**Real-Time Document Cluster Analysis for Dynamic Data Sets**

Mark T. Elmore, Joel W. Reed, Thomas E. Potok, and Robert M. Patton
Oak Ridge National Laboratory*
Computational Sciences and Engineering Division
Post Office Box 2008, Mail Stop 6364
Oak Ridge, Tennessee 37831-6364

## Abstract

One of the most challenging analysis problems in the data mining and information retrieval domains is organizing large amounts of information. In this paper, we present a fast agglomerative clustering technique used in the Virtual Information Processing Agent Research (VIPAR) project at the Oak Ridge National Laboratory. This approach extends the Vector Space Model (VSM) to provide near real-time clustering of a moderately large and dynamic set of text documents. In the traditional VSM, each document added or removed from the document set requires a computationally expensive set of operations to be performed before analysis can resume. While this prior calculation of all of the VSM values is feasible in some problem domains, it is computationally prohibitive for near real-time operation with large, dynamic sets of documents.

We present a method to quickly and accurately update the VSM values in an environment where articles are being continuously added and removed. We conducted a series of experiments and based on the results, we implemented a strategy that provides dynamic updates to the VSM values while preserving high accuracy. This approach allows documents to be quickly added and removed from the document set, providing accurate agglomerative clustering and enabling real-time document analysis.

## Introduction

One of the most challenging analysis problems in the data mining and information retrieval domains is organizing large amounts of information. One approach to this problem is to cluster information based on the content of a collection of documents. One widely used technique is to represent documents as vectors in a Vector Space Model (VSM), compare those documents in a dissimilarity matrix, and use agglomerative clustering to represent the document comparisons in a dendogram. The Virtual Information Processing Agent Research (VIPAR) project at the Oak Ridge National Laboratory makes significant use of this clustering technique with the goal of enabling information from a number of Internet media sources to be integrated, then rapidly searched, clustered, analyzed, and visually presented to an analyst for improved decision making.

In VIPAR, thousands of articles from Internet newspapers come streaming into the VIPAR system throughout the day in an asynchronous manner. These articles must be immediately placed into article clusters, and in near real-time, provided to analysts using VIPAR. However, these requirements of dynamic datasets and real-time responsiveness were contradictory. While the VSM-based agglomerative clustering approach has demonstrated value in organizing

documents, its substantial computational complexity requires that the entire document set must be available for calculation before analysis can begin. This means that every time a new document is added to the system, the system must pause while recalculations are made for the entire collection of documents. Due to the substantial computational complexity of the recalculation, this pause may also be substantial and thus conflicting with the real-time responsiveness required by VIPAR.

To resolve this conflict, we developed an approach that allows articles to be dynamically added to, and deleted from, the clustering system while provide real-time responsiveness. We conducted a series of experiments to determine how changes to the document set would affect the VSM vectors, and hence the dissimilarity values used for agglomerative clustering. Based on these experiments, we have developed a method to dynamically update the vectors and dissimilarity matrix while maintaining sufficient fidelity for accurate agglomerative clustering.

**Background**

Clustering a relatively large volume of up-to-date newspaper articles is a challenging problem with no simple solutions. In this environment, the document set is continuously changing as new articles arrive and out of date articles are removed. The approach that we used for organizing the newspaper information is to cluster them based on their content. One of the most widely used clustering techniques for text document is agglomerative clustering.[1] To apply agglomerative clustering algorithms, a proximity measure between each pair of documents in the document set is needed. We chose the vector space model (VSM) to determine this proximity measure, which is one of the most widely used document representation approaches, and well suited for this application. Using this method, each document is represented as a vector in hyperspace, with each unique word in a collection of documents representing a dimension in the hyperspace. This document vector is based on the frequency of the terms used within a document (local frequency) and on the frequency of the terms used in the entire set of documents (global frequency).

