News Clustering Using Spark

Jaimin Shah
Student, Department of CSE, Nirma University, Ahmedabad, India

***Abstract:* - Apache spark is well known tool for big data analytics. It is preferred to carry out data mining task on a very big dataset on distributed environment. It supports distributed computation (clusters) due which we can achieve parallelism for computation among the node in the cluster. Number of machines in the cluster is a major factor while setting up the cluster. More number of machines means you will get faster results. Major advantage of using spark is that, it supports variety of languages like Java, Python, Scala etc. This articles focuses on clustering of text data. As clustering of text data is very different than clustering of numeric data. Text clustering has many challenges like sparsity of data which we must consider while carrying out clustering. Another major challenge is building word vocabulary from dataset. All major steps which are needed to carry out text clustering are described in detail in this article.**

***Keywords:* - News clustering, Spark, LDA (Latent Dirichlet Allocation), Kmeans**

I. Introduction

In today's era we have plenty of data available with use. We can use some data mining techniques to extract as much information as we can from that data. To perform data analysis on that huge amount of data selecting a right tool is very much important. There are plenty of tools available when it comes to analyzing the data and extracting useful information from it. We have huge choice of tools with us to select from like weka, r, hadoop, spark. When we talk about big data the most famous tool is hadoop.

Spark is new competitor for hadoop. Spark is big data analytic tool which is based on hadoop. But spark is optimized for iterative computation which is used frequently in most of data mining algorithms. In one of the experiments researchers carried out an experiment of sorting 100TB of data and compared performance of both spark and hadoop. The cluster configuration used to carry out experiment is as stated below [7]

	Time (minutes)	Machines	Sorting rate(TB/min)
Spark	23	206	4.27
Hadoop	72	2100	1.42

As per statistics, researchers found that spark managed to complete the computation 3 times faster than hadoop with (1/10)th of machines than Hadoop. Even disk I/O operations for spark were (1/4)th of Hadoop.[7] After this experiment many people associated with data mining field termed spark as a successor of Hadoop. Even many companies are adopting Spark as their core tool for big data. As of now, spark is the fastest tool available for big data analysis. Here, it is explained how to do text clustering using Spark using Java as programming language.

II. Dataset

The dataset used for this experiment is BBC dataset. The raw version of the dataset is used for clustering. This dataset has 2225 documents related to different categories. These recent articles are of year 2004-2005. There are five categories of news articles in this dataset. The categories of the articles are business, entertainment, politics, sport, technology. Each text file represents the content of one article. So there are total 2225 text files.

Each text file is a raw text file which is procured by BBC. It is in original format, so no preprocessing steps are carried on that. So the first task which is very important to get accurate results is to carry out preprocessing to remove noise and unrelated terms from the text file.

There are various techniques for removing the noise form data. It is very important step because if dataset contains high noise, then it will affect accuracy of the clustering significantly.

III. Preprocessing

As described earlier, preprocessing is the most important step for any data mining task. The different preprocessing techniques used for clustering are low frequency filter, stop word removal and stemming. First step for preprocessing is to find all unique words used in dataset. We have to consider only those words which would help to identify the type of document. We ignored rest of the words. For that we have to prepare wordlist for the terms which helps to find out the topic. If we consider all words used in dataset then result will be highly inaccurate.

1. *Find all unique words:-*

Before removing noise from the dataset, first we have to identify all unique words and their frequencies in the dataset. For this word count, implementation works well. So after this step, we will get a file which will contain two columns having word and its frequency count. This file is very important for building the final vocabulary list.

2. *Removing stop words:-*

Removing stop words is very important as stop words are words like 'a', 'the' which don't have any significance in the meaning of the sentence. This words don't help for determining the topic of the document. In some cases, it can affect the accuracy as well.

For example, the cluster with high instances will have more frequency of the stop words so in many cases new instance will match that cluster only. So due to this, to get accurate results, we would have to remove this kind of words from the word count list. Also, removing this words will simplify the clustering process.

3. *Filtering less frequency words:-*

We must remove the words with less than some predefined frequency. Minimum frequency must be decided very carefully. If we set minimum frequency to high then we may miss out some important words, which help in determining the topic of the document. And if we set it very low, then it will include some unnecessary words like name of person, place etc., which are useless in determining the topic of the document. For this experiment, we set the minimum frequency count to 3. So we have to remove all the words which have frequency count less than 3.

4. *Stemming algorithm:-*

Next step is to apply stemming algorithm in the frequency count file. Stemming algorithm normalizes all words to its root, for example deal, dealing, dealer will be converted to the word deal. It will reduce the complexity of the clustering significantly.

After carrying out this filtering process, we have final vocabulary file in which we used three preprocessing techniques stop word removal, stemming and filtering out the words with low frequency count. Only the words which are in this list will be considered for carrying out clustering all other words in the dataset, will be ignored. The final vocabulary which is used in this experiment has 10,000 words in it.

IV. PREPARING CORPUS

Next step is to carry out transformation technique on the dataset. As we can't perform clustering on the text data directly, we have to carry out transformation to convert text data to numeric data. For this, we have to prepare corpus from the dataset. The corpus is a kind of a matrix in which each column represents particular word and each row represents particular document. For example, the element at [i,j] represents the frequency of the word associated in column j in document no i.

