

IBM Research Report

Automated Generation of Model Cases for Help-Desk Applications

Sholom M. Weiss, Chidanand V. Apte
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598



Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Automated Generation of Model Cases for Help-Desk Applications

Sholom M. Weiss and Chidanand V. Apte

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, USA
sholom@us.ibm.com and apte@us.ibm.com

Abstract. Document databases may be ill-formed, containing redundant and poorly organized documents. For example, a database of customers' descriptions of problems with products and the vendor's descriptions of their resolution may contain many descriptions of the same problem. A highly desirable goal is to transform the database into a concise set of summarized reports, model cases, which in turn are more amenable to search and problem resolution without expert intervention. In this paper, we describe techniques that try to automate the procedures for reducing a database to its essential components. Our initial application is self-help for resolution of product problems. A lightweight document clustering method is described that operates in high dimensions, processes tens of thousands of documents and groups them into several thousand clusters. Techniques for summarization and exemplar selection are described to further refine the database contents. The method has been evaluated on a database of over 100,000 customer service problem reports that are reduced to 3,000 clusters and 5,000 exemplar documents. Preliminary results are promising and demonstrate efficient clustering performance with excellent group similarity measures, reducing the original database size by several orders of magnitude.

1 Introduction

An ill-formed document repository may consist of documents covering the same topics and documents composed of unfocused text. Ideally, we would like to reduce the size of this database by eliminating the redundant documents and summarizing the remaining documents. To remove the redundant documents, we will consider the use of high-dimensional clustering techniques. To cleanse the remaining documents, we consider both knowledge extraction techniques and document summarization methods. Automated procedures cannot be expected to perform these tasks perfectly. However, we can find real-world circumstances where imperfect results will still provide large benefits.

Let's look at these concepts by way of a help-desk example, where users submit problems or queries online to the vendor of a product. Each submission can be considered a document. By clustering the documents, the vendor can obtain an overview of the types of problems the customers are having, for example a computer vendor might discover that printer problems comprise a large percentage

of customer complaints. Typically, the number of clusters or categories number no more than a few hundred and often less than 100.

Not all users of a product report unique problems to the help-desk. It can be expected that most problem reports are repeat problems, with many users experiencing the same difficulty. Given enough users who report the same problem, a model-case report may be created. To reduce the number of documents in the database of problem reports, redundancies in the documents must be detected. Unlike the summary of problem types, many problems will be similar but still have distinctions that are critical. Thus, while the number of clusters needed to eliminate duplication of problem reports can be expected to be much smaller than the total number of problems reports, the number of clusters is necessarily relatively large, much larger than the 100 clusters needed for summarization of problem types.

Ideally, the clusters will contain documents that address the same problem and present a solution. For example, many customers report the same software problem, and they receive the same fix. Looking at the individual reports within a cluster, we may still see some variability in their quality. Some reports may be concise, almost directly decomposing into a problem statement and solution. Others, such as those in IBM's call centers for software problems, are almost complete transcripts of customer and service representative discussions. In these case, individual documents may include much text that that does not relate to the ultimate problem resolution. The central purpose of the database is simply to maintain records of a customer's interaction with a service representative. Our ultimate goal is to summarize a individual document or to extract the relevant sections from multiple documents, yielding a model-case report for the problem. If that goal is achieved, the potential for self-help by customers is greatly increased.

In this paper, we describe a machine learning approach to automatically generating model-case reports. Redundant documents are detected by high dimensional clustering. Summaries of clusters can be found either by (a) topic summarization of multiple documents [Radev *et al.*, 2000] or by (b) exemplar selection and excerpt extraction. Results will show that the process can greatly reduce the size of a database while maintaining much of its integrity. The process is not perfect, but need not be to demonstrate efficacy.

2 Document Clustering Techniques

The classical k-means technique [Hartigan and Wong, 1979] can be applied to document clustering. Its weaknesses are well known. The number of clusters k must be specified prior to application. The summary statistic is a mean of the values for each cluster. The individual members of the the cluster can have a high variance and the mean may not be a good summary of the cluster members. As the number of clusters grow, for example to thousands of clusters, classical k-means clustering becomes untenable [McCallum *et al.*, 2000].

