Computers, Materials & Continua CMC, vol.62, no.1, pp.217-231, 2020

News Text Topic Clustering Optimized Method Based on TF-IDF Algorithm on Spark

Zhuo Zhou[1], Jiaohua Qin[1, *], Xuyu Xiang[1], Yun Tan[1], Qiang Liu[1] and Neal N. Xiong[2]

Abstract: Due to the slow processing speed of text topic clustering in stand-alone architecture under the background of big data, this paper takes news text as the research object and proposes LDA text topic clustering algorithm based on Spark big data platform. Since the TF-IDF (term frequency-inverse document frequency) algorithm under Spark is irreversible to word mapping, the mapped words indexes cannot be traced back to the original words. In this paper, an optimized method is proposed that TF-IDF under Spark to ensure the text words can be restored. Firstly, the text feature is extracted by the TF-IDF algorithm combined CountVectorizer proposed in this paper, and then the features are inputted to the LDA (Latent Dirichlet Allocation) topic model for training. Finally, the text topic clustering is obtained. Experimental results show that for large data samples, the processing speed of LDA topic model clustering has been improved based Spark. At the same time, compared with the LDA topic model based on word frequency input, the model proposed in this paper has a reduction of perplexity.

Keywords: News text topic clustering, spark platform, countvectorizer algorithm, TF-IDF algorithm, latent dirichlet allocation model.

1 Introduction

Text topic mining and clustering technology have been in existence for a long time. Recently, more and more researches are focusing on text topic clustering technology. The improvement of classical mining algorithm is one of the hotspots of current academic research. Frequently used text processing methods includes retrieval, sentiment analysis, theme extraction, etc. Dumais et al. [Dumais, Furnas and Landauer (1998)] proposed a Latent Semantic Analysis (LSA) algorithm for text Analysis, but it cannot solve the polysemy. Moreover, many iterations are used in the singular value decomposition, so the large-scale data processing cost is huge. The PLSA algorithm proposed by Hofmann [Hofmann (1999)] introduced implicit topic variables. The polysemy problem of the LSA algorithm was solved to some extent. The disadvantage is that the number of parameters of the PLSA increases linearly with the amount of text and the number of words, it is

[1] College of Computer Science and Information Technology, Central South University of Forestry & Technology, Changsha, 410114, China.

[2] Department of Mathematics and Computer Science, Northeastern State University, OK, 74464, USA.

[*] Corresponding Author: Jiaohua Qin. Email: qinjiaohua@163.com.

easy to cause over-fitting in the actual operation. To solve this problem, Blei et al. [Blei, Ng and Jordan (2003)] proposed the Latent Dirichlet Allocation (LDA) algorithm, which introduced control over text set, topic layer and feature word layer with a three-layer Bayesian model. The hyperparameter of the model solved the problem of too many PLSA parameters. The model algorithm has achieved great success in text topic mining and clustering. In recent years, many researchers have done a lot of research on the basis of the LDA algorithm. Yao et al. [Yao, Song and Peng (2011)] applied the LDA model to the SVM framework to model the text topic and achieved a good classification effect. Zhang et al. [Zhang, Sun and Ding (2011)] combined micro blog's social relationship with microblog text and proposed a microblog generation model based on LDA model, which was used to mine the topic of microblog. Based on the LDA topic model and Gibbs algorithm, Wang et al. [Wang, Gao and Chen (2015)] estimated the subject probability distribution of text, which used JS (Jensen-Shannon) distance as the text similarity measure and used hierarchical clustering method for clustering to obtain Higher cluster Purity and Fscore values. Shi et al. [Shi and Cong (2016)] used the maximum correlation minimum redundancy (mRMR) method to filter inactive words, and combined with the LDA topic model to propose the mRMR_LDA algorithm, which improved the accuracy of text clustering. Qu et al. [Qu, Chen and Zheng (2018)] used the LDA topic model to mine the topic, clustered the scientific report documents based on the clustering algorithm of Ward and K-means, and achieved good performance. Backenroth et al. [Backenroth, He, Kiryluk et al. (2018)] applied the LDA topic model to human genetics to predict functional non-coding genetic variation in cell type and tissue-specific manners, with higher precision prediction resolution than predecessors. Zhang et al. [Zhang, Lu and Du (2018)] proposed word merging and LDA topic model (WMF_LDA), which unified the mapping of domain words and synonyms, and filtered the text according to part of speech, and finally modeled the topic. This method reduced the modeling time overhead and improved the clustering speed.