Term Frequency, Inverse Document Frequency (TF-IDF) is a well established mechanism for weighting the value of terms in a document for building a document's vector. A term's frequency (TF) within a document (its local frequency) is a factor in determining the importance of the term in the document; the more often a term appears in a document, the more important the term is considered to be. Luhn[2] used this as a foundation for his early work in Automated Information Retrieval. However, as Zipf[3] previously pointed out, a term that appears in many documents in a collection becomes less important as a discriminator between documents. To balance this, the inverse of the term's frequency (IDF) within the entire document set (its global frequency) is factored in; the more often a term appears across the document set, the less important the term is considered to be in discriminating between documents. (An extreme case of this would be stop words, words that appear so frequently they have no discriminating value.) Salton[4] experimented with variations on the TF-IDF theme. Sparck Jones[5,6] compared different approaches to term weighting, as did Salton.[7] A number of variations on TF-IDF and of applications using TF-IDF have been built upon these seminal works.

Once document vectors have been built using TF-IDF, each document's vector is compared to every other document's vector to generate proximity measures. For each comparison, the

dissimilarity between the two vectors is recorded in a dissimilarity matrix. An agglomerative clustering algorithm is then applied to the dissimilarity matrix. In the document clustering domain, these algorithms are generally divided into two different families: hierarchical algorithms and partitional algorithms.[1,8]

Most of the current clustering literature reports that hierarchical clustering has better text document grouping performance compared to the partitional algorithms,[9,10] although there are reports of better results using hybrid hierarchical and partitional methods.[8,11] Hierarchical algorithms are better suited for use in the VIPAR system because the number of clusters to produce does not have to be chosen beforehand, and a dendogram is built during the clustering process. This dendogram is very helpful in visualizing the relationships among documents within a sub-cluster and therefore is very useful in VIPAR. However, these algorithms do have some features that make them difficult to use in the VIPAR environment. One difficulty is that the agglomerative clustering process is computationally complex, but the main difficulty is in dynamically updating the proximity measures needed for clustering. Typically, the matrix holding these values must be completely recalculated every time the document set changes. With the real-time demands placed on the system, it would be necessary to overcome this problem to be able to use an agglomerative clustering algorithm in VIPAR.

*A Typical Document Clustering System*

In order to more clearly illustrate the dynamic data clustering advancements of the VIPAR system, an overview of a typical document clustering system is in order. Although there are many variants, typical document clustering systems have many steps in common. The first step when adding a document to the document set is often the removal of all stop words. These are words that are common in speech, but carry little meaning, such as the words "the" or "and." Second, a stemming algorithm such as Porter's[12] is often applied to reduce the dimensionality of the vector space. For example, the words "walk," "walks," "walked," "walking," etc. would all be stemmed to the word "walk." The remaining words may then be counted to determine the frequency of each word within a given document (its local frequency) and determine the frequency of each word over the entire set of documents (its global frequency). These frequency counts are recorded in the local and global document frequency tables. The local document frequency table contains an entry for each document that records the frequency of each term in that document. The global frequency count table contains frequency counts for how often each unique term appears in the entire document set. From these local and global frequencies a document-term weighting is calculated by a TF-IDF function such as:[8]

$$Weight_{dt} = LF_{dt}\left(1 + \sum_{\forall d} \frac{LF_{dt}/GF_t * \log_2(LF_{dt}/GF_t)}{\log_2 n}\right)$$

Where *LF* is the local frequency for term *t* in document *d*, *GF* is the global frequency for term *t*, and *n* is the total number documents in the set.

Since each term in the document has a weight associated with it, the document can be represented by a vector composed of its term weights. The dimensionality of this vector space is equal to the quantity of unique terms represented in the document collection. One of several measures may be used to determine the dissimilarity between the vectors, (e.g.: dot product

between a pair of document vectors, Cartesian distance between the vector endpoints, etc.), and a dissimilarity matrix is built containing these measures between each document and every other document. For example, each cell of the dissimilarity matrix may be built by subtracting the dot product of a pair of document vectors from 1. Whenever a new document is added to a document set, a new row is added to the dissimilarity matrix. This row defines the dissimilarity between the new document and every other document that is currently in the document set.