More recent attention has been given to hierarchical agglomerative methods [Griffiths *et al.*, 1997]. The documents are recursively merged bottom up, yielding a decision tree of recursively partitioned clusters. The distance measures used to find similarity vary from single-link to more computationally expensive ones, but they are closely tied to nearest-neighbor distance. The algorithm works by recursively merging the single best pair of documents or clusters, making the computational costs prohibitive for document collections numbering in the tens of thousands.

To cluster very large numbers of documents, possibly with a large number of clusters, some compromises must be made to reduce the number of indexed words and the number of expected comparisons. In [Larsen and Aone, 1999], indexing of each document is reduced to the 25 highest scoring TF-IDF words (term frequency and inverse document frequency [Salton and Buckley, 1997], and then k-means is applied recursively, for k=9. While efficient, this approach has the classical weaknesses associated with k-means document clustering. A hierarchical technique that also works in steps with a small, fixed number of clusters is described in [Cutting *et al.*, 1992].

We will describe a lightweight procedure that operates efficiently in high dimensions and is effective in directly producing clusters that have objective similarity. Unlike k-means clustering, the number of clusters is dynamically determined, and similarity is based on nearest-neighbor distance, not mean feature distance. Thus, the document clustering method maintains the key advantage of hierarchical clustering techniques, their compatibility with information retrieval methods, yet performance does not rapidly degrade for large numbers of both documents and clusters.

3 Methods and Procedures

3.1 Data Preparation

Clustering algorithms process documents in a transformed state, where the documents are represented as a collection of terms or words. A vector representation is used: in the simplest format, each element of the vector is the presence or absence of a word. The same vector format is used for each document; the vector is a space taken over the complete set of words in all documents. Clearly, a single document has a sparse vector over the set of all words. Some processing may take place to stem words to their essential root and to transform the presence or absence of a word to a score, such as TF-IDF, that is a predictive distance measure. In addition weakly predictive words, stopwords, are removed. These same processes can be used to reduce indexing further by measuring for a document's vector only the top k-words in a document and setting all remaining vector entries to zero.

An alternative approach to selecting a subset of features for a document, described in [Weiss *et al.*, 2000], assumes that documents are carefully composed and have effective titles. Title words are always indexed along with the k most frequent words in the document and any human-assigned key words.

Not all words are of the same predictive value and many approaches have been tried to select a subset of words that are most predictive. The main concept is to reduce the number of overall words that are considered, which reduces the representational and computational tasks of the clustering algorithm. Reduced indexing can be effective in these goals when performed prior to clustering. The clustering algorithm accepts as input the transformed data, much like any information retrieval system, and works with a vector representation that is a transformation of the original documents.

3.2 Clustering Methods

Our method uses a reduced indexing view of the original documents, where only the k best keywords of each document are indexed. That reduces a document's vector size and the computation time for distance measures for a clustering method. Our procedure for clustering is specified in two parts (a) compute k most similar documents (typically the top 10) for each document in the collection and (b) group the documents into clusters using these similarity scores. To be fully efficient, both procedures must be computationally efficient. Finding and scoring the k most similar documents for each document will be specified as a mathematical algorithm that processes fixed scalar vectors. The procedure is simple, a repetitive series of loops that accesses a fixed portion of memory, leading to efficient computation. The second procedure uses the scores for the k most similar documents in clustering the document. Unlike the other algorithms described earlier, the second clustering step does not perform a "best-match first-out" merging. It merges documents and clusters based on a "first-in first-out" basis.

Table 3.2 describes the data structures needed to process the algorithms. Each of these lists can be represented as a simple linear vector. Table 3.2 describes the steps for the computation of the k most similar documents, typically the top 10, for each document in the collection. Similarity or distance is measured by a simple additive count of words found in both documents that are compared plus their inverse document frequency. This differs from the standard TF-IDF formula in that term frequency is measured in binary terms, i.e. 0 or 1 for presence or absence. In addition the values are not normalized, just the sum is used. In a comparative study, we show that TF-IDF has slightly stronger predictive value, but the simpler function has numerous advantages in terms of interpretability, simple additive computation, and elimination of storage of term frequencies. The steps in Table 3.2 can readily be modified to use TF-IDF scoring.

