IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 33, No. 6, December, 2019, pp. 453-460
Journal homepage: http://iieta.org/journals/ria

Design and Application of a Text Clustering Algorithm Based on Parallelized K-Means Clustering

Hui Wang[1,2], Chengdong Zhou[1,2], Leixiao Li[1,2*]

[1] College of Data Science and Application, Inner Mongolia University of Technology, Hohhot 010080, China
[2] Inner Mongolia Autonomous Region Engineering & Technology Research Center of Big Data Based Software Service, Hohhot 010080, China

Corresponding Author Email: 20070000036@imut.edu.cn

https://doi.org/10.18280/ria.330608

Received: 1 May 2019
Accepted: 8 August 2019

Keywords:
text clustering, Word2vec, K-means clustering (KMC), canopy algorithm

ABSTRACT

The traditional text clustering algorithms face two common problems: the high dimensionality of computing vectors and poor calculation efficiency. To solve these problems, this paper explores deep into the K-means clustering (KMC), Hadoop and Spark big data technique, and then proposes a novel text clustering algorithm based on the KMC parallelized on big data platform. The propose algorithm is denoted as the SWCK-means. First, the Word2vec was adopted to calculate the weights of word vectors, and thus reduce the dimensionality of the massive text data. Next, the Canopy algorithm was introduced to cluster the weight data, and identify the initial cluster centers for the KMC. On this basis, the KMC was employed to cluster the preprocessed data. To improve the efficiency, a parallel design for the Canopy algorithm and the KMC was developed under the Spark architecture. The proposed algorithm was verified through experiments on a massive amount of online text data. The results show that our algorithm achieved more accurate classification effects than the traditional KMC, especially in handling a huge amount of data.

1. INTRODUCTION

In recent years, there is a rapid growth in online information, creating a massive amount of Internet text data. The text data is a type of unstructured data, featuring high dimensions, large data volume, and low value density. In the field of Chinese information processing, it is a research hotspot to process the massive amount of online text data effectively, and mine out the value of such data. How to classify large quantities of text is a key research direction in this field [1-3].

Currently, the clustering technique has been applied to several aspects of the mining of massive online text data, ranging from preprocessing, semantic analysis, file similarity analysis, corpus classification, and topic analysis [4-6]. During text clustering, the original texts are divided into several meaningful classes, such that the texts in the same class are more similar than those in different classes [6]. In this way, the text data can be effectively organized and managed. Through effective text clustering, the results of information retrieval tools are understood and utilized in a better way.

The most popular clustering technique is the k-means clustering (KMC), which is based on partitioning [7]. During clustering, the KMC needs to identify the number of clusters, k, and randomly select k initial cluster centers. However, it is often difficult to predetermine the number of clusters or initialize the suitable cluster centers. As a result, the KMC is prone to the local optimum trap, which severely affects the clustering result.

Many scholars have attempted to improve the random selection of the initial cluster centers of the KMC. For example, Yang et al. [8] introduced density and neighbors to the clustering process, and proposed the initial cluster center selection algorithm, which improves the clustering quality and stability of the KMC. Based on the latent Dirichlet allocation (LDA) topic analysis model, Zhang et al. [9] put forward an initial cluster center selection algorithm that is much more efficient than the KMC in operations. Considering the difficulty of k value predetermination in the KMC, Limwattanapibool and Arch-int [10] performed regional clustering selection through density-based spatial clustering with additive noise (DBSCAN), which automatically finds the suitable number of clusters and initial cluster centers for the KMC. Compared with the other improved algorithms, the DBSCAN enjoys high operation efficiency and good clustering effects. Wang et al. [11] developed an improved k value selection algorithm, in the light of exponential function, weight adjustment, bias term and the elbow method.

Overall, the available clustering algorithms only apply to the processing of small-scale data, failing to adapt to the information explosion in the Internet era. The exponential growth in online data files, coupled with the drastic increase in the dimension of text feature space, has seriously suppressed the classification ability and extended the runtime of clustering algorithms, raising higher requirements on clustering techniques [12].

Parallel computing is an effective way to improve the efficiency in largescale computing. However, some of the many traditional methods of parallel computing can no longer satisfy the growing demand for large-scale data processing on the Internet. For example, the message passing interface (MPI) [13], which emerged at the end of the 20th century, and grid computing [14], which appeared at the beginning of the 21st century, both face problems like complex development and poor scalability.