Further, and of great importance, for each new document added to a document set, every document vector and hence every cell in the dissimilarity matrix must be recalculated. This is because the dimensionality of the vector space is based on the global terms. The weights that make up each document vector are based, in part, upon these global frequency counts. Any change in the document set changes the global frequency counts and thus, all of the weights within the vector space.

To produce the necessary document groupings and a dendogram that displays those groupings, we chose to use an agglomerative clustering algorithm. This method initially treats each document as a cluster. Among all cluster pairs, the method then locates the most similar pair of clusters using the dissimilarity matrix, and agglomerates this pair of clusters into a single cluster. The dissimilarity matrix is then updated to reflect the merged clusters using Ward's Method:[13]

$$D_{MC} = \left[ \frac{((A_n + C_n) * D_{AD} + (B_n + C_n) * D_{BD} - C_n * D_{AB}}{A_n + B_n + C_n} \right] \forall C$$

Where $D$ represents the dissimilarity measure between two documents, $M$ is the new cluster built when clusters $A$ and $B$ are merged, and where $C$ represents the cluster whose dissimilarity is being updated. Also, $A_n$ and $B_n$ are the number of documents that make up the clusters being merged to make cluster $M$, and $C_n$ is the number of documents that make up the cluster being updated. This merging process is repeated until all of the documents are in a single cluster

Depending on the size of the document set, the algorithms used, and the processing power available, the recalculation of the document vectors and the dissimilarity matrix may take minutes or hours (or days!). This is a significant roadblock to meeting an analyst's need for an agile, real-time system that allows an analyst to continue working while new documents are continually added into the system. How to circumvent this roadblock was the challenge before us.

**Approach**

In VIPAR, we needed something beyond the capabilities of a typical document clustering system. While much has been done in document clustering, the ability process dynamic streams of data in real-time is apparently an open issue. After much searching of the literature, we could find no applicable solutions, so we developed a new approach to the problem and devised a series of experiments to test its utility. The overall process is to adapt the typical document clustering system described above to create a vector representation of the newspaper articles that can be dynamically updated. From this, the dissimilarity matrix is updated with lower accuracy but with sufficient accuracy for the agglomerative clustering. When the users of the system

make a clustering request, this matrix is then used to agglomerate the requested subset of articles and generate a dendogram, which is displayed for the user.

To avoid recalculating all of the vectors every time a new document is added or deleted, we proposed an experiment to recalculate only a portion of the vectors and the cells within the dissimilarity matrix, and compare the results to a fully recalculating system. Although the fully recalculating system would be prohibitively slow for a real-time updating system, it would provide a benchmark to compare with the proposed real-time system. If the experimental results showed minimal deviation between the results of the two systems, it would show that a real-time system for a constantly changing document set was a viable solution for the analysts' real-time needs.

The novelty of our approach is to create a list of the matrix cells which is ordered by when they were last updated. Using this list, each time a new document is added to the document set, the oldest 5 percent of the matrix is updated. In other words, each time a document is added to the document set, the pairs of document vectors corresponding to least recently updated 5 percent of the matrix cells are recalculated, and then those matrix cells are updated using the new vectors. Documents being removed from the system are handled in a very similar manner. This allows documents to be quickly added to the system as they stream in, and removed from the system as they are no longer needed.

**Results**

The VIPAR system is operational at a number of organizations. Each organization has different information feeds. For this experiment, we gathered information from thirteen different Internet newspaper sites. These newspapers are:

1. Asahi Shimbun
2. Asia Times
3. BBC
4. Japan Times Online
5. Japan Update
6. Korea Times
7. Manila Times
8. Pacific Islands Report
9. Sydney Morning Herald
10. Taipei Times
11. The Hindu
12. The Star
13. Times of India

Each newspaper has an information agent[14] associated with it that sends new articles to the VIPAR system on an hourly basis. The information agents are not synchronized, so news articles can enter the VIPAR system at any time. Typically, averages of 20 to 30 new articles arrive every hour depending on how rapidly the news is changing, although this number can be much higher during system startup. The news articles are kept within the VIPAR system for approximately 48 hours, and then are discarded from the system.