The remaining task is to group the documents into clusters using these similarity scores. We describe a single pass algorithm for clustering, with at most $k*n$ comparisons of similarity, where n is the number of documents.

For each document D_i , the scoring algorithm produces a set of k documents, $\{D_j\}$, where j varies from 1 to k . Given the scores of the top- k matches of each document D_i , Table 3.2 describes the actions that may be taken for each matched pair during cluster formation. Documents are examined in a pairwise fashion

doclist: The words (terms) in each document. A series of numbers; documents are separated by zeros. *example:* Sequence = 10 44 98 0 24 ... The first document has words 10, 44 and 98. The second document has words 24...

wordlist: The documents in which a word is found. A series of consecutive numbers pointing to specific document numbers.

word(c): A pointer to wordlist indicates the starting location of the documents for word c. To process all documents for word c, access word(c) through word(c+1)-1. *example:* word(1)=1, word(2)=4; wordlist={18 22 64 16 ...} Word 1 appears in the documents listed in locations 1, 2, and 3 in wordlist. The documents are 18, 22, and 64.

pv(c): predictive values of word c = 1+idf, where idf is 1/(number of documents where word c appears)

Table 1. *Definitions for Top-k Scoring Algorithm*

1. Get the next document's words (from doclist), and set all document scores to zero.
2. Get the next word, w, for current document. If no words remain, store the k documents with the highest scores and continue with step (i).
3. For all documents having word w (from wordlist), add to their scores and continue with step (ii).

Table 2. *Steps for Top-k Scoring Algorithm*

proceeding with the first document and its top-k matches. Matches below a preset minimum score threshold are ignored. Clusters are formed by the document pairs not yet in clusters. Clusters are merged when the matched pair appear in separate clusters. As we shall see in Section 5, not allowing merging yields a very large number clusters whose members are highly similar. The single setting of the minimum score has a strong effect on the number of clusters; a high value produces a relatively large number of clusters and a zero value produces a relatively small number of clusters. Similarly, a high minimum score may leave some documents unclustered, while a low value clusters all documents. As an alternative to merging, it may be preferable to repeat the same document in multiple clusters. We do not report results on this form of duplication, typically done for smaller numbers of documents, but the procedure provides an option for duplicating documents across clusters.

3.3 Measures for Evaluation of Clustering Results

How can we objectively evaluate clustering performance? Very often, the objective measure is related to the clustering technique. For example, k-means clustering can measure overall distance from the mean. Techniques that are based on nearest neighbor distance, such as most information retrieval techniques, can measure distance from the nearest neighbor or the average distance from other cluster members.

1. If score for D_i and D_j is less than Minimum Score, next pair.
2. If D_i and D_j are already in the same cluster, next pair.
3. If D_i is in a cluster and D_j isn't, add D_j to the D_i cluster, next pair.
4. Cluster Merge Step: if both D_i and D_j are in separate clusters:
 - (a) If action plan is "no merging", next pair.
 - (b) If action plan is "repeat documents", repeat D_j in all the D_i clusters, next pair.
 - (c) Merge the D_i cluster with D_j cluster, next pair.

Table 3. *Actions for Clustering Document Pairs*

For our clustering algorithm, distance is measured mostly in terms of counts of words present in documents. A natural measure of cluster performance is the average number of indexed words per cluster, i.e. the local dictionary size. Analogous measures of cluster "cohesion," that count the number common words among documents in a cluster, have been used to evaluate performance[Zamir *et al.*, 1997]. The average is computed by weighing the number of documents in the cluster as in equation 1, where N is the total number number of documents, m is the number of clusters, $Size_k$ is the number of documents in the k -th cluster, and $LDict_k$ is the number of indexed words in the k -th cluster.

$$AverageDictionarySize = \sum_{k=1}^m \frac{Size_k}{N} \cdot LDict_k \quad (1)$$

Results of clustering are compared to documents randomly assigned to the same size clusters. Clearly, the average dictionary size for computed clusters should be much smaller than those for randomly assigned clusters of the same number of documents.

4 Summarization and Excerpt Extraction

The goal of document clustering is to separate the documents into groups of similar documents. Clustering does not reduce the number of documents or the size of the document repository. The individual documents remain same. Here are two basic approaches to eliminating redundancy and reducing the size of the database.