MapReduce [15] is one of the key technologies of parallel computing. The Hadoop provides a parallel computing

interface with MapReduce as the core and a Hadoop distributed file system (HDFS), and can process up to millions of nodes and the data at the magnitude of ZB [16]. Zhang et al. [17] presented a Hadoop-based parallel KMC algorithm for text clustering, and used the algorithm to solve the scalability problem of the traditional clustering algorithm in processing largescale data. However, the MapReduce module, which executes parallel processing, consumes too many time in reading and writing online and disk input/output (I/O) rather than computing tasks. This drags down the efficiency of the Hadoop architecture [18].

In addition to the merits of traditional MapReduce, Spark also boasts the advantages of fast processing, complex queries, and seamless integration with Hadoop. The efficiency of Spark is far better than that of Hadoop [19]. The Spark-based KMC parallel algorithm for text clustering greatly outshines the Hadoop-based algorithm in acceleration ratio and scalability [12], and thus fulfills the demand of largescale text data mining. Nevertheless, the Spark architecture often falls into the local optimum trap, for the KMC algorithm in the architecture is not optimized.

To improve the accuracy and efficiency of the KMC, this paper proposes a Spark-based Word2vec KMC algorithm (SWCK-means). Taking the HDFS of Hadoop as text storage system, the original data were preprocessed, and the feature word weights were calculated by Word2vec. Next, the text data were clustered by a hybrid algorithm based on the Canopy algorithm and the KMC. The cluster centers were identified by the CC, and taken as the initial cluster centers of the KMC. Finally, the proposed algorithm was verified through experiments on Precision, acceleration ratio and expansibility.

2. ALGORITHM DESIGN

2.1 Word2Vec algorithm

Text, as a type of unstructured data, must be represented in a form that can be recognized and processed by a computer [12]. The Word2Vec [20] algorithm provides an excellent text representation method. This is an open source word embedding method developed by Google in 2013. By this algorithm, the numerical weight of a word in a file can be determined under a given corpus and represented as a word vector. The Word2vec model is trained by a series of typical words from various files. Each word is mapped into a fixed-size vector. The dimension of the word vector generally falls between 50 and 200. The word vector can express the semantic information of the word to a certain extent. The probability that a word occurs can be calculated based on the C continuous words before or after that word.

There are two Word2vec models, namely, the continuous bag-of-words (CBOW) and the skip-gram. The former predicts the word vector of the central word based on the word vector of each word in the context window, that is, computes the occurrence probability of a word based on the C continuous words before or after that word. The Skip-gram follows an opposite principle. First, the word in the context window is estimated based on the central word. Then, the occurrence probabilities of the words preceding and following that word are computed, and used to modify the word vector of the central word, making the semantic information of the file more accurate. Figure 1 illustrates the two Word2vec models.

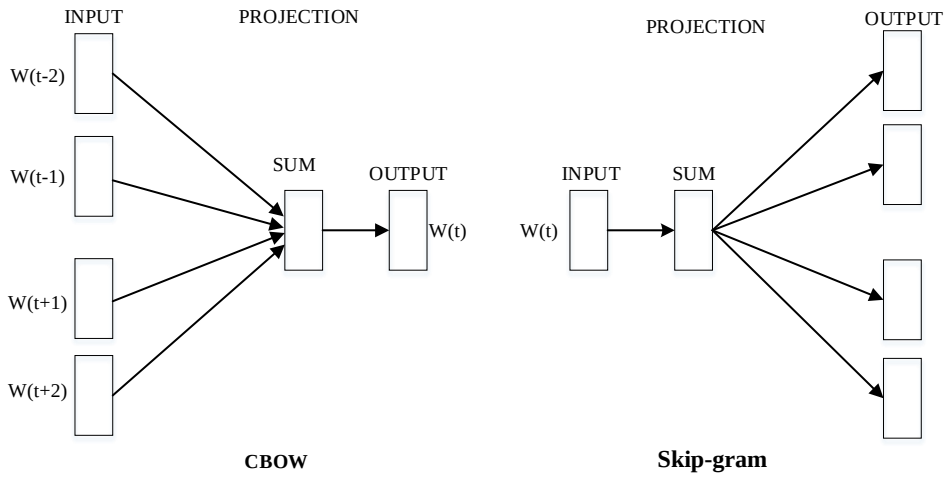

Figure 1. Word2vec models

By a vector space model, the Word2Vec algorithm expresses each file as weight vectors of feature words. Considering its rapid training and accuracy of semantic similarity, this paper chooses the skip-gram model of Word2vec to calculate the weights of word vectors, before clustering the text data.

2.2 Canopy algorithm for center point selection

Canopy [21] is a fast approximation clustering technique. The unique advantage of Canopy lies in the fast determination of clusters. Only one traversal is needed to identify the clusters. However, this advantage comes at the cost of the clustering accuracy [22].

As shown in Figure 2 below, the Canopy algorithm mainly involves the following steps [23]:

Step 1: Input the two distance thresholds (T1 and T2; T1 > T2) of the dataset D and Canopy.