We first look at the performance increases associated with this new approach. Using the typical approach of fully recalculating all of the document vectors every time an article is added or removed is very time consuming. This causes the clustering process to lag far behind the document gathering process. Also, it is important to note that any requests by users for document

grouping only causes more lag because the dissimilarity matrix needs to be locked during part of the clustering process.

Conversely, dynamically updating the vectors in the VSM resulted in minimal lag of the clustering process. Figure 1 shows how many documents have been downloaded but not yet incorporated into the clustering system. The number of unclustered articles is shown on the vertical axis versus time on the horizontal axis for both for the fully-recalculating clustering method (line with open boxes) and for the dynamic clustering method (line with solid diamonds). This data was gathered during the first 10 minutes of a typical VIPAR system startup and no clustering requests were processed during this time. Clearly, the full update method lags far behind and then begins to catch up to the download process, while the dynamic update method is able to easily keep up.

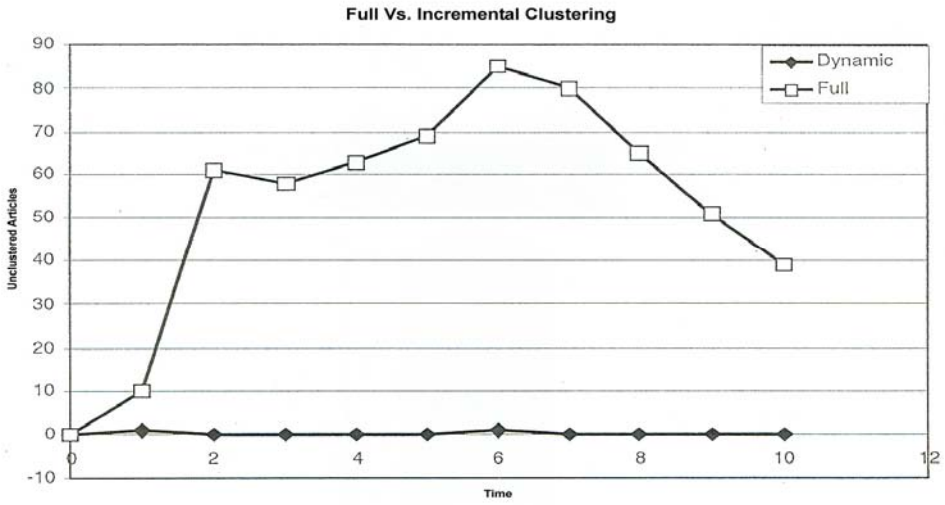


**Figure 1. A comparison of the full clustering methods versus the dynamic clustering method.**

Next we look at how the clustering results from the dynamic approach compared with the full update approach. Figure 2 shows a screen shot results of clustering a small document set, six articles, using our dynamic clustering approach. This cluster diagram is a dendogram similar in appearance to the Phylip Tree[15] visualizations for evolutionary trees. Here, the dendogram is used for visualization of agglomerative hierarchical clustering. The leaves of the tree represent each article while the links between the nodes represents relationships. In general, the closer two leaves are, the more similar the articles. When links from two leaves share a vertex, then these articles are the most closely related in the set of articles. The longer the links between articles, the greater the dissimilarity is between the articles. The clustering of these produced three distinct groups of articles.

