIDF1* (on experiment `S+W-F-`), it is clearly not as robust. Except for the *TF* model, stemming terms always produced better quality clusters.

7. EXAMPLE

We collected the titles and abstracts for the 46 papers presented at SIGIR-2001 and clustered them using CBC. For each paper, we used as part of its filename the session name in which it was presented at the conference and a number representing the order in which it appears in the proceedings. For example, Cat/017 refers to a paper that was presented in the *Categorization* session and that is the 17th paper in the proceedings. The results are shown in Table 3.

The features of many of the automatically generated clusters clearly correspond to SIGIR-2001 session topics (e.g. clusters 1,

Table 3. Output of CBC applied to the 46 papers presented at SIGIR-2001: the left column shows the clustered documents and the right column shows the top-7 features (stemmed terms) forming the cluster centroids.

#	*CLUSTER ELEMENTS*	*TOP-7 FEATURES* (STEMMED)
1	Cat/017, Cat/019, Lrn/037	text, featur, learn, categor, classif, approach, svm
2	MS/035, Eval/011, MS/036, Sys/006, Cat/018	threshold, score, term, base, distribut, optim, scheme
3	LM/015, LM/041, LM/016, CL/012, CL/013, CL/014	model, languag, translat, expans, estim, improv, framework
4	US/029, US/028, US/027, Sys/008, Sum/024, Eval/009, Lrn/038	user, result, use, imag, system, search, index
5	Web/030, Sys/007, MS/033, Eval/010	search, engin, page, web, link, best
6	Sum/002, Lrn/039, LM/042	stori, new, event, time, process, applic, content
7	Sum/003, Sum/004, Sum/025, LM/043, Sum/001	summar, term, text, summari, weight, scheme, automat
8	Lrn/040, Sys/005	level, space, framework, vector, comput, recent
9	QA/046, QA/044, QA/045, MS/034	answer, question, perform, task, passag, larg, give
10	RM/021, RM/022, RM/023, RM/020	queri, languag, similar, data, framework, seri
11	Web/032, Sum/026, Web/031	link, algorithm, method, hyperlink, web, analyz, identifi

Cat=Categorization, CL=CrossLingual, Eval=Evaluation, LM=LanguageModels, Lrn=Learning, MS=MetaSearch, QA=QuestionAnswering, RM=RetrievalModels, Sum=Summarization, Sys=Systems, US=UserStudies, Web=Web

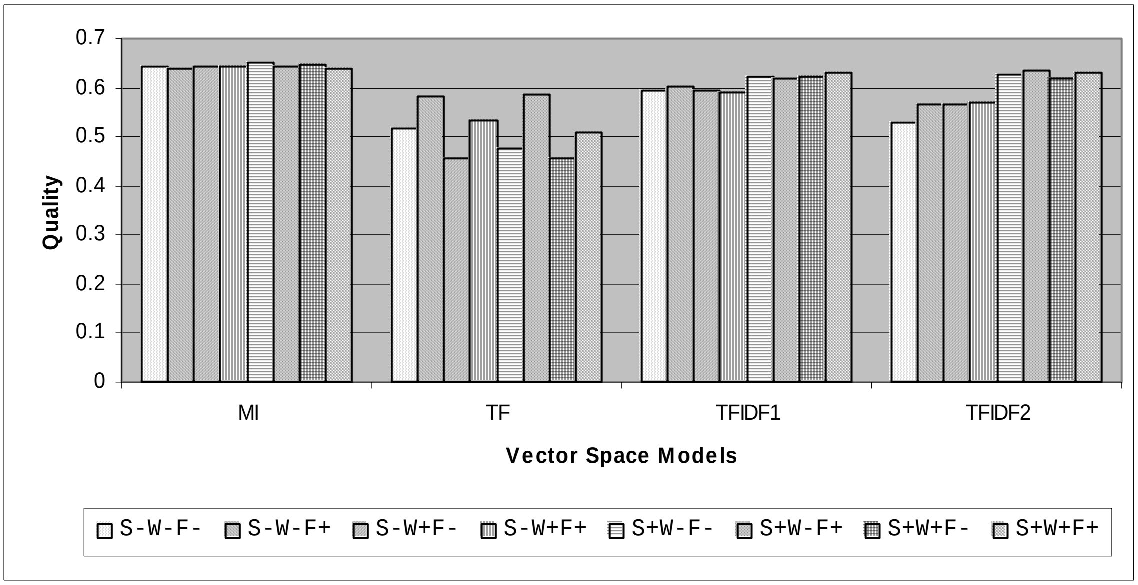

Figure 7. CBC evaluation of cluster quality when using different clustering parameters (Reuters corpus).

4, 7, 9, 10 and 11). Applying the evaluation methodology of Section 5 gives a score of 32.60%. This score is fairly low for the following reasons:

- Some documents could potentially belong to more than one session. For example, Lrn/037 was clustered in the categorization cluster #1 because it deals with learning and text categorization (it is titled “A meta-learning approach for text categorization”). Using the sessions as the answer key, Lrn/037 will be counted as incorrect.
- CBC generates clusters that do not correspond to any session topic. For example, all papers in Cluster #6 have news stories as their application domain and the papers in Cluster #5 all deal with search engines.