The Spark Big Data platform is based on a distributed memory computing framework which uses Resilient Distributed Datasets (RDD) as the parallel data structure in version 1.X. Although some researchers have proposed optimization for the caching mechanism of RDD, the distributed DataSet introduced by spark2.X, which optimized RDD and provided a powerful distributed computing engine, gradually replaced the previous RDD. Spark's computing framework includes Hadoop's MapReduce framework and has been improved on the basis of Hadoop. The biggest advantage is that it overcomes the problem that Hadoop can only process offline data and has large disk IO overhead. It is faster than Hadoop in data processing speed up to 100 times [http://spark.apache.org/]. As a significant distributed computing system, Spark has been used to solve increasingly complex problems. In recent years, since big data processing platforms such as Hadoop and Spark have emerged, researchers have begun to con-sider parallelizing traditional machine learning algorithms and applying them to big data processing platforms. For example, Zhang et al. [Zhang, Zhang, Zheng et al. (2016] implemented the parallelization of the LDA topic model on Hadoop's MapReduce distributed computing framework, which solved the problem that a single machine could not analyze the hidden topic information in a large-scale corpus. This paper mainly studies the application of LDA theme model in Spark big data processing platform by taking news text as research objects. Firstly, we segment the news text into words by using the Chinese word segmentation tool. Secondly, an optimized TF-IDF algorism is proposed to extract news text features which inputted to Spark-based LDA model. Finally, the experimental results show that the clustering processing speed of the Spark-based LDA topic model is faster than the stand-alone environment. At the same time, compared with the LDA topic model based on the word frequency inputted to LDA, the model perplexity is reduced.

2 Related work

2.1 Preprocessing of news text

News is usually composed of text, pictures, videos, and link information, and the length of each news text is different. Due to the features of strong real-time, a large amount of data, many noise data, and many themes, the news texts are different in performance, and the core themes are annihilated by a large amount of invalid information. Therefore, in the process of extracting the topic model of the news text, it is necessary to filter out the invalid information and improve the quality of the text content to increase the extraction accuracy of the core topic.

The pre-processing of the news text mainly includes the filtering of meaningless texts of the collected news texts, Chinese word segmentation; stop word filtering and single word filtering. The specific steps are shown in Fig. 1.

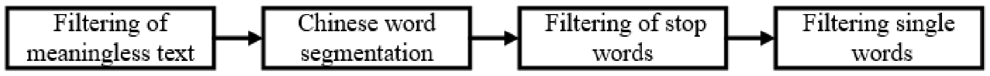


Figure 1: News text preprocessing steps

Filtering of meaningless text: Some news texts should be removed if its number of words is too short to express its topic.

Chinese word segmentation: Different from English, Chinese cannot be separated by words according to blank space, so it is necessary to segment words by first.

Filtering stop words: There are a large number of words in the text that have no practical meaning and cannot highlight specific topics. They only connect sentences or enhance tone. These words are called stop words, such as “oh”, “ah”, “we” and so on. The filtering of stop words not only can reduce the dimension of the word matrix and improve the accuracy of extracting the keywords, but also can greatly reduce the number of calculations and the scale of calculation to improve the efficiency in time. It mainly includes the following three categories:

(1) Modal particle: such as “ah”, “yeah”, “oh”, “and”, “but” and so on. These words only enhance the tone and increase the consistency of the sentence, which are not the influencing factors of the text topic.

(2) Words with low frequency and followed by a single word: the subject of the news text will appear in the text multiple times, some words with very low frequency and single words may appear after the word segmentation. We think this class is not enough to relate to the topic, so it should be filtered.

(3) Specific vocabulary: Different from the two cases above, there are some fixed vocabularies irrelevant to the subject in news texts. For example, there will be “a reporter report”, “a certain network first”, “copyright”, and other news additional vocabulary in the news. Such words should be removed.

2.2 Vector space model

After the word segmentation, a news text becomes a collection of several words, but this still cannot be recognized by the computer. Therefore, it is necessary to use some means to convert these unstructured data into structured data that can be recognized by computers. In text analysis, the vector space model is a widely-used processing method. The vector space model is also called the bag of words model. It maps the weight of the text word into a high-dimensional vector so that the problem of processing the text word is changed to the processing of vector space. In terms of processing word weights, there are word frequency methods, word frequency-anti-document frequency method, and so on.

Definition 1: Suppose there are n pieces of text in the text set D, namely $D = \{d_1, d_2, ..., d_n\}$; after all the text set word segmentation, m words are generated to form the thesaurus W, where $W = \{w_1, w_2, ..., w_m\}$; As shown in Eq. (1), $tf\left(\mathrm{i}, j\right)$ is represented a function F that the frequency at which the word w_i appears in the document d_j ; w_{ij} is the weight of the word w_i in the document d_j .

