

## A New Agglomerative Hierarchical Clustering Algorithm Implementation based on the Map Reduce Framework

<sup>1</sup> Hui Gao, <sup>2</sup> Jun Jiang, <sup>3</sup> Li She, <sup>4</sup> Yan Fu

<sup>1, First Author</sup> *Web Sciences Center, School of Computer Science and Engineering, University of Electronic Science and Technology of China, huigao@uestc.edu.cn*

<sup>\*2, Corresponding Author</sup> *School of Software, University of Electronic Science and Technology of China, jiangjuntokyo@163.com*

<sup>3,4</sup> *School of Computer Science and Engineering, University of Electronic Science and Technology of China, sheli@uestc.edu.cn, fuyan@uestc.edu.cn*

doi: 10.4156/jdcta.vol4.issue3.9

### Abstract

*Text clustering is one of the difficult and hot research fields in the text mining research. Combing Map Reduce framework and the neuron initialization method of VPSOM (vector pressing Self-Organizing Model) algorithm, a new text clustering algorithm is presented. It divides the large text vector dataset into data blocks, each of which then processed in different distributed data node of Map Reduce framework with agglomerative hierarchical clustering algorithm. The experiment results indicate that the improved algorithm has a higher efficiency and a better accuracy.*

**Keywords:** *Text clustering, Map Reduce, Initial classification, Agglomerative hierarchical clustering*

### 1. Introduction

Text clustering has been adopted as an effective tool for sorting a large amount of documents to assist readers with different interests. It mainly based on the famous cluster assumptions: the elements in the same cluster would be more similar than the elements outside the cluster. As an unsupervised machine learning methods, there is no need training process, with a higher degree of flexibility and the automation of handling capacity, text clustering has become one of important contents in text mining. Text clustering tasks [1] with high dimension [2] and large datasets [3] will extremely compromise the performance of clustering algorithms. Therefore, the common technique is feature selection from reduced feature dimensions [4] and the utilization of distributed processing [5].

In the last decades, many methods have been introduced for text clustering, such as: K-means based on partition, AHC based on hierarchical and so on. K-means method is the prevalent method which need selecting K clustering centers at random in advance, then allocating each text to the nearest center based on Euclidean distance, and calculating average vector of every reallocated clustering, then executing iteration on new clustering centers [6]. AHC (agglomerative hierarchical clustering) method can display the clustering process clearly [7]. However, these methods have the same disadvantage that need make sure the number of clustering in advance, the time and space complex rate of them make it almost impossible for them to process the large dataset, so they can not apply in text clustering satisfactory.

To solve these problems, many improved algorithms are proposed [8][9][10]. Aim to improve performance, most of them choose to reduce the distance calculation process. Like the method based on the k-d tree structure and pruning function proposed by Alsabti [9], P-CLUSTER, the parallel clustering algorithm utilizes three kinds of pruning methods proposed by Dan Judd [11] and the parallel algorithm based on the k-d tree structure proposed by Attila Gursoy and ilker Cengiz[12]. Obviously, by reducing distance or similarity calculation, the algorithm doesn't guarantee accuracy. We use parallel computing, assign the distance computing to different nodes in a distributed environment, which improved the efficiency and ensure the effectiveness [13], also the algorithm is relatively simpler.

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. One of its subprojects, named Map Reduce, provides a software framework for easily

writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner [14]. SOM is the Self-Organizing feature Map proposed by Kohonen. Kohonen believed that, a nervous network's outer input receiving model was to divide the nervous network into different regions, these regions have different corresponding features to the input model, and such an input process is finished automatically [15]. Aiming to solve the curse of dimensionality in large-scale documentary dataset clustering task, a novel self-organizing-mapping algorithm for large-scale and high dimensional data is proposed [16]. This algorithm presents a method to divide the large dataset into initial clusters and ensure there's no intersection between two clusters. By compressing neurons' feature sets and only selecting relative features to construct neurons' feature vectors, the clustering time can be dramatically decreased. Simultaneously, because the selected features can effectively distinguish different documents which are mapped to different neurons, the algorithm can avoid interferences of irrelative features and improve clustering precision. Based on the considerations above, we present a new agglomerative hierarchical clustering algorithm based on the Hadoop framework and initial classification.