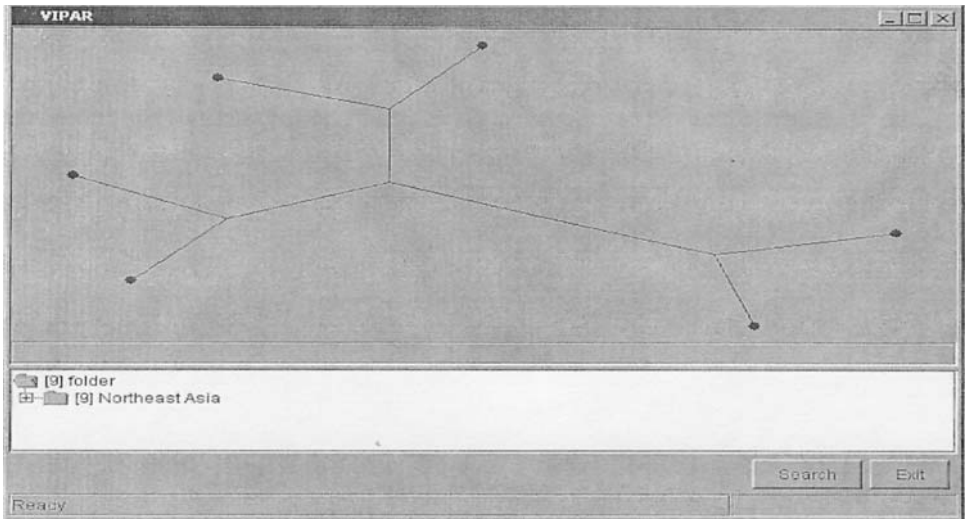


**Figure 2. Small scale dynamic clustering.**

Figure 3 shows the same articles, but clustered using the slower full-clustering approach. There appears to be no difference between Figure 2 and 3. This is most likely due to the fact that this small set of articles does not have a significant difference in the words used in each article. Therefore, there is no benefit from full clustering versus dynamic clustering.

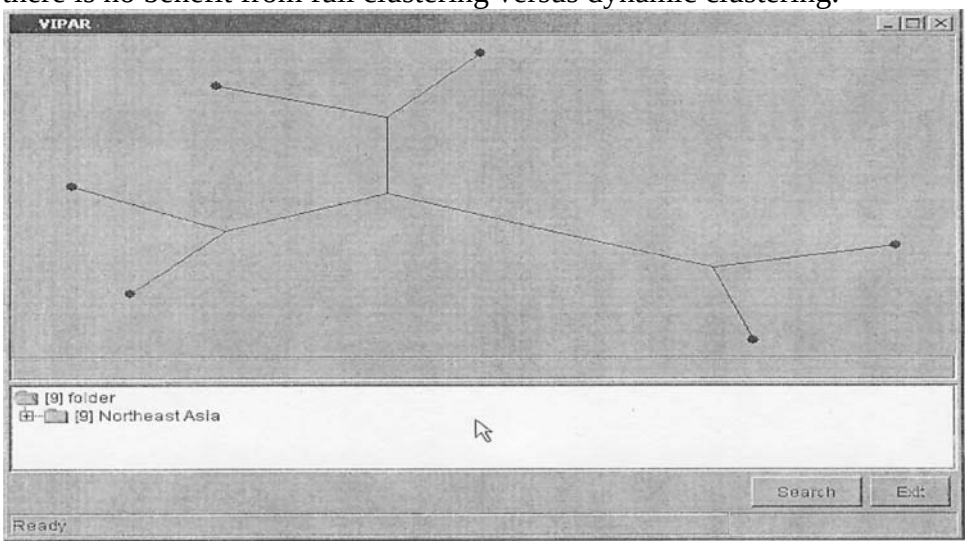


**Figure 3. Small scale full clustering.**

Figure 4 shows the dynamic clustering of hundreds of articles, forming several sub-clusters. The cluster shown in Figure 5 represents a full clustering of this same set of documents. From a qualitative analysis, the clusters represented in these two figures differ only slightly. Individual articles may minimally change position, but the overall sub-clusters remain much the same. It appears that from an analyst's viewpoint, there is little discernable distinction between the figures.