Step 2: Take any point from the dataset. If no Canopy exists, treat the point as a Canopy center, and remove it from the dataset.

Step 3: Continue to take points from the dataset, calculate the distances from each point P to the center of each existing Canopy, and add P to the Canopy with a distance smaller than

T1. If a point P is farther than T1 from any of the existing Canopy centers, take this point as a new Canopy center.

Step 4: If the point is less than T2 to a Canopy center, add it to this Canopy and remove it from the dataset. The point is so close to the Canopy that it cannot be the center of any other Canopy.

Step 5. Perform Steps 3 and 4 on the points of the dataset until all the points have been allocated to their corresponding Canopies. Then, terminate the algorithm to end the clustering.

On the downside, the Canopy algorithm is a coarse clustering algorithm with a low accuracy. On the upside, the Canopy clustering is equivalent to preprocessing, which prevents the random selection of initial cluster centers in the KMC. Thus, the coupling between the Canopy algorithm and the KMC can effectively reduce the number of iterations and improve the clustering effect of the KMC.

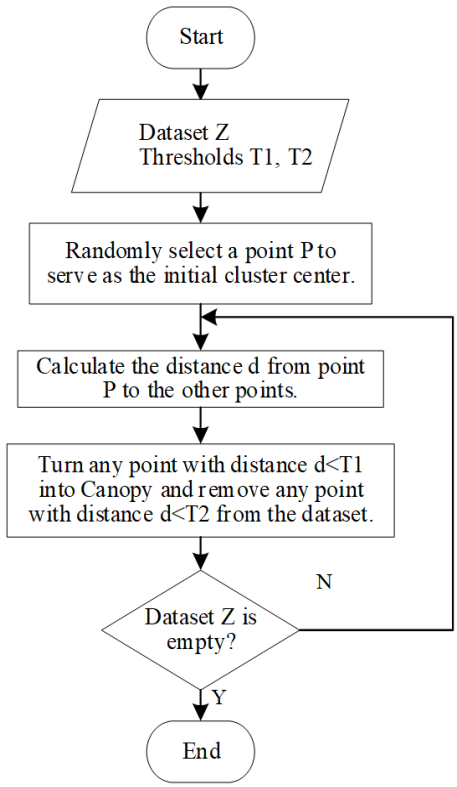

Figure 2. Workflow of Canopy algorithm

2.3 The KMC

The KMC [7] is the most widely used distance-based clustering algorithm. The core idea is to generate independent classes through iteration. In other words, all the objects should be classified into k different clusters, such that the texts in the same class are more similar than those in different classes.

As shown in Figure 3 below, the KMC is implemented in the following steps:

Step 1: Input the dataset D and the number of clusters k.

Step 2: Select k random points from D to serve as initial cluster centers.

Step 3: Calculate the distance from each remaining point to each cluster center, and allocate the point to the cluster with the shortest distance.

Step 4: Calculate the mean value of all points in each cluster and take it as the new cluster center.

Step 5: Repeat Steps 3 and 4 until the change of the cluster center is smaller than the preset threshold or the maximum number of iterations is reached.

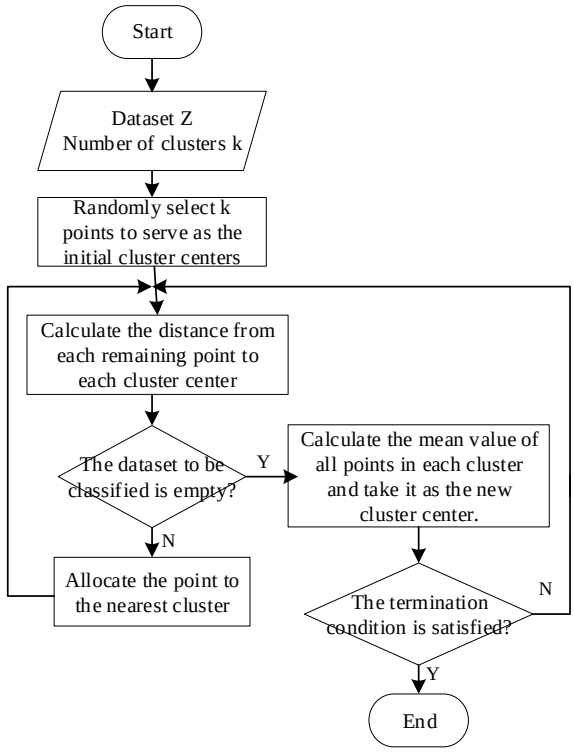

Figure 3. Workflow of the KMC

2.4 The WCK-means text clustering algorithm

The WCK-means text clustering algorithm consists of three parts: Word2vec algorithm, Canopy algorithm and the KMC.

