A Multi-Agent System for Distributed Cluster Analysis

Joel W. Reed, Thomas E. Potok, and Robert M. Patton
Oak Ridge National Laboratory
P. O. Box 2008, MS 6359
Oak Ridge, Tennessee 37831
{reedjw, potokte, pattonrm}@ornl.gov

Abstract

One of the approaches used to improve the accuracy and relevancy in information retrieval is cluster analysis. Clustering methods determine relationships among text documents, and allow the determination of similar groups or clusters of documents. These methods are computationally expensive, thereby limiting their use to a relatively small set of documents. This paper describes a multi-agent system to cluster large data sets. This technique is then compared to hierarchical agglomerative clustering using a small set of text data. Results show that the agent-based approach can significantly reduce the time required to cluster large data sets.

1. Introduction

There is a wealth of textual information readily available over the Internet. There are many search engines and portals available to retrieve this information. Unfortunately, the retrieval accuracy and relevancy is often quite low [2]. One of the approaches used to improve this accuracy and relevancy in information retrieval is to use advanced textual analysis methods, such as cluster analysis. These clustering methods determine relationships among text documents, and allow the determination of similar groups or clusters of documents. However, clustering methods are computationally expensive, typically between O(n^2) and O(n^3), where *n* is the number of documents to be clustered [10]. Practically, this limits its use of these methods to a relatively small set of documents.

To address this problem, this paper describes a new distributed clustering technique based on agent technology. This new clustering technique is then compared to hierarchical agglomerative clustering (HAC). Preliminary results show that this new approach can handle larger document sets much more quickly and efficiently than the agglomerative hierarchical approach.

2. Background

The basic technique of hierarchical agglomerative clustering involves several steps. First, the set of documents is processed by removing all of the stop words, or commonly occurring words that have little meaning within a document. The next step is to stem the remaining words, or to reduce them to their root form. The words within the document set are then used to create a vector that represents the document. A set of these document vectors can be used to create a vector space model (VSM) that represents the relationships between the documents [7]. In a VSM, each unique word within a document collection is represented as a dimension in space and each document is represented by a vector in that multidimensional space. The numeric representation of a word within a document (which is a single element of the document's vector and a single dimension in the VSM) is typically based on the frequency of the word within a specific document (local term frequency), and the frequency within the document set (global term frequency). A word with a high frequency within a specific document and low frequency within a set of documents, often called Term Frequency Inverse Document Frequency (TFIDF) [8], produces a high value. Words with high values have been shown to be very useful in accurately classifying and retrieving documents. Since a word frequency over a set of documents is required, all documents within a set must be analyzed before a VSM can be constructed. This part of the clustering process has a time complexity of O(n^2).

The VSM can be used to define a similarity value between a pair of documents. Typically, this value is obtained by using Euclidian distance between the vector endpoints or using the dot product to calculate the cosine of the angle between the pair of document vectors. All of the possible pairs of document in the collection can be compared and their similarity values collected to create a similarity matrix. This similarity matrix is needed to compute the document clusters. The agglomerative clustering process begins by placing each document within its own cluster. Next, the pair of clusters that contain the most similar documents (as defined by the similarity matrix) are merged into a single cluster. At this point, the similarity matrix values must be updated to reflect the merge. This process iterates until all of the documents are in a single cluster. This part of the clustering process has a time complexity of $O(n^3)$.

There are several issues with this approach to clustering. First, the TFIDF calculation cannot be easily distributed across multiple computer systems because of its dependence on a global term frequency. Second, using TFIDF and HAC on a large, dynamic data set does not work very well. As new documents are added to the data set, global term frequencies would need to be updated, which would ultimately require the need to re-cluster the data set. Finally, the combined computational complexity of TFIDF and HAC makes this approach infeasible for large data sets.

3. Multi-agent distributed clustering

To address these issues, a multi-agent system for distributed clustering of text documents was developed. This approach does not depend on a global term frequency count, and is essentially a hybrid HAC and K-means clustering approach.