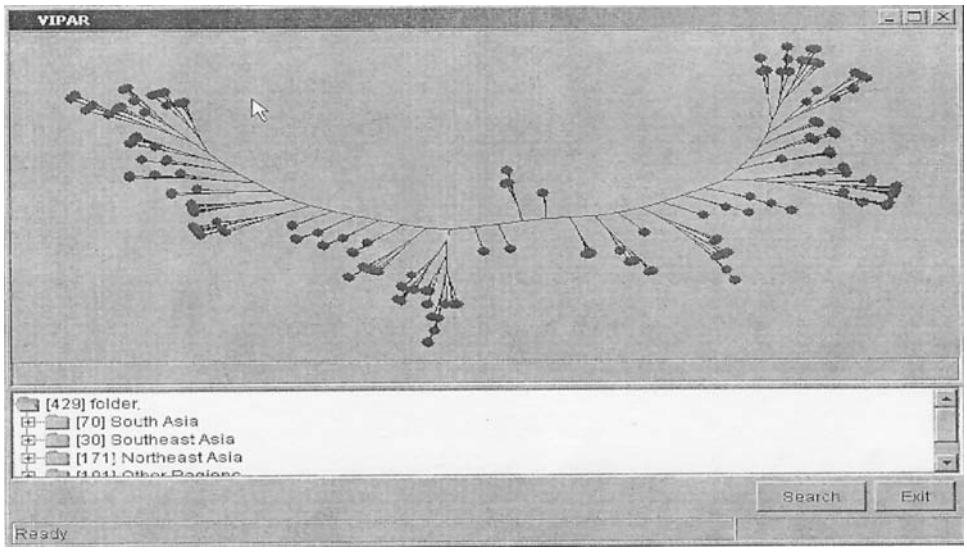


**Figure 4. Large scale dynamic clustering.**

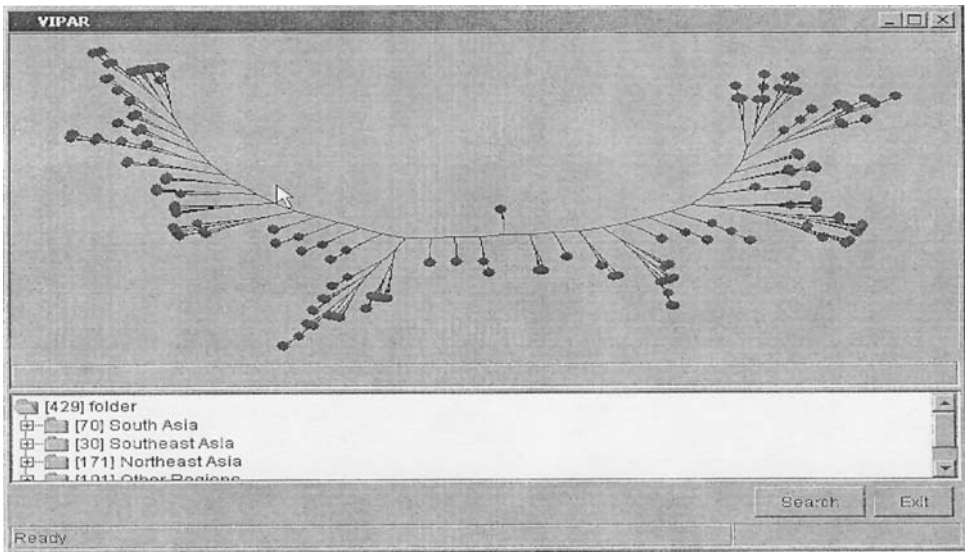


**Figure 5. Large scale full clustering.**

The results of this dynamic clustering algorithm are very positive. There is a significant reduction in the amount of time required to add and remove documents from the system. This allows the algorithm to be used in applications where the document set changes at a fast pace. Additionally, and most importantly, it appears that there is little difference in the fidelity of the resulting clusters between the full and approximate methods.

## Discussion

The results visibly indicate that it is indeed possible to cluster large volumes of textual information in an asynchronous environment. During the testing phase of development, we compared this algorithm against one that was identical except that the entire dissimilarity matrix was updated each time an article was added or removed. While the full update algorithm performance would not allow real-time use of the system, it provided comparison of the clusters produced using actual newspaper data. We were surprised to find that experiments using multiple

data sets resulted in very similar dendograms. It was very rare for any of the documents to agglomerate in a different order and generally the only difference was a slight change in the position of the leaf points of the dendogram. Since we use the distance between two clusters being agglomerated as a factor in the distance between leaves in the dendogram, this variance in the leaf node position is attributable to the approximations in the dissimilarity matrix.