First, the text data were preprocessed, and the parallel Word2vec algorithm was implemented to express each file as the feature word weights by the vector space model. Then, the Canopy algorithm and the KMC were executed in parallel form to cluster the text data. The T1 and T2 values of the Canopy algorithm were set through cross-validation [22, 23]. The Canopy algorithm was called to pre-cluster the preprocessed text data, producing the initial cluster centers. Then, the clustered results were inputted into the KMC to prevent the random selection of the initial cluster centers, and thus improve the speed and effect of the KMC. The clustering process of the WCK-means can be explained as:

Step1: Text representation. Based on the preprocessed unstructured text data, calculate the weight of each word by Word2vec algorithm, and represent each text by the vector space model.

Step2: Center point selection by Canopy algorithm. Input the text dataset and the two distance thresholds T1 and T2 (T1>T2). Randomly select a point from the dataset as the Canopy center, and remove the point from the dataset.

Step 3: Continue to take points from the text dataset, calculate the Euclidean distances from each point P to the center of each existing Canopy by formula (1), and add P to the Canopy with a distance smaller than T1. If a point P is farther than T1 from any of the existing Canopy centers, take this point as a new Canopy center.

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \quad (1)$$

$$E=\sum_{i=1}^{k}(x_i-c_i)^2 \quad (2)$$

Step 4: If the point is less than T2 to a Canopy center, add it to this Canopy and remove it from the dataset. The point is so close to the Canopy that it cannot be the center of any other Canopy.

Step 5: Perform Steps 3 and 4 on the points of the dataset until all the points have been allocated to their corresponding Canopies. Then, terminate the algorithm, remove the relatively small Canopies, and obtain the Canopy centers.

Step 6: The KMC operation

Calculate the distance from each remaining point to each cluster center, and allocate the point to the cluster with the shortest distance.

Step 7: Recalculate the mean value of all points in each cluster and take it as the new cluster center.

Step 8: Repeat Steps 6 and 8 until the change of the cluster center is smaller than the preset threshold or the maximum number of iterations is reached. The judgement criterion can be expressed as:

where, k is the number of clusters; x is the new center of a cluster; C_i is the original center of the cluster.

2.5 Parallel design of the SWCK-means text clustering algorithm

The parallelization of the SWCK-means algorithm is to parallelize the Canopy algorithm and the KMC. In the SWCK-means algorithm, the Word2vec algorithm was selected from the Spark MLlib machine learning database, which is already parallelized. The parallelization of the Canopy algorithm and the KMC was carried out based on the Spark, following the serial logic. The Spark can automatically parallelize the two algorithms by the serial logic. The difference is that our algorithm adopts the resilient distributed dataset (RDD), which automatically realizes parallel data distribution.

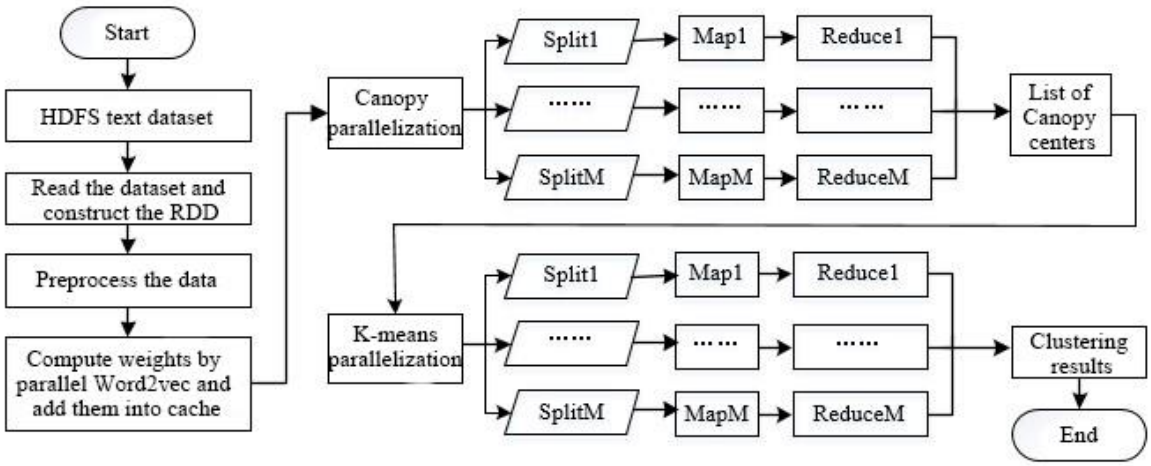

Figure 4. Parallelization design of the SWCK-means text clustering algorithm

As shown in Figure 4 above, the Spark-based parallelization of the Canopy algorithm and the KMC is roughly divided into two parts: Canopy center selection and the final clustering of the KMC. The parallel design can be implemented in the following steps:

Step 1: Read the text dataset from the HDFS and generate the initial RDD.