- Select exemplars from each cluster.
- Produce a summary document for each cluster.

If the documents in a cluster are redundant, for example each describes the same self-help problem and solution, then selecting one document from the cluster can be sufficient to describe the cluster. Because the clustering procedure is imperfect and some clusters may be large, it may be safer to select more than one exemplar documents to represent the cluster, i.e. the problem-solution pair.

An alternative to the exemplar summarization technique is topic summarization [Radev *et al.*, 2000]. Each cluster contains documents for the same topic. Unlike single document summarization, summarizing documents on a common topic is sample-based and explores common patterns across many documents.

4.1 Critical Section Extraction

We have described a method for document reduction that is completely automated. However, its success depends on the quality and clarity of the original documents. It is best, especially for exemplar-based summarization, to have the original documents stripped to their bare essentials, for self-help, stripped to the problem-solution pairs. One can readily envision many real-world scenarios where the documents are poorly structured. Consider the following possibilities for our self-help example:

- The user composed the problem statement and a customer representative writes a solution. Because a problem-solution model is expected, and the discussants are asked to compose their thoughts in writing, the resultant document tends towards clarity and conciseness.
- The user communicates by phone to a call center and the representative creates a real-time approximate transcript of the their dialogue. This is the actual situation at IBM’s call centers, where thousands of these documents are created by thousands of service representatives each day. Many documents are rambling with extraneous text.

Cnum	AveSize	RndRatio	Unclust %	MinScore	Merge
49	1027.3	1.4	1.5	1	yes
86	579.6	1.4	2.5	2	yes
410	105.5	1.5	16.2	3	yes
3250	15.5	1.8	1.5	1	no
3346	14.9	1.8	2.5	2	no
3789	11.4	1.9	16.2	3	no

Table 4. *Results for Clustering Help-Desk Problems*

For the first possibility, little additional preparation is needed. For the second possibility, we need additional effort for automated procedures in extracting the critical sections. Knowledge-based models can be very helpful. For example, we know that the problem statement is typically at the beginning of the document and the solution is at the end. Moreover, the service representatives are told to prefix critical sections with key words like "action taken." Far more helpful, and far more powerful for a self-help document, would be for the customer representative to write a one or two line summary of the solution. This takes a little

extra time, reduces the time available for a single representative to take more calls, and is not needed when the main purpose of the document is to maintain a record of the customer's problem. If the expectation is for redundancy, i.e. repeat problems, then concise and consistent summarization by the author's warrants the individual effort because of the long-term gains in more quickly matching new problems to the stored documents in the database.

4.2 Exemplar Selection

The same measure of evaluation can be used to find exemplar documents for a cluster. The local dictionary of a document cluster can be used as a virtual document that is matched to the members of the cluster. The top-k matched documents can be considered a ranked list of exemplar documents for the cluster.

Selecting exemplar documents from a cluster is a form of summary of the cluster. The technique for selecting the exemplars is based on matching the cluster's dictionary of words to its constituent documents. The words themselves can provide another mode of summary for a cluster. The highest frequency words in the local dictionary of a cluster often can distinguish a cluster from others. If only a few words are extracted, they may be considered a label for the cluster.

5 Results

To evaluate the performance of the clustering algorithms, we obtained 51,110 documents, taken from reports from customers having IBM AS/400 computer systems. These documents were constructed in real-time by customer service representatives who record their phone dialog with customers encountering problems with their systems.

The documents were indexed with a total of 21,682 words in a global dictionary computed from all the documents. Table 4.1 summarizes the results for clustering the document collection in terms of the number of clusters, the average cluster size, the ratio of the local dictionary size to random assignment, the percentage of unclustered documents, the minimum score for matching document pairs, and whether merging was used. The first row in the table indicates that 49 clusters were found with an average size of 1027 documents. A random cluster's dictionary was on average 1.4 times larger than the generated cluster; and 1.5% of the documents were not clustered. These results were obtained by using a minimum score of 1 and cluster merging was allowed. All results are for finding the top-10 document matches.

A single clustering run, one row in Table 4.1 currently takes 15 minutes on a 375 MHz RS6000 running AIX. The code is written in Java.