We were very pleased at the efficiency gains we observed, particularly since the cost in accuracy was so small. We have many ideas for future expansion of this algorithm. One idea is to incorporate multi-threading. For example, to have one higher priority thread that handles document additions and removals and user clustering requests, while another lower priority thread updates the dissimilarity matrix. This would result in dissimilarity matrix that is less approximate during times of low system activity and more approximate during the time when many articles are being added and removed and when many users are requesting groupings. Another idea is the incorporation of Latent Semantic Indexing techniques to allow reduction of dimensionality of the vector space and allow even larger article sets. We also have relied on qualitative comparisons of the similarity of clusters. We plan to provide more formal metrics in the future.

**Future Work**

One of the challenges that we still face in this project is performance as the number of articles increases over 1000. The initial requirement of the project was to support 5 newspapers to demonstrate the feasibility of the approach. Over time, we have expanded these numbers to as high as 21 newspapers and a maximum of 2500 articles. Performance quickly degraded on our test computer with 256M of memory. The VSM model has a computational complexity of approximately $O(n^2)$, while the agglomerative clustering algorithm (using Ward's Update function) has a computational complexity of approximately $O(n^3)$ where n is the number of documents to be processed. This quickly limits the size of document sets that can be clustered. From our analysis, it appears that the memory usage in this algorithm is the major performance bottleneck. A quick evaluation shows that memory usage is increasing at approximately an $O(n^2)$ rate, see Figure 6. In figure 6, the memory usage on the vertical axis is shown versus the number of documents on the horizontal axis, with the solid line showing actual values from the VIPAR system. The dotted line is the result of using regression analysis to fit a curve to the data points; extending the curve give an estimate of memory requirements as the number of documents grows. This regression analysis indicates that using the current algorithms, doubling the number of articles to be processed required quadrupling the amount of memory. Even with our dynamic algorithm, additional algorithmic improvements are going to be required to see significant increases in the number of newspaper articles that can be processed. We are currently working to address this problem so that tens of thousands of news articles could be comfortably processed without significant lost of precision.