Step 2: Preprocess the data, convert them into vectors, and save them in cache.

Step 3: Train the Word2vec model in parallel, and express the text as vectors.

Step 4: Execute the parallel Canopy algorithm, segment the RDD, and distribute the segments to each parallel node in the cluster.

Step5: Perform a Map operation to calculate the distances between each text in each segment and the Canopy centers, and determine the local Canopy centers.

Step 6: Perform the Reduce operation to merge the local Canopy centers into the global Canopy center.

Step 7: Each node performs a Map operation according to the global Canopy center, divides the points in the dataset to different Canopies, and save the results into the cache.

Step 8: Remove the relatively small Canopies, and add the remaining Canopy centers to the list of initial cluster centers of the KMC.

Step 9: Each node perform a Map operation on the cached RDD and execute the local KMC.

Step 10: The master control node performs a Reduce operation to merge the local clustering results generated by each node into a global clustering result, and updates the center of each class.

Step 11: Judge if the termination condition is satisfied. If yes, output the result; Otherwise, execute Steps 9-10.

3. EXPERIMENTAL DESIGN AND RESULTS ANALYSIS

3.1 Experimental design

3.1.1 Experimental flow

This paper explores text clustering through Hadoop-based parallel SWCK-means algorithm. First, the unstructured text data were preprocessed through word segmentation, stop words filtering, word frequency analysis, feature selection, and setting up a text representation model. The text data were

quantified into a numerical form for clustering. Next, parallel clustering was performed on the preprocessed data. The text clustering was made more efficient through parallel computing, eliminating the time-consuming iterations of traditional text clustering process. Finally, the Precision of our algorithm, as well as the acceleration ratio and expansibility of parallel computing were verified through experiments. The experimental flow is illustrated in Figure 5 below.

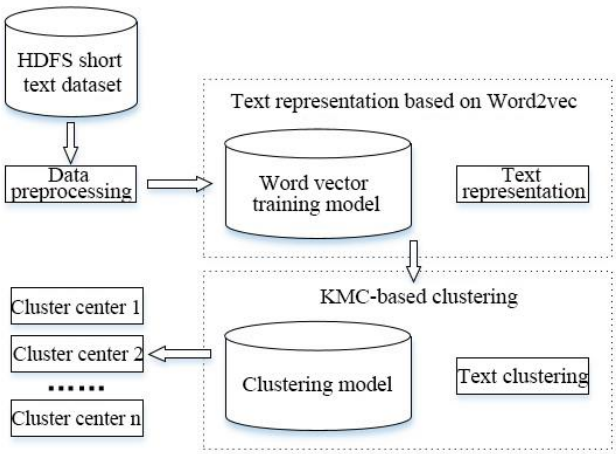

Figure 5. Experimental flow

3.1.2 Data preprocessing

The experiments were carried out on the THUCNews dataset on Sina text clustering corpus (http://thuctc.thunlp.org/). The dataset has been widely used in text classification experiments. The data contained in the THUCNews are filtered results of the historical data from the Sina News RSS subscription channel from 2005 to 2011. The dataset contains 740,000 files on 14 types of news: finance, lottery, real estate, stocks, home design, education, technology, society, fashion, current politics, sports, horoscope, and games. Before clustering analysis, the text data were preprocessed through integration, segmentation and stop words filtering.

(1) Integration. The text data contain multiple small files, which occupy lots of memory in the HDFS. Hence, the multiple small files were combined into a large file, with each row containing the name and content of one small file.

(2) Segmentation and stop words filtering. Before computing the word weights, the integrated data were segmented and removed of the stop words. The HanLP natural language processing package was employed to segment the words and remove meaningless data. The specific steps are introduced below:

1) Removing numbers: Numbers are generally meaningless in text analysis, and were thus removed before further analysis.

2) Removing URLs: The URLs were removed by regular expressions.

3) Filtering stop words: Stop words are defined as the common words that are meaningless in text analysis. In Chinese, words like “is” and “but” fit this definition. The stop words in the preprocessed dataset were filtered out, according to the list of stop words in the news field provided in the HanLP.

4) Removing special symbols: Special symbols like punctuation marks and blank characters are not helpful for text analysis, and were therefore removed.

5) Checking the corpus: The processed corpus was checked again to remove the special URLs by regular expressions, which should have been removed but not removed in the previous steps.