8. CONCLUSION

Document clustering is an important tool in information retrieval. We presented a clustering algorithm, CBC, which can handle a large number of documents, a large number of output clusters, and a large sparse feature space. It discovers clusters using well-scattered tight clusters called committees. In our experiments on document clustering, we showed that CBC outperforms several well-known hierarchical, partitional, and hybrid clustering algorithms in cluster quality. For example, in one experiment, CBC outperforms *K*-means by 4.14%.

Evaluating cluster quality has always been a difficult task. We presented a new evaluation methodology that is based on the editing distance between output clusters and manually constructed classes (the answer key). This evaluation measure is more intuitive and easier to interpret than previous evaluation measures.

CBC may be applied to other clustering tasks such as word clustering. Since many words have multiple senses, we can modify Phase III of CBC to allow an element to belong to multiple clusters. For an element *e*, we can find its most similar cluster and assign *e* to it. We can then remove those features from *e* that are shared by the centroid of the cluster. Then, we can recursively find *e*'s next most similar cluster and repeat the feature removal. This process continues until *e*'s similarity to its most similar cluster is below a threshold or when the total mutual information of all the residue features of *e* is below a fraction of the total mutual information of its original features. For a polysymous word, CBC can then potentially discover clusters that correspond to its senses. Preliminary experiments on clustering words using the TREC collection (3GB) and a proprietary collection (2GB) of grade school readings from Educational Testing Service gave the following automatically discovered word senses for the word *bass*:

```
(clarinet, saxophone, cello, trombone)
(Allied-Lyons, Grand Metropolitan, United Biscuits,
 Cadbury Schweppes)
(contralto, baritone, mezzo, soprano)
(Steinbach, Gallego, Felder, Uribe)
(halibut, mackerel, sea bass, whitefish)
(Kohlberg Kravis, Kohlberg, Bass Group, American
 Home)
```

and for the word *China*:

```
(Russia, China, Soviet Union, Japan)
(earthenware, pewter, terra cotta, porcelain)
```

The word senses are represented by four committee members of the cluster.

9. ACKNOWLEDGEMENTS

The authors wish to thank the reviewers for their helpful comments. This research was partly supported by Natural Sciences and Engineering Research Council of Canada grant OGP121338 and scholarship PGSB207797.

10. REFERENCES

[1] Buckley, C. and Lewit, A. F. 1985. Optimization of inverted vector searches. In *Proceedings of SIGIR-85*. pp. 97–110.

[2] Church, K. and Hanks, P. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of ACL-89*. pp. 76–83. Vancouver, Canada.

[3] Cutting, D. R.; Karger, D.; Pedersen, J.; and Tukey, J. W. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR-92*. pp. 318–329. Copenhagen, Denmark.

[4] Guha, S.; Rastogi, R.; and Kyuseok, S. 1999. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of ICDE'99*. pp. 512–521. Sydney, Australia.

[5] Han, J. and Kamber, M. 2001. *Data Mining – Concepts and Techniques*. Morgan Kaufmann.

[6] Hearst, M. A. and Pedersen, J. O. 1996. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR-96*. pp. 76–84. Zurich, Switzerland.

[7] Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data Clustering: A Review. *ACM Computing Surveys* 31(3):264–323.

[8] Jardine, N. and van Rijsbergen, C. J. 1971. The use of hierarchical clustering in information retrieval. *Information Storage and Retreival*, 7:217–240.

[9] Karypis, G.; Han, E.-H.; and Kumar, V. 1999. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer: Special Issue on Data Analysis and Mining* 32(8):68–75.

[10] Kaufmann, L. and Rousseeuw, P. J. 1987. Clustering by means of medoids. In Dodge, Y. (Ed.) *Statistical Data Analysis based on the L1 Norm*. pp. 405–416. Elsevier/North Holland, Amsterdam.

[11] King, B. 1967. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101.

[12] Koller, D. and Sahami, M. 1997. Hierarchically classifying documents using very few words. In Proceedings of ICML-97. pp. 170–176. Nashville, TN.

[13] McQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematics, Statistics and Probability*, 1:281-298.

[14] Porter, M. F. 1980. An algorithm for suffix stripping. In *Proceedings of SIGIR-80*. pp. 318–327.

[15] Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

[16] Sneath, P. H. A. and Sokal, R. R. 1973. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. Freeman. London, UK.

[17] Steinbach, M.; Karypis, G.; and Kumar, V. 2000. A comparison of document clustering techniques. *Technical Report #00-034*. Department of Computer Science and Engineering, University of Minnesota.

[18] van Rijsbergen, C. J. 1979. *Information Retrieval*, second edition. London: Buttersworth. Available at: http://www.dcs.gla.ac.uk/Keith/Preface.html

[19] Wagstaff, K. and Cardie, C. 2000. Clustering with instance-level constraints. In *Proceedings of ICML-2000*. pp. 1103–1110. Palo Alto, CA.