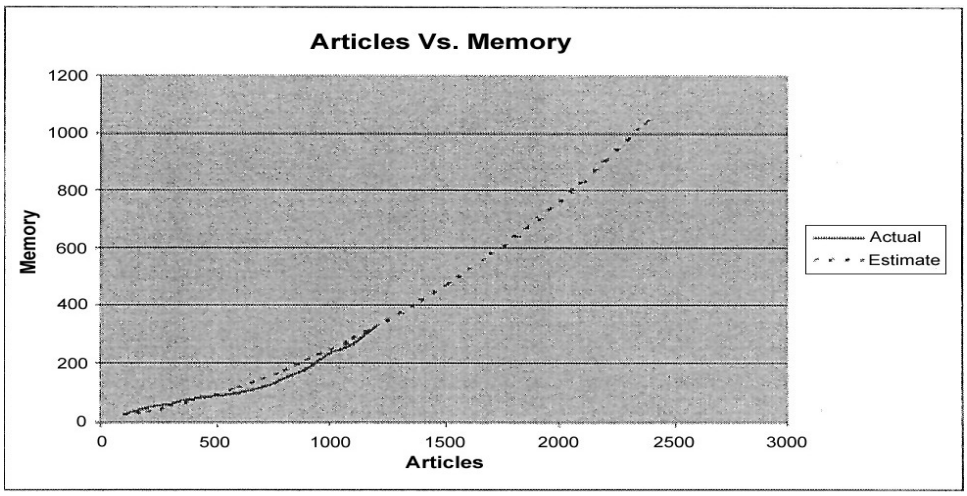


**Figure 6. Graph of the projected memory usage based on existing algorithms.**

## Conclusion

Previous agglomerative clustering variants have shown the value of the approach to organizing and classifying large amounts of information. However, before analysis could begin, a prior calculation was required because of the substantial computational complexity of (1) creating vectors to represent the documents using TF-IDF, then (2) comparing the vectors with proximity measures and storing the results in a dissimilarity matrix, and then finally (3) performing the agglomerative clustering. Each new document added or old document removed changed the underlying TF-IDF vectors and required another set of prior calculations before analysis could resume. This prevents the use of this approach to provide the capability of real-time analysis of dynamic datasets. Responding to the need for such a capability, we have developed a method to dynamically update the vectors and dissimilarity matrix while preserving sufficiently high accuracy.

Using this strategy, every time a single document is added or deleted, the oldest 5% of the cells in the dissimilarity matrix and the document vectors that correspond to them are updated. This approach allows documents to be quickly added and removed from the document set and results in sufficiently accurate agglomerative clustering that provide little discernable difference to an information analyst.

The significance of this computationally feasible approach is that it shows *little discernable difference and no functional difference* when compared to the computationally too-expensive approach of fully updating the VSM and dissimilarity matrix. While not as mathematically accurate, this partial updating of the dissimilarity matrix was computationally feasible within the dynamic data environment of VIPAR. This approach provided the same document groupings with only insignificant variations in the dendogram visualizations, while calculating the groupings in a fraction of the time and thus providing real-time responsiveness.

In this paper, we show an example of software agents traversing 13 Internet newspapers, downloading thousands of articles throughout the course of each day, and making these articles available to analysts in real-time. In this dynamic environment, the clustering algorithm has performed well, providing increased power to analysts with its novel approach.

## References


[1] A. K. Jain, M. N. Murtly, and P. J. Flynn, *Data Clustering: A Review*. ACM Computing Surveys, Vol. 31, No.3., 1999

[2] H.P. Luhn, *A statistical approach to mechanized encoding and searching of literary information*, IBM J. Res. Develop. I, 4 , 1957

[3] G.K. Zipf, *Selective Studies and the Principle of Relative Frequency in Language*, Harvard University Press, Cambridge,Massachusetts,1932

[4] G. Salton, M.E.Lesk, *The SMART Automatic Document Retrieval System"* Communications of the ACM, vol 8, Number 6, June 1965

[5] K. Sparck Jones, *Index term weighting*, Information Storage and Retrieval, 9, 619-633, 1973

[6] K. Sparck Jones, *Automatic Indexing: A State of the Art Review*, review commissioned by the Office for Scientific and Technical Information, London, 1974

[7] G. Salton , C. Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing and Management: an International Journal, v.24 n.5, p.513-523, 1988

[8] M. Steinbach, G. Karypis, and V. Kumar, *A Comparison of Document Clustering Techniques*. University of Minnesota Technical Report #00-034, 2000

[9] A. V. Leouski, W. B. Croft, *An Evaluation of Techniques for Clustering Search Results*, Technical Report IR-76, Department of Computer Science, University of Massachusetts. 1996

[10] Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu, *Learning Approaches for Detecting and Tracking News Events*. IEEE Intelligent Systems, Vol 14, No.4., 1999

[11] H. Frigui and R. Krishnapuram *A Robust Competitive Clustering Algorithm with Applications in Computer Vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 21, No. 5, pp. 450-465,1999

[12] M.F. Porter, *An Algorithm for Suffix Stripping*, Program, Vol. 14, No. 3, pp. 130-137, 1980

[13] J. H. Ward *Hierarchical Grouping to Optimize an Objective Function*, Journal of the American Statistical Association, Vol 24, No 301, 1963

[14] T. E. Potok, M. T. Elmore, J. Reed, S. F. Samatova, and N. Ivezic, *VIPAR: Advanced Information Agents discovering knowledge in an open and changing environment*, Fifth International Workshop CIA-2001 on Cooperative Information Agents 2001

[15] J. Felsenstein, *PHYLIP (Phylogeny Inference Package)*, Distributed by the author. Department of Genetics, University of Washington, Seattle 1980, 2004