## 2. The algorithm

The main idea of our approach is: Divide the original documents' vectors' set into partitions with the neuron set initialization method. A large-scale dataset clustering task is reduced to acceptable clustering tasks on relatively smaller dataset. We process the documents' vectors with traditional agglomerative hierarchical clustering algorithm and get the final clustering results.

The first step of our algorithm is initial classification. Fine divided documents' vectors sets with low intersection ratio are decisive in our clustering task. By means of initial classification we distribute the original dataset to many data nodes in Map Reduce framework. We introduce the "Forward-backward" initialization method to solve the unbalanced distribution of documents' vectors among the partitions.

Before the initialization step we have to extract the characteristic of the documents and express it. Vector space model (VSM) is commonly adopted to express documents. In this model, each document  $d$  is considered as a vector in a vector space. TF-IDF is used as a common measurement of characteristic weight. IDF means inverse document frequency, which is used to avoid the words that may have very high term frequency but aren't really important in the document. Considering the limitation of documents' samples in our database, we decide to utilize the term frequency of Sougou and introduce our keyword weight calculating method. See Eq. (1) below.

$$\frac{para \cdot TF(d, t) \cdot \sigma}{(1 + \log(1 + SogouF(t))) \cdot F\_MAX} \quad (1)$$

In Eq. (1), the value of  $para$  depends to the part keyword  $t$  located in document  $d$ . One word in the title is certain to have larger weight than the words in the body.  $SogouF(t)$  means the term frequency of keyword  $t$  in the Sougou dictionary.  $F\_MAX$  means the largest term frequency in document  $d$ .  $\sigma$  is an attenuation coefficient. Weights of the words out of range of Sougou dictionary are calculated with Eq. (2).

$$\frac{Pre\_T(d, t)}{Max\_T(d)} \cdot Value\_Max(d) \quad (2)$$

In Eq. (2),  $Pre\_T(d, t)$  means the term frequency of  $t$  in document  $d$ ,  $Max\_T(d)$  means the largest term frequency in document  $d$ ,  $value\_Max(d)$  means the largest word weight in document  $d$ .

Now, suppose  $DOC$  is the documents' vectors' list prepared to be clustered, and its' size is  $N$ . Marked the  $k$ th document's vector as  $D_k$ . Correspondingly, the neuron set named as  $NEURON$ , the  $i$ th element of which marked as  $N_i$ . Initial classification process is described below.

- 1) Scan the set  $DOC$  sequentially.  $D_k$  is the document's vector scanning currently.

- 2) Calculate the similarities between  $D_k$  and each element  $N_i$  in the NEURON with Eq. (3). If all of the similarities equal 0, go to step 3, otherwise go to step 4.

$$Sim(D_k, N_i) = \begin{cases} 0 & |ISF_{ki}| < \frac{|FS(D_k)|}{3} \\ \sum_{inf \in ISF_{ki}} \frac{DW_k(inf) \cdot NW_i(inf)}{DW_k(inf) + NW_i(inf)} \cdot \log |ISF_{ki}| & \text{others} \end{cases} \quad (3)$$

In Eq. (3),  $FS(D_k)$  is the characteristic set of  $D_k$ ,  $ISF_{ki}$  is the intersection of characteristic set between  $D_k$  and  $N_i$ .  $DW_k(inf)$  is the weight of characteristic  $inf$  in document  $D_k$ , while  $NW_i(inf)$  means the weight of characteristic  $inf$  in neuron  $N_i$ .