The word frequency method considers that the word which reflects the subject will repeatedly appear in the text. Based on this idea, the number of times that a word appears in the text is equal to the weight of the corresponding vocabulary.

$$tf\left(\mathrm{i}, j\right) = F\left(\omega_i, \mathrm{d}_j\right) \tag{1}$$

Since the word frequency method only considers a factor of word frequency, it is found that some words with high word frequency are not subject words of the document, so the researchers added the reverse-document frequency factor to the word frequency statistics as the word frequency-reverse-document frequency method. It considers the importance of a word to increase proportionally with the number of times it appears in the file, but at the same time it decreases inversely with the frequency it appears in the corpus. In other words, the more times a word appears in a text and the fewer times it appears in all text collections, the more likely the word is to be the subject word. TF-IDF is one of most widely statistical measurement in text process. For example, Liu Yuling et al. used TF-IDF as keyword weight in verifiable search algorithm for encrypted outsourcing data [Liu, Peng and Wang (2018)]. The TF-IDF calculation is as shown in Eq. (2), where $Num(w_i \in D)$ represents the total number of documents containing the word w_i

$$w_{ij} = tf(i, j) * \log\left(\frac{n}{Num(w_i \in D)} + 1\right) \tag{2}$$

2.3 LDA model

The LDA [Hofmann (1999)] model is a document model technique following the LSA model and the PLSA model. The LDA model not only generates a document based on the topic vocabulary, it is also an unsupervised machine learning technique that recognizes topics in large document sets or corpora.

Definition 2: On the basis of definition one, assume that the news text distribution has k topics $T=(t_1,t_2,...t_k)$.The LDA model considers each text is distributed across topics and each subject is polynomial distributed with a parameter of θ, as shown in Eq. (3).

$$t_i \mid d_i, \theta^{(d_i)} \sim Multi(\theta^{(d_i)}) \tag{3}$$

Each theme consists of a mixture of multiple keywords, each of which also obeys a polynomial distribution with a parameter of φ, as shown in Eq. (4).

$$w_i \mid d_i, \varphi^{(t_i)} \sim Multi(\varphi^{(t_i)}) \tag{4}$$

The parameter θ and the parameter φ respectively obey the Dirichlet distribution with the hyperparameter α and the hyperparameter β, as shown in Eq. (5) and Eq. (6).

$$\theta^{(\mathrm{d}_i)} \sim Dirichlet(\alpha) \tag{5}$$

$$\varphi^{(t_i)} \sim Dirichlet(\beta) \tag{6}$$

The LDA calculates the conditional distribution of the implicit variable under the given observed variable value by using the joint probability distribution. The core part is shown in Eq. (7).

$$p(w|d)=\sum_{i=1}^{k} p(w \mid t_k) * p(t_k \mid d) \tag{7}$$

$p(w|d)$ is the probability that the word w appears under the text d, and $p(w \mid t_k)$ represents the probability that the word w appears on the premise that the topic t_k appears. $p(t_k \mid d)$ indicates the probability that the subject t_k occurs on the premise that the document d appears. Three matrices are shown in Fig. 2: The document-word matrix is the word feature of each text, and the document-word matrix is split into the product of the document-topic matrix and the topic-word matrix, i.e., corresponding to the right side of Eq. (7).

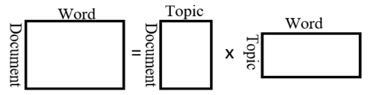


Figure 2: Schematic diagram of the LDA theme mode

2.4 Spark platform

Fig. 3 is the ecosystem structure of Spark. Spark includes the core part and four official sub-modules: Spark SQL, graph calculation module GraphX, machine learning model library MLlib, and real-time stream data processing model Spark Streaming. The sub-module only focuses on the calculation of the data, while the underlying data storage is still carried by Hadoop's Distributed File System (HDFS). Spark MLlib [Meng, Bradley, Yavuz et al. (2016)] is a set of interfaces provided by Spark for distributed machine learning on Spark. It provides a library of most machine learning algorithms such as classification, regression, collaborative filtering, clustering, and underlying optimization.

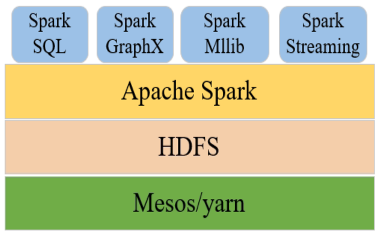


Figure 3: Spark ecosystem structure

As shown in Fig. 4, it is the cluster structure diagram of Spark. This cluster consists of four parts: the client, the driver, the management node Master and the compute node Worker. The client is mainly used for submitting the program to the Driver side, and the Driver creates a directed acyclic scheduler (DAGScheduler) and a task scheduler (TaskScheduler). In addition, the Driver side completes the RDD generation, divides the RDD into a directed acyclic graph, generates a task, and accepts the instruction of the management node Master to send the task to the computing node Worker for execution. The management node Master mainly performs resource scheduling. The compute node Worker is responsible for executing tasks using an executor. The compute node worker keeps in touch with the management node through the heartbeat mechanism, and the worker periodically feeds back the resources and operation status to the master and the management node notifies the driver when to distribute the task to the worker whose resources are idle.