3.1.3 Experimental environment

Before the experiments, the Spark architecture was deployed on Hadoop YARN, a specific component of the open source Hadoop platform for big data analytics. Five virtual machines (VMs) were established based on the VMware, and a Hadoop + Spark cluster platform was created. There are 5 nodes with the same configuration, including a Master node and five Worker nodes (the Master node also serves as a Worker node). Each node has 2G memory and 20G hard disk. The development environment is VMware10.0.4, jdk1.8.0_131, Spark 2.1.1 and Hadoop 2.7.2, and the programming language is Scala 2.11.8.

Three experiments were carried out on the Hadoop + Spark cluster platform, namely, a Precision experiment (Experiment 1), an acceleration ratio experiment (Experiment 2), and an expansibility experiment (Experiment 3). As their names suggest, the three experiments aim to verify the accuracy of the proposed SWCK-means algorithm, as well as the acceleration ratio and expansibility of parallel computing.

3.2 Results analysis

3.2.1 Experimental 1

To verify its effectiveness, the accuracy of the SWCK-means algorithm was tested on four text datasets: Data (200), Data (500), Data (1000) and Data (2000). The four datasets respectively contain 200, 500, 2,000 and 2,000 files in three classes of the Sina text clustering corpus.

The four datasets were respectively clustered and analyzed by the proposed SWCK-means algorithm, the k-means text clustering algorithm based on term frequency-inverse document frequency (TF-IDF) (TK-means), and the means text clustering algorithm based on Word2vec (WK-means).

The clustering effects of the three algorithms were evaluated by Precision and Recall. The Precision is defined as the fraction of relevant instances among the retrieved instances:

$$P = \frac{TP}{TP + FP} \tag{3}$$

where, *TP* (True-positive) means two similar files are correctly categorized into the same cluster; *FP* (False-positive) means two dissimilar files are incorrectly categorized into the same cluster.

Recall is defined as the fraction of the total amount of relevant instances that were actually retrieved:

$$R = \frac{TP}{TP + FN} \tag{4}$$

where, *FN* (False-negative) means two similar files are incorrectly categorized into difference clusters; *TP+FN* is the total number of this type of files.

To fully display the performance of the KMC, it is necessary to set its parameters properly to avoid the local optimum trap. Hence, the feature extraction rate of the TF-IDF was set to 0.025, the word vector dimension of the Word2vec was set to 200, and the minimum number of words was set to 1. Through cross validation, the thresholds of our algorithm were set to T1=0.98 and T2=0.49. The Precision and Recall of each

algorithm were determined by taking the average of 20 runs.

Table 1. The precisions of the three algorithms

Dataset	TK-means / %	WK-means / %	SWCK-means/ %
Data (200)	50.54	63.02	70.75
Data (500)	51.50	64.40	71.60
Data (1,000)	52.29	61.60	71. 20
Data (2,000)	53.50	62.48	69. 24

Table 2. The Recalls of the three algorithms

Dataset	TK-means / %	WK-means / %	SWCK-means/ %
Data (200)	55.13	66.26	71.50
Data (500)	53.35	69.39	74.49
Data (1,000)	54.48	66.95	74. 25
Data (2,000)	57.32	65.14	73. 78

As shown in Table 1, the proposed SWCK-means algorithm is more accurate than the two text clustering algorithms based on traditional KMC. As shown in Table 2, in terms of Recall, our algorithm was 18.8% higher than TK-means and 7.8% higher than WK-means. The result reflects the effectiveness of the SWCK-means in text clustering, thanks to the optimization based on Canopy algorithm.

3.2.2 Experiment 2

The parallelization efficiency of the SWCK-means text clustering algorithm was measured by acceleration ratio and expansibility. Four text datasets were constructed for Experiments 2 and 3. As shown in Table 3, the four datasets respectively contain 2,000, 20,000, 100,000 and 500,000 files from the Sina text clustering corpus.

Table 3. The four datasets

Dataset	File size
Data 1 (2,000 files)	5.5M
Data 2 (20,000 files)	53.1M
Data 3 (100,000 files)	252M
Data 4 (500,000 files)	1.15G

In parallel computing, the acceleration can enhance the algorithm performance by reducing the runtime. The acceleration ratio is an important metric of parallel computing:

$$E_r = \frac{T_s}{T_r} \tag{5}$$

where, T_s is time of serial operation of the algorithm on a single node; T_r is the time of parallel operation of the algorithm on r nodes. The greater the acceleration ratio, the less the time required for the parallel execution of the algorithm, and the higher the efficiency of parallel execution.