To implement this multi-agent clustering system, several types of agents are used. At the lowest level, there are sub-cluster agents. Each sub-cluster agent represents a set of documents that are very similar. Above these, there are cluster agents. Each cluster agent represents a set of sub-cluster agents whose document sets have some similarities, but are not as similar as documents in a sub-cluster agent's document set. Above the cluster agent is the master cluster agent. These master cluster agents manage a set of cluster agents. There is not any implied relationship between the set of cluster agents (and their associated documents) managed by a master cluster agent. The master cluster agents are used to move cluster agents between other master cluster agents on different computer systems and therefore achieve better load balancing. Each computer in the distributed clustering system has only one master cluster agent. Finally, there are document multiplexer agents in the system. The document multiplexer agents accept new documents (and their representative vectors) and help to insert them into the clustering system. All of these types of agents work together to organize a set of documents as shown in Figure 1.

Incorporating a new document into the distributed clustering system requires several steps. First, a document vector representing the document must be created. The document vector is then used to evaluate how the document compares to the documents that already exist in the clustering system. Finally, either the document is given to a software agent representing a set of document very similar to it, or if the document is unlike any currently in the system, a new software agent is created to represent it.

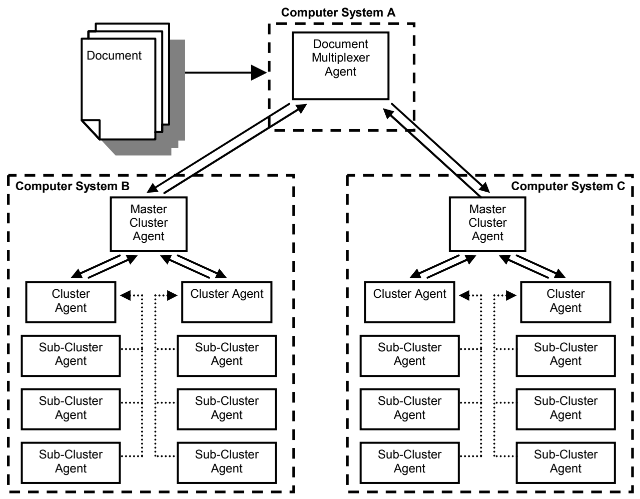

Figure 1. Multi-agent architecture

A new document is first processed into a document vector, which will represent the document's content. This vector can then be used to compare it with other documents or sets of documents. To create a document vector, all of the words in the document are filtered by a set of rules. Word below a minimum length (typically 3 or 4 characters) and common words like "and" and "the" are ignored. Once this is complete, the remaining words are processed into a set of tokens. This process can be as simple as creating a token for each word in the document or it can use some very complex algorithms to group several word together into a single token. We use a reasonable simple algorithm that creates a token for each word in the document and creates a token for each pair of adjacent words that are not separated by a word below the minimum allowable length or a stop word. Each word or phrase in the token list is replaced with a canonical synonym if it exists. For example, all occurrences of "bin laden", "usama bin laden", "usama", and "osama" are replaced with the token "osama bin laden". Next, each token is stemmed [3][4]. A vector is created by taking each value in the token list and associating it with its frequency in the original document. The final document vector is obtained by normalizing this vector. Once this vector is calculated, the new document can begin the evaluation process.

The document evaluation process calculates where a new document will best fit into the clusters of documents already in the system. This process begins when the document multiplexer agent receives a new document and its representative document vector. As soon as this happens, the document vector is sent to all of the master cluster agents for evaluation. The master cluster agent then passes the vector to each of the cluster agent that it oversees. As mentioned above, each of these cluster agents is responsible for a set of sub-cluster agents, which are responsible for a set of documents. The sub-cluster agent maintains a single composite vector that represents all of the documents it oversees. Likewise, each cluster agent maintains a single composite vector that represents all of the sub-cluster agents under it. These composite vectors are calculated by summing the vectors to be represented and the normalizing the result. Each of the cluster agents then evaluates the new document, by comparing the document's vector to its own composite vector using cosine measure, Euclidean distance, or one of the other common methods. The comparison result is sent back to the master cluster agent. The master cluster agents collect all of the comparisons and send the closest one back to the multiplexer agent. Similarly, the multiplexer agent will collect all of the values from the master clusters agents and find the best match value. This value will indicate either that a cluster of documents similar to the new document already exists in the system or that no similar documents are in the system. If a cluster of similar documents exists, the new document needs to be incorporated into that cluster; otherwise, a new cluster needs to be created.