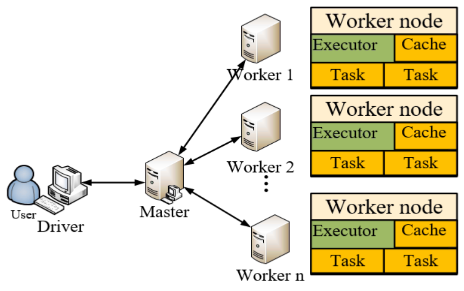


Figure 4: Spark cluster structure diagram

3 LDA topic model clustering based on Spark platform

3.1 Spark distributed processing

Similar to MapReduce in Hadoop, the Spark machine learning system always prioritizes splitting data and distributing it among the compute nodes in the cluster. And each computing node calculates its own data and then aggregates results back to the user, which applies to all calculations in Spark. Fig. 5 is a model for clustering the topic models of news texts in the Spark cluster environment. The user only needs to provide all the news text collections to the Driver side of Spark. The Driver side distributes the data to each compute node according to the preset parallel parameters, and each node will have a subset of the news text. Each compute node processes and computes the news text, which includes the segmentation of the text, the establishment of index of words, the quantization of words and the clustering of topics. The results of the topic models calculated by each compute node will be aggregated to the driver side.

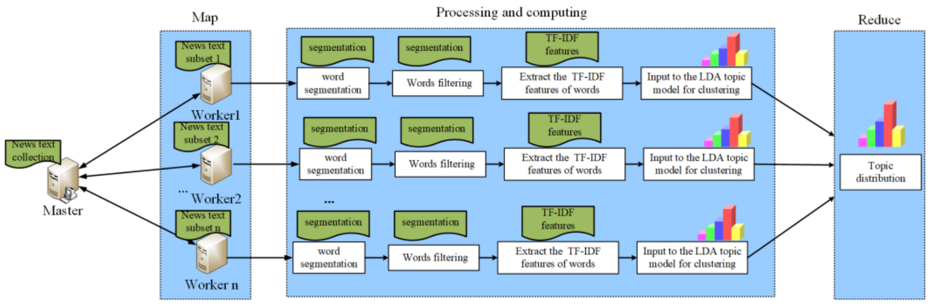


Figure 5: Model of topic model clustering of news text in Spark cluster environment

3.2 Spark EM LDA algorithm based on an optimized algorithm of TF-IDF

In this section, we use the following data and functions to express our algorithm.

Data Statement

texts: all of news text;

sentenceData: A collection of statements of text;

wordsData: The collection of words after the segmentation of all statements;

featurizedData: Vectorization results after TF-IDF training;

Define Function:

Segment (): segment text to sentenceData

Tokenizer():Convert all statements into words

HashingTF: TF-IDF hash algorithm that converts wordsData into TF-IDF features

CVmodel: the instantiation of CountVectorizerModel

In the Spark platform, the TF-IDF algorithm on Spark is shown as Algorithm 1. The first step is that news text will be split into a collection of statements namely *sentenceData*. Step 2 is to initialize a tokenizer. The main function is to recognize the text sentence of the input word segmentation. In the Step 3, *sentenceData* will be converted by *Tokenizer* into the collection of words after the segmentation of all statements namely *wordsData.* In the Step 4, *wordsData* will converted into TF-IDF features by *HashingTF* by specifying the number of hashes. Step 5 is to save the model.

Algorithm 1: TF-IDF algorithm on Spark

```
input: All news texts
output:  words TF-IDF features
1: SentenceData←Segment (text)//segment text to sentenceData
2:  initial tokenizer
3:  wordsData←tokenizer.input(SentenceData)//sentence convert into words
4: featurizedData ←HashingTF.input((wordsData, Hashing_num). transform)
        //Convert to TF-IDF features by entering the number of words and hashes
5:  save the model
```

From the above algorithm, it can be known that the TF-IDF algorithm uses hashmap in the vectorization representation of words, i.e., each word is mapped to a hash index. Since the generation of Hash is irreversible, once the TF-IDF converts the word into the index, the index will not correspond to the original words. Moreover, the text number and text length are different for each text topic clustering, so that the hash number of words to be converted cannot be accurately determined in line 3. Although the TF-IDF algorithm is suitable for clustering text, there is still a lack of tracking of word and adaptive adjustment of the number of word hashes. In this situation, we incorporate the CountVectorizer algorithm in the TF-IDF calculation process. In the training process of the CountVectorizer model, the number of text words will be counted automatically, which avoids the problem that only TF-IDF algorithm needs to specify the hash number of words manually. In addition, the CountVectorizer can save the corresponding relationship between the index of words and the words, avoiding the problem that the index of words in TF-IDF algorithm cannot be traced back to the words. The TF-IDF algorithm combined the CountVectorizer algorithm is shown as Algorithm 2.

Algorithm 2: TF-IDF algorithm combined with CountVectorizer

```
input: All news texts
output: word IDF features, index-word map table
1: SentenceData←Segment (text) //segment text to sentenceData
2: initial tokenizer
3: wordsData←tokenizer.input(SentenceData)
                            //sentence convert into words
4: initial Cvmodel ←CountVectorizerModel
                            //initial CountVectorizerModel
5: wordcount_features ←Cvmodel.fit(wordsData)
                            //Convert words into word frequency features
6: word=Cvmodel.word   //Construct index-word map table
7:initial  idfModel  //initial TF-IDF model
8: featurizedData←idfModel.input ("wordcount_features")
                            //Calculate TF-IDF features
9:  save the model
```

Lines 1 to 3 of the algorithm are consistent with Algorithm 1. The fourth line of the algorithm initializes the CountVectorizerModel and uses it to perform word frequency statistics on the words in the second line. This not only replaces the function of hashing the words in the third line of TF-IDF algorithm on Spark and performing word frequency statistics, but also adds index-words mapping. Line 5 to 8 converts the word frequency feature information generated by the CountVectorizerModel into TF-IDF feature information, and Construct index-word map table.