In Experiment 2, our algorithm was applied to cluster the four datasets. The acceleration ratio (Figure 6) of our algorithm was measured under each dataset at different number of computing nodes in the Hadoop + Spark cluster environment. As shown in Figure 6, despite the growing number of computing nodes, the acceleration ratio was always close to 1 under Data1, indicating that the acceleration ratio is unobvious for small dataset in cluster environment. By contrast, the acceleration ratio curves under large datasets, Data2, Data3 and Data4, rose significantly; under each of these three datasets, the acceleration ratio increased with the number of computing nodes. The results show that the SWCK-means text clustering algorithm has a good acceleration ratio under the Hadoop + Spark cluster environment.

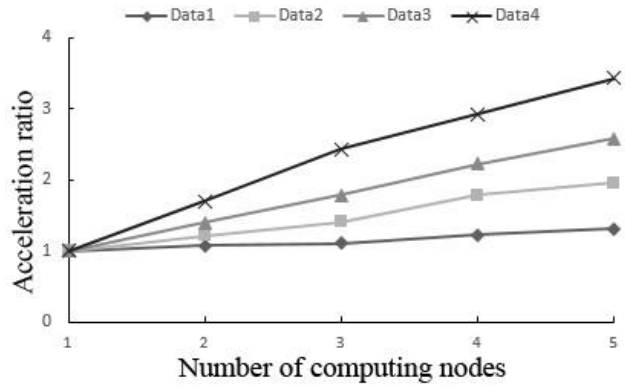

Figure 6. The acceleration ratio of the SWCK-means algorithm

3.2.3 Experiment 3

In parallel computing, the acceleration ratio cannot increase infinitely. When there are many computing nodes in the cluster environment, the utilization efficiency of the computing power in the environment cannot be well demonstrated by acceleration ratio. In this case, the expansibility can be introduced to measure the parallel performance:

$$J = \frac{E_r}{r} \tag{6}$$

where, E_r is the acceleration ratio of the algorithm; r is the number of computing nodes. The expansibility is positively correlated with the overall utilization efficiency of the computing power, and the parallel performance of the algorithm.

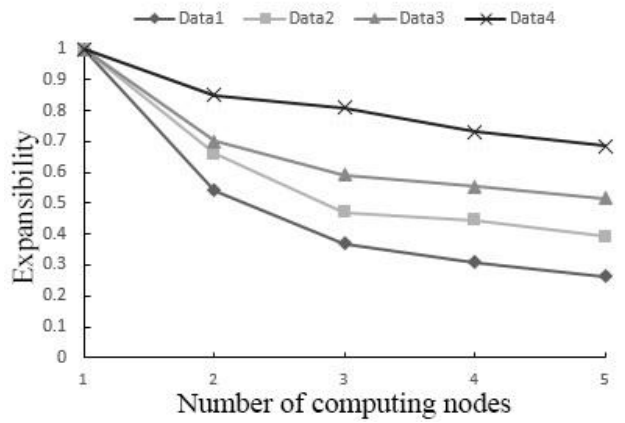

Figure 7. The expansibility of the SWCK-means algorithm

In Experiment 3, our algorithm was applied to cluster the four datasets with different number of computing nodes. The results in Figure 7 show that, with the growth in data volume and the number of computing nodes, the expansibility of our algorithm gradually decreased and slowed stabilized. Comparatively, the expansibility curve of our algorithm declined at the fastest speed under Data1, and at a slightly slower speed under Data2. The curves under Data3 and Data4 decreased very slowly. Hence, the SWCK-means text clustering algorithm boasts excellent expansibility facing large datasets, and slightly poorer expansibility facing small datasets.

4. CONCLUSIONS

This paper probes deep into various techniques for text clustering, including the KMC, Hadoop and Spark big data technique, and proposes the parallel SWCK-means algorithm for text clustering based on the big data platform. Experimental results prove that our algorithm outperforms the traditional KMC in the accuracy and precision text clustering, especially in dealing with massive data. The research results provide a valuable guide for text data mining in real-world scenarios.

ACKNOWLEDGEMENT

This work is supported by the School Fund of Inner Mongolia University of Technology (Grant No.: X201332) for "Parallelization of Clustering Algorithms Based on Hadoop Platform", and the Key Technology Breakthrough Plan of Inner Mongolia Autonomous Region, 2019 for "Design and Application of Big Data Storage and Analysis Mining Platform for Smart Transportation".

REFERENCES

[1] Zhao, T., Liu, B., Li, T. (2017). Research on parallel text classification system based on non-balanced LSH. Microelectronics & Computer, 34(12): 73-79.