Exemplar documents were selected for each of the 3250 clusters found in the fourth entry of the table. For some large clusters, two or three exemplars were selected for a total of 5,000 exemplar documents. Using the same scoring scheme, each of the exemplars was matched to the original 51,110 documents. 98.5% of the documents matched at least one of the exemplars, having at least one

indexed word in common. 80% of the documents matched an exemplar of their assigned cluster, rather than an exemplar of an alternative cluster.

6 Discussion

Our techniques for detecting and removing redundant documents from the repository use high-dimensional clustering. By some empirical measures, we can demonstrate that the process is effective in achieving our stated objectives. The process is efficient in high dimensions, both for large document collections and for large numbers of clusters. No compromises are made to partition the clustering process into smaller sub-problems. All documents are clustered in one stage. In the self-help application, it is important to remove duplication, while still maintaining a large number of exemplar documents. The help-desk clusters have strong similarity for their documents, suggesting that they can be readily summarized by a one or two documents. For the largest number of clusters, dictionary size is nearly half that for random document assignment, far better than for smaller number of clusters.

While our methods have many desirable properties for operating in high dimensions, we have not demonstrated that the lightweight algorithm is optimal in any sense. Moreover, we have not given any empirical comparisons to other clustering methods that show its superiority. That awaits further experimentation. For our higher goal of extracting the critical segments of documents in an ill-formed repository, the superiority of the clustering algorithm is not the only component of evaluation. We know that the extracted documents will be accessed by a search engine that will give multiple answers in response to a query or to a document matcher that matches problem descriptions to stored documents. To be effective, we need a high-dimensional clustering method that operates in reasonable times, but we need not have perfect clustering to eliminate redundancy.

For topic summarization [Radev *et al.*, 2000], researchers have reported some good preliminary results by k-means clustering sentences or paragraphs for the pooled documents, and selecting those sentences that are most similar to a cluster's mean vector. We have not yet tried this summarization technique. It remains a promising, but more complex approach. The exemplar approach keeps documents intact. Summarization by topic merges excerpts of many documents, and is therefore more susceptible to mistakes in the extraction of the excerpts. However, the exemplar approach is more dependent on starting with cleansed documents that contain the critical sections of the document, for example the problem statement and solution pair. The topic summarization technique has the potential to find those sections because they will appear in many samples, while discarding those sections are unique to single documents.

References

[Cutting *et al.*, 1992] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/Gather: a Cluster-based Approach to Browsing Large Document collections. In

- Proceedings of the 15th ACM SIGIR*. ACM, 1992.
- [Griffiths *et al.*, 1997] A. Griffiths, H. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. In P. Sparck-Jones, K. and Willet, editor, *Readings in Information Retrieval*, pages 365–373. Morgan Kaufmann, 1997.
- [Hartigan and Wong, 1979] J. Hartigan and M Wong. A k-means clustering algorithm. *Applied Statistics*, 1979.
- [Larsen and Aone, 1999] B. Larsen and C. Aone. Fast and Effective Text Mining Using Linear-time Document Clustering. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 16–22. ACM, 1999.
- [McCallum *et al.*, 2000] A. McCallum, K Nigam, and L. Ungar. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proceedings Sixth (ACM) International Conference on Knowledge Discovery and Data Mining*, 2000.
- [Radev *et al.*, 2000] D. Radev, H. Jing, and M. Budzikowska. Summarization of Multiple Documents: Clustering, Sentence Extraction, and Evaluation. In *Proceedings ANLP/NAACL Workshop on Summarization*, 2000.
- [Salton and Buckley, 1997] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In P. Sparck-Jones, K. and Willet, editor, *Readings in Information Retrieval*, pages 323–328. Morgan Kaufmann, 1997.
- [Weiss *et al.*, 2000] S. Weiss, B. White, C. Apté, and F. Damerou. Lightweight document matching for help-desk applications. *IEEE Intelligent Systems*, page in press, 2000.
- [Zamir *et al.*, 1997] O. Zamir, O. Etzioni, O. Madani, and R. Karp. Fast and Intuitive Clustering of Web Documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*. Morgan Kaufmann, 1997.