Spark MLlib provides EM and Online [Hoffman, Bach, and Blei (2010)] to implement the LDA theme model. They use the same data input, but their internal implementation principles are different, as shown in Tab. 1.

Table 1: Two implementations of Spark LDA

Spark LDA model	Method to realize	Parameter prediction method	Storage method
EM LDA	Based on Spark GraphX	Gibbs sampling	Stored on the vertices
Online LDA	Sampling method	Bayesian variational judgment	Stored in a matrix

Since Spark GraphX data storage is a distributed storage structure, this allows EM LDA to do large-scale distributed computing. Words and texts are the two main types of data for LDA, so Spark GraphX constructs word nodes and text nodes to hold words and texts. The word node stores a word and the probability that the word belongs to each topic; the document node stores a document and the probability that the document belongs to each topic. Spark GraphX builds text-word mapping algorithm is shown as Algorithm 3. For each text, when a word appears in a text, GraphX builds an EdgeRDD between the corresponding word node and the text node, meaning that it connects them together to form an edge. For example, as shown in Tab. 2, d_1 and d_2 represent two news texts, w_1, w_2, w_3 and w_4 represent four words, and the data in the table indicates the word frequency of the words in the corresponding news text. Spark EM LDA builds a GraphX graph based on the relationship between words and text, as shown in Fig. 6. After constructing the graph vertex-edge data, Spark EM LDA iteratively calculates the data according to the preset global hyper parameters.

Algorithm 3: Spark GraphX builds text-word mapping algorithm

```
Parameter Description:
$n$ : Total number of text
$m$ : Total number of word after word segmentation
Matrix: Word frequency matrix
1: for document $i$ from $[0, n-1]$:
2:    for word $j$ in document $i$, $j$ from $[1, m]$:
3:       generate an edge $(i, j, Matrix[i][j])$ as an element of EdgeRDD
4:    end for
5: end for
```

Table 2: Spark GraphX builds text-word relationship examples

	w_1	w_2	w_3	w_4	…
d_1	2	0	4	3	…
d_2	0	2	5	3	…
…	…	…	…	…	…

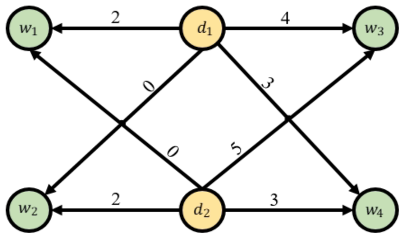


Figure 6: Text-word diagram

4 Experiment and result analysis

4.1 Dataset

The data set from Sogou Labs provides Sohu news data [https://www.sogou.com/labs/resource/cs.php]. The news content is from the news data of 18 channels of domestic, international, sports, social, and entertainment and so on from June to July 2012, about 2 GB after decompression. The data set word segmentation tool uses the open source Hanlp [http://hanlp.linrunsoft.com/]. Hanlp is a Java toolkit composed of a series of models and algorithms. It provides complete functions such as word segmentation, lexical analysis, syntactic analysis, semantic understanding and so on. Its underlying algorithm has been carefully optimized, and it can reach 20 million words per second under the fast word segmentation mode.

4.2 System environment

The experiment uses Alibaba Cloud's ECS elastic cloud server, installing Spark standalone to a Cluster, and the hardware, software and system configuration information are shown in Tab. 3.

Table 3: Experimental hardware and software configuration information

Nodes	CPU	Memory	Operating System	Spark version
Mater (1 set)	8 cores	32 GB	CentOS7.2	Spark2.3.0
Worker (5 set)	8 cores	32 GB	CentOS7.2	Spark2.3.0

4.2 Experimental parameters and evaluation indicators

The parameters in the experiment mainly including the number of cluster topics k, Dirichlet parameters α and β, max Iterations and the Spark cluster checkpoint interval, are followed:

k: The number of cluster centers, because the topic cluster belongs to unsupervised clustering, there is no absolute judgment standard for this parameter setting. It can be within a reasonable range.

α: The docConcentration in Spark, refers to the a priori parameter of the document distributed on the subject. In Spark, the parameter must be greater than 1. When the amount of data is small, the parameter has a great influence on the clustering. As the data scale increases, the influence of the parameter on the clustering effect is gradually reduced.

β: The topicConcentration in Spark, the priori distribution parameter of the subject on the word. This parameter must be greater than 1 in Spark. Its influence on clustering is the same as the Dirichlet parameter α.

maxIterations: The maximum number of iterations of the EM algorithm. The maximum number of iterations depends on the size of the data set. When data is large, the number of iterations is recommended to be between 50 and 100 times.