Once the document evaluation process is complete, if the document multiplexer agent has decided that a similar set of document exists in the system, the new document needs to be incorporated into that document set. The multiplexer agent will send the document and its vector to the master cluster agent that it determined represented a similar cluster set. The master cluster agent will then forward the document and vector to the cluster agent, which responded with the closest similarity value. During the evaluation process, the document vector was compared to the cluster agent's composite vector rather than to each sub-cluster's composite vector, so to find out which sub-cluster agent contains the most similar documents, this must occur now. Each sub-cluster agent compares the document vector to its composite vector and sends the similarity value up to the cluster agent above it. The cluster agent examines all of the similarity values and determines whether the document is close enough to a sub-clusters document set to be added to it or whether a new sub-cluster agent needs to be created for the document.

If the new document evaluation process determined that there were no other documents similar to the new document, then a new cluster agent must be created for it. First, the multiplexer agent will query master cluster agents to see what their current load is like. The multiplexer agent will determine which master cluster agent has the least load and it will send the document and vector there. The master cluster agent will create a new cluster agent, which will create a new sub-cluster agent for the new document.

Because of the process used to add documents to the system, the set of documents represented by a sub-cluster agent are very similar and likewise, the set of sub-cluster agents represented by a cluster agent contain documents less similar, but still related to one another. These relationships can be used to generate any type of desired clustering visualization. At the cluster agent level and at the sub-cluster agent level, the composite vectors can be compared to one another to determine relationships between sets of documents.

4. Comparison

As a data source for comparing HAC and the multi-agent approach described here, the Text Retrieval Conference (TREC) 1996 corpus was used. This corpus contains 130,000 documents of worldwide news events [9]. To compare the two clustering methods, several sets of randomly selected documents were chosen from the TREC corpus.

Six sets of documents were created that varied in size. For the purposes of this paper, the primary focus of the comparison is on the time required and the memory usage needed for clustering. Both approaches were performed on a single machine and were implemented using Java. For larger or distributed data sets, the multi-agent approach could be used on multiple machines.

Table 1 and Figure 2 show the time results of both approaches. Notice that for smaller data sets, the multi-agent approach performed poorly, requiring nearly twice the amount of time. This is due to the communication overhead of the agent architecture. However, this communication overhead becomes insignificant once the document set reaches 500 in size. The speed-up then becomes increasingly significant as the number of documents increases. In fact, the multi-agent system could handle 2000 documents in less time than the TFIDF/HAC approach could handle 1000 documents.

Table 1. Time Comparison

Number of Documents	Multi-agent Time (seconds)	TFIDF/HAC Time (seconds)
50	2.9	1.8
100	6.3	3.3
250	24.1	15.6
500	71.1	119.8
1000	195.3	868.1
2000	576.0	N/A

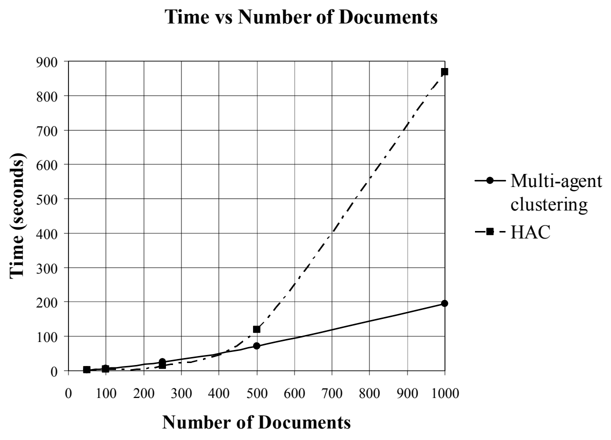

Figure 2. Comparison of time between the multi-agent and HAC approaches

Table 2 and Figure 3 show the memory usage comparison between the two approaches. Notice that the usage is not significantly different. Notice also that as the data set doubled in size, nearly doubled as well. While the agent architecture does not store or use global term frequency counts, the memory usage for the multi-agent system remains approximately the same due to the memory requirements of the agent infrastructure and communication.