- 3) Insert  $D_k$  to NEURON as a new neuron, and go to step 5.
- 4) Find out the neuron which has the largest similarity value with  $D_k$ , named it as  $N_m$ . Adjust the characteristics and the weight of them in  $FS(N_m)$  with Eq. (4).

$$NW_m(df) = \begin{cases} 0.1 & df \notin FS(N_m) \\ NW_m(df) + 0.1 \cdot (DW_k(df) - NW_m(df)) & df \in FS(N_m) \end{cases} \quad (4)$$

In Eq. (4), if  $FS(N_m)$  contains the characteristic  $df$ , calculate the new weight of it, otherwise simply set it to 0.1.

- 5) If  $D_k$  is the last document in  $DOC$ , invert the original  $DOC$  list and go to step 6, otherwise scan the  $k+1$  th document and go to step 2.
- 6) Process the inverted  $DOC$  list from step 1 to step 5.

After initial classification, we process the results with traditional agglomerative hierarchical clustering algorithm. See the details about this algorithm in [7].

### 3. The Implementation

When the size of dataset is getting large the calculation of the similarities between all pairs of vectors in agglomerative hierarchical clustering process becomes very time consuming, and therefore we introduce the Map Reduce framework for the implementation of our algorithm.

Map Reduce is a software framework for distributed processing of large data sets on compute clusters. It consists of a single master and one slave per data node. The master is responsible for scheduling the jobs' component tasks on the slaves. It uses SSH (Secure Shell) to start and stop the guard processes in each data node, monitor them and re-executing the failed tasks. The slaves parallel execute the tasks as directed by the master. There is no communication between each two data node. We configure SSH to authenticate with the public key, so the implementation of instructions from master in each data node doesn't have to enter security password.

The Map Reduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

Based on the considerations above, our cluster algorithm has to construct Map and Reduce class of its' own. Before this process, we put all characteristic vectors of the cluster into a global mapping structure for the facilitation of calculating. In this mapping structure, the key means the global ID of this cluster and the value represents the characteristic vectors in this cluster. Map takes these ID value as input. Cluster pair in the similarity calculating function also is represented by their ID pair. For example, in similarity calculating function, "1 - 2" means calculating the similarity between cluster 1 and cluster 2.

Suppose the framework consists of N computers, one of which is master host H, the other are data nodes  $D_i$  ( $i=1 \dots N-1$ ). And initial classification process results with M initial clusters. The implementation can be outlined as follows:

## A New Agglomerative Hierarchical Clustering Algorithm Implementation based on the Map Reduce Framework

Hui Gao, Jun Jiang, Li She, Yan Fu

- 1) Initialization: Use the clusters' set, result of initial classification, to build the global mapping structure of cluster characteristic vectors in the master host;
- 2) Similarity calculation process: Suppose  $m$  ( $m > N$ ) is the number of clusters in this iteration,  $H$  distributes the tasks to each  $D_i$  with the cluster ID pair: <1-2> assigned to  $D_1$ , <1-3> assigned to  $D_2$ ,... <1-N> assigned to  $D_{i-1}$ , <1-N+1> assigned to  $D_2$ ,...
- 3) Data nodes deal with their tasks parallel, one task <i-j> in  $D_i$  is described below:
  - a) It takes out the corresponding characteristic vectors  $V_i$  and  $V_j$  from the global characteristic vectors' mapping structure;
  - b) Processes the similarity calculating with cosine similarity calculation method. See Eq. (5) below;

$$Sim(\vec{V}_i, \vec{V}_j) = \frac{\sum_{k=1}^n \omega_k(\vec{V}_i) \times \omega_k(\vec{V}_j)}{\sqrt{\left(\sum_{k=1}^n \omega_k^2(\vec{V}_i)\right) \times \left(\sum_{k=1}^n \omega_k^2(\vec{V}_j)\right)}} \quad (5)$$

- c) Takes the mapping structure of clusters' ID pair and corresponding similarity <"i-j",  $Sim(V_i, V_j)$ > as the result, and saves it in a temporary file;
  - d) Deals with next cluster pair.
- 4) According to the similarity value, each data node processes reverse sorting;
- 5) Each node sends its' result list to the Master, which processes merging operation with the help of Reducer thread and get the overall sorted list;
- 6) Check the result of this iteration: The key of the first entry of global sorting result is the most similar clusters' ID pair in this iteration. If the value is less than the initial threshold  $t$ , go to step 8, else go to step 7;
- 7) Merge the two clusters and adjust the characteristic vector of the new cluster. Renew the global characteristic vector mapping structure and go to step 2;
- 8) Stop AHC process.