Checkpoint: In Spark, when the number of iterations is large, the checkpoint interval can help reduce the shuffle file size in Spark and help recover faults. We use perplexity measurement as an experimental indicator. Perplexity is a kind of entropy of information, which is a measure of the quality of the model in natural language processing. It is the uncertainty of a probabilistic model, in other words, the lower the perplexity, the better the model. It is shown in Eq. (8) specifically.

$$Perplexity = \exp\left(-\frac{\sum_{i=1}^{M} \log p(w_i)}{\sum_{i=1}^{M} N_i} \right) \tag{8}$$

where M represents the number of texts in the test news text set N_i presents the number of i-th text words, and $p(w_i)$ represents the probability of the text.

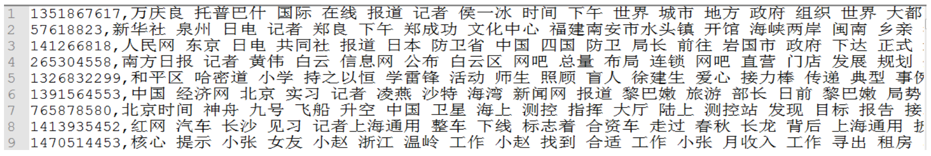
4.2 Experimental process and results analysis

This paper consists of two groups of experiments including seven sub-experiments. The first group experiments included Experiment 1 and Experiment 2, which are used to compare the advantages and disadvantages of the proposed algorithm with the traditional word frequency-based LDA algorithm. The second group experiments included Experiments 2 through 7 to compare the clustering time of the algorithm in a single-machine architecture and cluster-based case under different scale datasets. The data sets are filtered from the Sogou data set. The experimental design is shown in Tab. 4.

Table 4: Experimental design

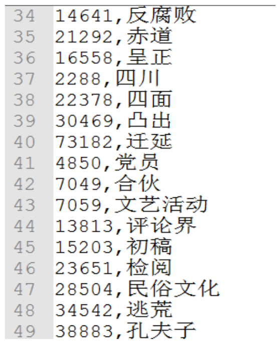
Experimental	Number of data sets	Experimental platform and algorithm
1	10000 news data	word frequency based LDA algorithm under stand-alone architecture
2	10000 news data	This article LDA algorithm in single machine architecture
3	10000 news data	This article LDA algorithm based on Spark cluster
4	50000 news data	This article LDA algorithm in single machine architecture
5	50000 news data	This article LDA algorithm based on Spark cluster
6	100000 news data	This article LDA algorithm in single machine architecture
7	100000 news data	This article LDA algorithm based on Spark cluster

The results of the experimental set after the Hanlp word segmentation algorithm are shown in Fig. 7. Each text is divided into one line and attached with a text label. Fig. 8 is an index-word mapping relationship table obtained by combining the TF-IDF algorithm of CountVectorizer, where in (a) is a local index-word mapping calculated in a stand-alone environment, and (b) is a distributed index-word mapping calculated on a Spark cluster.



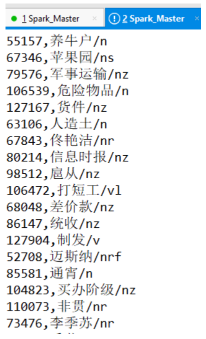
```
1351867617,万庆良 托普巴什 国际 在线 报道 记者 侯一冰 时间 下午 世界 城市 地方 政府 组织 世界 大都
57618823,新华社 泉州 日电 记者 郑良 下午 郑成功 文化中心 福建南安市水头镇 开馆 海峡两岸 闽南 乡亲
141266818,人民网 东京 日电 共同社 报道 日本 防卫省 中国 四国 防卫 局长 前往 岩国市 政府 下达 正式
265304558,南方日报 记者 黄伟 白云 信息网 公布 白云区 网吧 总量 布局 连锁 网吧 直营 门店 发展 规划
1326832299,和平区 哈密道 小学 持之以恒 学雷锋 活动 师生 照顾 盲人 徐建生 爱心 接力棒 传递 典型 事
1391564553,中国 经济网 北京 实习 记者 凌燕 沙特 海湾 新闻网 报道 黎巴嫩 旅游 部长 日前 黎巴嫩 局势
765878580,北京时间 神舟 九号 飞船 升空 中国 卫星 海上 测控 指挥 大厅 陆上 测控站 发现 目标 报告 接
1413935452,红网 汽车 长沙 见习 记者上海通用 整车 下线 标志着 合资车 走过 春秋 长龙 背后 上海通用
1470514453,核心 提示 小张 女友 小赵 浙江 温岭 工作 小赵 找到 合适 工作 小张 月收入 工作 寻出 租房
```