[2] Liu, Y., Xu, L., Li, M. (2017). The parallelization of back propagation neural network in MapReduce and spark. International Journal of Parallel Programming, 45(4): 1-20. https://doi.org/10.1007/s10766-016-0401-1

[3] Phu, V.N., Chau, V.T.N., Tran, V.T.N. (2017). SVM for English semantic classification in parallel environment. International Journal of Speech Technology, 20(3): 487-508. https://doi.org/10.1007/s10772-017-9421-5

[4] Štajner, T., Mladenić, D. (2019). Cross-lingual document similarity estimation and dictionary generation with comparable corpora. Knowledge & Information Systems, 58: 729-743. https://doi.org/10.1007/s10115-018-1179-9

[5] Karisani, P., Rahgozar, M., Oroumchian, F. (2016). A query term re-weighting approach using document similarity. Information Processing & Management, 52(3): 478-489. https://doi.org/10.1016/j.ipm.2015.09.002

[6] Mu, Y.K., Feng, S.W., Zhang, J. (2019). Image retrieval based on text and semantic relevance analysis. Computer Engineering and Applications, 55(1): 202-208. https://doi.org/10.3778/j.issn.1002-8331.1709-0209

[7] Macqueen, J. (1965). Some methods for classification and analysis of multi variate observations. Proc of Berkeley Symposium on Mathematical Statistics & Probability, pp. 281-297.

[8] Yang, J., Ma, Y., Zhang, X., Li, S., Zhang, Y. (2017). An initialization method based on hybrid distance for K-Means algorithm. Neural Computation, 29(22): 3094-3117. https://doi.org/10.1162/neco_a_01014

[9] Zhang, J.R., Chai, Y.M., Zan, H.Y., Gao, M.L. (2017). Weakly supervised text classification method based on LDA. Computer Engineering and Design, 38(1):86-91. https://doi.org/10.16208/j.issn1000-7024.2017.01.017

[10] Limwattanapibool, O., Arch-int, S. (2017). Determination of the appropriate parameters for K-means clustering using selection of region clusters based on density DBSCAN (SRCD-DBSCAN). Expert Systems, 34(3): e12204. https://doi.org/10.1111/exsy.12204

[11] Wang, J.R., Ma, X., Duan, G.L. (2019). Improved K-means clustering k-value selection algorithm. Computer Engineering and Applications, 55(8): 33-39. https://doi.org/10.3778/j.issn.1002-8331.1810-0075

[12] Liu, P., Teng, J.Y., Ding, E.J., Meng, L. (2017). Parallel K-means algorithm for massive texts on spark. Journal of Chinese Information Processing, 31(4): 145-153. https://doi.org/10.3969/j.issn.1003-0077.2017.04.021

[13] Al-Refaie, A.F., Yurchenko, S.N., Tennyson, J. (2017). GPU Accelerated in tensities MPI (GAIN-MPI): A new method of computing Einstein-A coefficients. Computer Physics Communications, 214: 216-224. https://doi.org/10.1016/j.cpc.2017.01.013

[14] Xu, Z.W., Feng, B.M., Li, W. (2004). Grid Computing Technology. Beijing: Electronic Industry Press

[15] Dean, J., Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1): 107-113. https://xs.scihub.ltd/https://doi.org/10.1145/1327452.1327492

[16] White, T. (2012). Hadoop: The Definitive Guide. O'Reilly Media, Inc.

[17] Zhang, S.F., Dong, Y.Y., Chen, X.B. (2018). HKM clustering algorithm design and research based on Hadoop platform. Journal of Applied Sciences, 36(3): 524-534. https://doi.org/10.3969/j.issn.0255-8297.2018.02.012

[18] Liao, B., Zhang, T., Yu, J., Guo, B.L., Liu, Y. (2017). Efficiency optimization method for MapReduce similarity computing based on spark. Computer Science, 44(8): 46-53. https://doi.org/10.11896/j.issn.1002-137X.2017.08.009

[19] Wang, C.X., Lv, F., Cui, H.M., Cao, T., Zigman, J., Zhuang, L.J., Feng, X.B. (2018). Heterogeneous memory programming framework based on spark for big data processing. Journal of Computer Research and Development, 55(2): 246-264. https://doi.org/10.7544/issn1000-1239.2018.20170687

[20] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 26: 3111-3119.

[21] McCallum, A., Nigam, K., Ungar, L.H. (2000, August). Efficient clustering of high-dimensional data sets with application to reference matching. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 169-178.

[22] Zhang, L., Mou, X.W. (2018). Chinese text clustering algorithm based on Canopy+K-means. Library Tribune, 38(6): 113-119.
[23] Tong, J.F. (2017). User clustering based on Canopy+K-means algorithm in cloud computing. Journal of Interdisciplinary Mathematics, 20(6-7): 1489-1492. https://doi.org/10.1080/09720502.2017.1386476