## 4. Experiments' result

Experiments are implemented on the Map Reduce framework, which consists of 20 computers, and each of them equipped with dual-core Intel processor, 4 GB memory, 320 GB hard disks and a gigabit network interface card. These machines are placed in the same LAN segment. The master host, which is the name node, maintains the IP address and the corresponding host name of the other data nodes.

### 4.1. Accuracy and Recall rate test

The experiment samples are extracted from Google time tunnel information by web crawler. Categories of the samples are human-annotated before. Manual annotation of mass data is unrealistic. So we use a small number of data in this task to ensure there is no taxonomy overlaps or redundancy. See the comparison experiments' result in Table 1.

**Table 1.** Result of comparison test

| Category | Number of samples | Accuracy rate (%)          |            |                      | Recall rate (%)            |            |                      |
|----------|-------------------|----------------------------|------------|----------------------|----------------------------|------------|----------------------|
|          |                   | <i>Traditional k-means</i> | <i>AHC</i> | <i>Our algorithm</i> | <i>Traditional k-means</i> | <i>AHC</i> | <i>Our algorithm</i> |
| Type 1   | 101               | 87.5                       | 63.6       | 90.91                | 83.2                       | 59.8       | 89.1                 |
| Type 2   | 1491              | 90.6                       | 93.6       | 98.58                | 49.7                       | 53         | 88.46                |
| Type 3   | 82                | 93.7                       | 96.2       | 67.96                | 70.8                       | 60.5       | 85.37                |
| Type 4   | 542               | 80.5                       | 100        | 96.15                | 45.5                       | 19.7       | 50.74                |
| Type 5   | 44                | 51.7                       | 77.2       | 85.42                | 42.3                       | 72.3       | 93.18                |

|         |     |       |       |       |      |       |       |
|---------|-----|-------|-------|-------|------|-------|-------|
| Type 6  | 227 | 99.2  | 98.7  | 95.59 | 51.1 | 65.6  | 95.59 |
| Average |     | 83.87 | 88.22 | 89.1  | 57.1 | 55.15 | 83.74 |

The accuracy of traditional k-means algorithm is lower than our approach by 5.23%, and the recall rate is lower by 26.64%; the result of traditional AHC (agglomerative hierarchical clustering) algorithm is lower than our solution by 0.88% and 28.59%. For the consideration of efficiency, our algorithm processes the raw dataset with initial classification before the AHC process. Therefore the accuracy rate declined in some case, but the overall recall rate is much higher.

## 4.2. Efficiency test

We use historical data crawling from web last year (2009-1-1 ~2009-12-31) in the database to test the efficiency of our approach. The number of documents in this test set is up to 1,217,680. See the result in Table 2.

**Table 2.** Result of efficiency test

| Number of samples | Single computer |                       | Our framework   |                       |
|-------------------|-----------------|-----------------------|-----------------|-----------------------|
|                   | <i>Time (s)</i> | <i>Speed-up ratio</i> | <i>Time (s)</i> | <i>Speed-up ratio</i> |
| 1,217,680         | 8553.46         | 1                     | 1153.69         | 7.414                 |

The speed-up ratio of one Map Reduce framework consists of N computers is defined as the processing time in one computer divides the time consumed in this framework on the same test set. The speed-up ratio of our cluster is 7.414, and the global efficiency is improved by 86.5%.