Figure 7: Experiment on news text segmentation results



```
14641,反腐败
21292,赤道
16558,呈正
2288,四川
22378,四面
30469,凸出
73182,迁延
4850,党员
7049,合伙
7059,文艺活动
13813,评论界
15203,初稿
23651,检阅
28504,民俗文化
34542,逃荒
38883,孔夫子
```

(a)



```
55157,养牛户/n
67346,苹果园/ns
79576,军事运输/nz
106539,危险物品/n
127167,货件/nz
63106,人造土/n
67843,佟艳洁/nr
80214,信息时报/nz
98512,扈从/nz
106472,打短工/vl
68048,差价款/nz
86147,统收/nz
127904,制发/v
52708,迈斯纳/nrf
85581,通宵/n
104823,买办阶级/nz
110073,非贯/nr
73476,李季苏/nr
```

(b)

Figure 8: Index-word mapping

Text words will be vectorized when segmenting words and building the words. The LDA can calculate the topic word under each topic and the topic distribution information of each news text by receiving the word frequency feature or the TF-IDF feature information. As shown in Tab. 5, the weight information of the first 4 core keywords of each topic in the case of the TF-IDF feature input, the number of topics k=8, α=7, and β=2. The schematic diagrams show the distributions of topics for randomly extracting 8 pieces of news in this case in Fig. 9.

Table 5: Topic-Subject word-Topic Distribution

Topic	Subject word	Subject weight
0	[164, 116, 6, 35]	[0.0045335, 0.0042228, 0.0037616, 0.0037323]
1	[44, 72, 133, 151]	[0.0036255, 0.0035607, 0.0026286, 0.0025112]
2	[8, 43, 60, 171]	[0.0085596, 0.0048892, 0.0041579, 0.0029936]
3	[9, 0, 51, 211]	[0.0127042, 0.0050826, 0.0031506, 0.0029446]
4	[14, 16, 13, 37]	[0.0052911, 0.0041124, 0.0038823, 0.0037967]
5	[318, 338, 630, 796]	[0.0037961, 0.0023438, 0.0022488, 0.0019364]
6	[0, 18, 20, 15]	[0.0086184, 0.0056217, 0.0047544, 0.0046929]
7	[1, 3, 12, 5]	[0.0249111, 0.0209015, 0.0130717, 0.0116296]

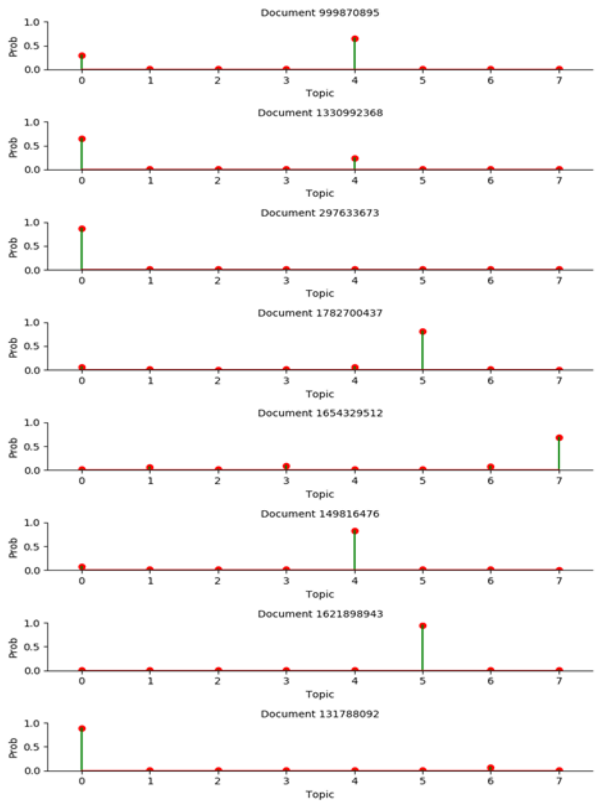


Figure 9: News text topic distribution

The first set of experiments take 10,000 news texts, the number of topics is 8 and the number of iterations is 20. By comparing the LDA topic model algorithm of word frequency feature input and the results of the algorithm, it is found that the algorithm is obviously higher in the similarity degree, and the algorithm is also reducing in the confusion degree, as shown in Tab. 6.

Table 6: Perplexity comparison

Algorithm	Log (perplexity)
Word frequency based LDA	26.811657551355566
This article LDA algorithm	21.777937111042842

For the second group of experiments, when the amount of data is 10000, 50000, 100000, and the number of subject clusters is 8, we compared the clustering time in the stand-alone environment and the Spark cluster. The specific experimental parameters and results are shown in Tab 7. It can be seen from the table that when the data volume is 10000, the clustering time of the topic in the cluster environment is longer than that of the single-machine cluster. This is due to the communication be-tween the nodes in the cluster spends a lot of time. When the data volume is 50,000, the topic clustering time in the cluster environment is approximately equivalent to the single-machine topic clustering time, and when the data volume rises to 100000, the topic clustering time in the cluster environment is significantly lower than that in the standalone environment.