Table 2. Multi-agent clustering memory usage

Number of Documents	Multi-agent Memory Usage (MB)	TFIDF/HAC Memory Usage (MB)
50	9.4	13.5
100	12.9	17.6
250	22.5	29.3
500	49.8	52.2
1000	101.2	109.1
2000	212.6	N/A

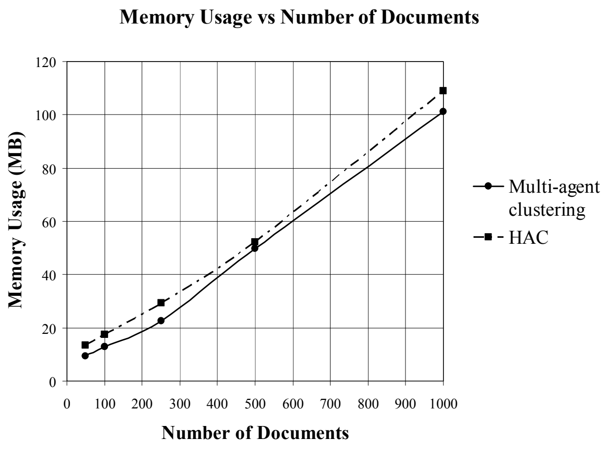

Figure 3. Comparison of memory usage between the multi-agent and HAC approaches

There are a number of complex issues in comparing clustering results. In a preliminary experiment, we had several human subjects manually cluster a set of documents so that the results could then be compared to both the HAC and multi-agent clustering methods. The multi-agent approach yielded results closer to the manually generated clusters than the HAC method. Clearly more analysis is required; however, using our test dataset, we can not reject the hypothesis that the multi-agent clustering method achieves results very similar to manually derived sets.

5. Conclusion

The distributed agent-based clustering worked surprisingly fast for large document sets. In addition, it allows for distributed processing of documents, and a hybrid k-mean and hierarchical clustering result. With this approach, it will be possible to cluster massive amounts of textual information in relatively short amounts of time, due to the scalability of the agent architecture. We plan to explore further the scalability of the agent architecture presented in this paper.

6. References

[1] Mark T. Elmore Thomas E. Potok and Frederick T. Sheldon "Dynamic Data Fusion Using An Ontology-Based Software Agent System", Proceedings of the IIIS Agent Based Computing, Orlando, 7/2003.

[2] D. Hawking, N. Craswell, P. Thistlewaite, D. Harmon, "Results and Challenges in Web Search Evaluation", Computer Networks, Vol. 31, No. 11-16, pages 1321-1330, 1999.

[3] J. B. Lovins, “Development of a Stemming Algorithm”, Mechanical Translation and Computational Linguistics, Vol. 11, pages 22-31, 1968.

[4] M. F. Porter, “An Algorithm for Suffix Stripping”, Program, Vol. 14, No. 3, pages 130-137, 1980.

[5] Thomas E. Potok, Mark Elmore, Joel Reed and Frederick T. Sheldon, “VIPAR: Advanced Information Agents discovering knowledge in an open and changing environment” Proc. 7th World Multiconference on Systemics, Cybernetics and Informatics, 7/2003.

[6] Joel W. Reed and Thomas E. Potok, “A Multi-Agent System for Analyzing Massive Scientific Data” Proceedings of the International Conference on Software Engineering, 2003.

[7] G. Salton, M. Lesk, “Computer Evaluation of Indexing and Text Processing”, Journal of the ACM, Vol. 15, No. 1, pages 8-36, 1968.

[8] G. Salton, C. Buckley, “Term Weighting Approaches in Automatic Text Retrieval”, Information Processing and Management, Vol. 24, No. 5, pages 513-523, 1988.

[9] E. Voorhees, and D. Harman, “Overview of the Fifth Text REtrieval Conference (TREC-5),” Proceedings of the Fifth Text Retrieval Conference.

[10] Y. Zhao, and G. Karypis, “Evaluation of Hierarchical Clustering Algorithms for Document Datasets”, University of Minnesota, Technical Report #02-022, 2002.