## 5. Conclusion

A new agglomerative hierarchical clustering algorithm is presented in this paper. It is implemented by Map Reduce framework. The approach divides the original documents' vectors' set into partitions with the method of initial classification, and then distributes the partitions to different data nodes of Map Reduce framework. Finally it processes the documents' vectors in each data node with traditional agglomerative hierarchical clustering algorithm. Benefit from the paralleled procession in each data node, the efficiency of clustering is improved by 86.5%. The speed-up ratio of our framework, which consists of 20 computers, is up to 7.414. Compared with traditional k-means and AHC algorithm, the accuracy especially the recall rate of our new approach is improved. Result of experiments show that our new algorithm implemented on Map Reduce framework can apply in large-scale dataset clustering satisfactory.

## 6. Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (60973069, 90924011) and the Scientific Research Foundation for the Returned Overseas Chinese Scholars (GGRYJJ08-2).

## 7. References

- [1] A. K. Jain, R. C. Dubes, Algorithms for Clustering Data, Prentice Hall, Englewood Cliffs, NJ, 1988.
- [2] P. Bradley, U. Fayyad, C. Reina, "Clustering very large database using EM mixture models", in Proc. 15th Intern. Conf. on Pattern Recognition, 2000, pp.76-80.
- [3] K. Nigam, A.K. McCallum, S. Thrun, T.M. Mitchel, "Text classification from labeled and unlabeled documents using EM", Machine Learning, 39 (2/3), 2000, pp.103-134.

A New Agglomerative Hierarchical Clustering Algorithm Implementation based on the Map Reduce Framework

Hui Gao, Jun Jiang, Li She, Yan Fu

- [4] M.H.C. Law, M.A.T. Figueiredo, A.K. Jain, "Simultaneous feature selection and clustering using mixture models", IEEE Transaction on Pattern Analysis and Machine Intelligence, 26(9), 2004, pp.1154-1166.
- [5] Lichen An, New Methods for Cluster Analysis in Distributed Environments, Zhejiang: University Of Zhejiang, 2006.
- [6] Maojia Li, K-means Algorithm and Parallelization, Chongqing: University Of Chongqing, 2003.
- [7] Zhang Zhen Ya, Cheng Hong Mei, Wang Jin, Wang Xu Fa, An Approach on the Data Structure for the Matrix Storing Based on the Implementation of Agglomerative Hierarchical Clustering Algorithm, Computer Science, 2006(1), pp.14-17.
- [8] Manasi N. Joshi, Parallel K-Means Algorithm on Distributed Memory Multiprocessors, 2003.
- [9] K.Alsabti, S.Ranka,V, Singh, An Efficient K-Means Clustering Algorithm, [http://www.cise.ufl.edu/\\_ranka/](http://www.cise.ufl.edu/_ranka/), 1997.
- [10] Liu Liping, Zhi-Qing Meng, A method of choosing the initial cluster centers, Computer Engineering and Applications, 2004.8, pp.179-180.
- [11] Dan Judd, P.McKinley, A.Jain, Large-Scale Parallel Data Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998.8, Vol.20, No.8, pp.871-876.
- [12] M.Hemalatha , P. Ranjith Jebah Thangiah, K.Vivekanandan, "A Distributed and Parallel Clustering Algorithm for Massive Biological Data", JCIT: Journal of Convergence Information Technology, Vol. 3, No. 4, pp. 84 ~ 88, 2008
- [13] Attila Gürsoy, Ilker Cengiz, Parallel Pruning for K-Means Clustering on Shared Memory Architectures, Springer-Verlag Berlin Heidelberg, 2001, pp.321-325.
- [14] Information on <http://hadoop.apache.org>.
- [15] Suo H.G., Wang Y.W., An improved k\_means algorithm for document clustering, Journal of Shandong University (Natural Science), 43(1), 2008, pp.60-64.
- [16] Liuming, Wangxiao Long, Liuyuan Chao, A Fast Clustering Algorithm for Large-scale and High Dimensional Data, ACTA AUTOMATICA SINICA, 35(7), 2009, pp859-866.