Table 7: Cluster time results under different scale data

Data size	Data block	α	β	MaxIterations	Platform	Clustering time (second)
10000	8	7	2	20	Single machine	25.61
10000	8	7	2	20	Spark cluster	39.32
50000	32	7	2	30	Single machine	92.31
50000	32	7	2	30	Spark cluster	94.35
100000	64	7	2	60	Single machine	436.76
100000	64	7	2	60	Spark cluster	292.65

5 Conclusion

In this paper, we propose to take TF-IDF features as input to the LDA topic model and integrate the CountVectorizer algorithm into the TF-IDF algorithm on Spark, which optimize the irreversible problem when words are converted into vectors. In terms of clustering effect, compared with LDA topic clustering based on word frequency feature input, the perplexity of LDA topic model in this paper is significantly reduced. Moreover, through testing different scale dataset on the single machine and Spark distributed cluster, it concludes that although the clustering speed of small sample data in the single machine environment is faster than that in Spark distributed cluster environment, the processing speed of a single machine is slow with the increase of data scale. The processing speed under the Spark distributed cluster is significantly improved. However, the LDA algorithm based on Spark distributed cluster is not perfect either. Since Spark is memory-based distributed computing platform, it will consume a large number of memory resources when the amount of data is doubled. In the future work, the optimization of Spark cluster resources will be further studied.

Acknowledgment: This work is supported by the Science Research Projects of Hunan Provincial Education Department (Nos. 18A174, 18C0262), the National Natural Science Foundation of China (No. 61772561), the Key Research & Development Plan of Hunan Province (Nos. 2018NK2012, 2019SK2022),the Degree & Postgraduate Education Reform Project of Hunan Province (No. 209), and the Postgraduate Education and Teaching Reform Project of Central South Forestry University (No. 2019JG013).

References

Backenroth, D.; He, Z.; Kiryluk, K.; Boeva, V.; Pethukova, L. et al. (2018) FUN-LDA: a latent dirichlet allocation model for predicting tissue-specific functional effects of noncoding variation. *Methods and Applications. American Journal of Human Genetics*, vol. 102, no. 5, pp. 920-942.

Blei, D.; Ng, A.; Jordan, M. (2003): Latent dirichlet allocation. *Journal of Machine Learning Research*, vol. 3, no. 5, pp. 993-1022.

Dumais, S. T.; Furnas, G.; Landauer, T.; Deerwester, S. (1998): Using latent semantic analysis to improve information retrieval. *Conference on Human Factors in Computing*, pp. 281-285.

HanLP v1.2.8 Link (2019): http://hanlp.linrunsoft.com/.

Hofmann, T. (1999): Probabilistic latent semantic indexing. *Fifteenth Conference on Uncertainty in Artificial Intelligence*, vol. 51, no. 2, pp. 50-57.

Hoffman, M.; Bach, F.; Blei, D. (2010): Online learning for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, vol. 23, pp. 856-864.

Shi, Q. W.; Cong, S. Y. (2016): Text classification based on mRMR and LDA topic model. *Computer Engineering and Applications*, vol. 52, no. 5, pp. 127-133.

Liu, Y. L.; Peng, H.; Wang, J. (2018): Verifiable diversity ranking search over encrypted outsourced data. *Computers, Materials & Continua*, vol. 55, no. 1, pp. 37-57.

Liu, X. W.; Zhu, X.; Li, M.; Wang, L.; Tang, C. et al. (2018): Late fusion incomplete multi-view clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10.

Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S. et al. (2016): MLlib: machine learning in apache spark. *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1-7.

Qu, J. Y.; Chen, Z.; Zheng, Y. N. (2018): Research on clustering method of science and technology report documents based on topic mode. *Library and Information Service*, vol. 4, pp. 113-120.

Shi, Q. W.; Cong, S. Y. (2016): Text classification based on mRMR and LDA topic model. *Computer Engineering and Applications*, vol. 52, no. 5, pp. 127-133.

Sougou Dataset Link (2019): https://www.sogou.com/labs/resource/cs.php.

Spark Homepage (2019): http://spark.apache.org/.

Wang, P.; Gao, W.; Chen, X. M. (2015): Research on text clustering based on LDA model. *Information Science*, vol. 1, pp. 63-68.

Yao, Q. Z.; Song, Z. L.; Peng, C. (2011): Text classification based on LDA model. *Computer Engineering and Applications*, vol. 47, no. 13, pp. 150-153.

Zhang, C. Y.; Sun, J. B.; Ding, Y. Q. (2011): Microblogging topic mining based on MB-LDA model. *Journal of Computer Research and Development*, vol. 48, no. 10, pp. 1795-1802.

Zhang, L.; Lu, T. L.; Du, Y. H. (2019) Text similarity calculation based on WMF_LDA theme model. http://www.arocmag.com/article/02-2019-10-028.html.

Zhang, Z.; Zhang, X. F.; Zheng, N.; Gui, M. J. (2016): Parallelization of LDA algorithm based on Hadoop platform. *Computer Engineering and Science*, vol. 38, no. 2, pp. 231-239.