As we have vocabulary of 10000 words, there are 10000 columns in the matrix (corpus). For preparing the corpus first, we have to map each word of vocabulary with a unique column number.

Then, we have to scan document sequentially. For each document, we have to map it with a row number. For each word in the document, we have to apply stemming algorithm on it and then check whether that word is present in our vocabulary or not. If a word is there in our vocabulary, we have to increase count at the position of respective row and column. Otherwise, if that word is not in our vocabulary, then we will just ignore that word.

After scanning the whole dataset, we have a corpus with 2225 rows where each row represents a particular document. There are total 10000 columns in the corpus. So, after completion of this transformation technique we have final corpus which is a matrix of dimension [2225 , 10000].

So, now dataset is transformed from text to numeric values. So, we have Document Id which indicates row number in the corpus and frequencies of all the words in that document which is stored at respective Column Id which we obtained from mapping word and column id.

This is most the important step for preparing the dataset for clustering. As currently no algorithm supports clustering of raw text data that is why the transformation technique is very important for any algorithm. Most of the algorithm take numeric input, so after applying a transformation technique, we have variety of algorithms to choose from we can choose the most accurate algorithm. In this experiment, we will compare the results of two clustering algorithms K-means and LDA model.

V. ALGORITHM SELECTION

Algorithm selection is very important part of clustering. For results, we get heavily dependent on the algorithm which we have selected to carry out clustering process. Some algorithms work good for time variant data while some work good for time invariant data. Same way, some algorithm doesn't give good results for sparse data. So, we have to select the algorithm very carefully, after observing the general characteristics and properties of the dataset. The comparison of two different algorithms K-means and Latent Dirichlet allocation (LDA) are described here. Input parameters for both algorithms are maximum number of iterations, number of clusters in corpus as an input parameter.

VI. KMEANS

K-means is the most basic and most widely used algorithm. K-means takes maximum number of iterations, number of clusters to be created and corpus as an input. K-means works by measuring distance from centroid of each cluster. Instance is assigned to a cluster which has the least distance from it. For measuring distance from the cluster, different distance measures like Euclidean distance , Manhattan distance , Minkowski distance etc. are available.

In this experiment, Euclidean distance measure is used. Optimally, the algorithm terminates when no change in cluster assignment is made during an iteration. But, it might not be possible practically. So, we will have to set maximum number of iterations after which algorithm terminates. Setting high value for this, will give more accurate result, but on the other hand computation time would also increase. We know that dataset is categorized into five topics so we set required number of clusters to 5 for optimal results.

K-means model for this dataset doesn't give good results for this dataset. It creates five clusters out of which two clusters have less than 10 instances in each. Reason of this low accuracy is the sparsity in the dataset. We have a vocabulary of 10,000 words but average size of these news articles is just 1000-2000 words. And many words are not even present in the vocabulary, so the effective size of news article is relatively very less compared to size of the vocabulary. So K-means algorithm doesn't give good result when the data is very sparse. In this dataset there are ten thousand dimensions which is considered as a major factor that affects the results of this algorithm.

If we want better results with same algorithm then we have to apply dimensionality reduction principle on the dataset to reduce the dimensions. Reducing a dimension is a complex task and moreover, in most of the cases there is loss of information associated when we reduce dimension of dataset.

Other alternative we have, is applying some another algorithm which handles sparsity of data very well. In this option, there would not be any loss of information as we are not making any changes in the original dataset and moreover, we are not dropping any of the dimension. Some algorithms which handles sparse dataset well are Latent Dirichlet allocation (LDA), Distribution of the Mean Estimate, (Reminder) Multinomial Distribution.

VII. Latent Dirichlet allocation (LDA)

Latent Dirichlet allocation (LDA) is a probabilistic model which can handle sparse data well. Its input parameters are same as K-means. It gives the probability of belonging to each topic or cluster. The working of LDA is similar to probabilistic analysis. Initially, topic is assigned to each word of the vocabulary on random basis. Then, for each document in corpus and then for each word in that document, we find two probabilities:

I. P(topic(t)/document) denotes the document probability that it belongs to particular topic(t) depending on the number of words on document which are currently assigned to topic 't'.

II. P(word w/ topic t) for a given topic t denotes the probability that word 'w' is there. It denotes the probability that word 'w' is there in a document which is assigned to topic t.

Multiplication of this probabilities are found for all combination of document, word in vocabulary and topic.The word is assigned to the topic which has highest multiplicative values of the above mentioned two probabilities. This whole process is performed in single iteration.

Optimally this algorithm is terminated when no change in the assignment of word is observed during the entire iteration. Though, after performing large number of iterations, word assignments become quite stable so practically we will stop execution after any large number of iterations.

On termination, we have probability values for each word to be in each topic. This algorithm can handle the sparsity of data because it doesn't use any distance measure. Mostly the algorithms which use distance measure for assigning an instance to a particular cluster doesn't perform well for sparse data.

We have probability of the word for each topic so when new document is added in dataset for clustering we can compute the probability of each of the topic and assign it the topic which has the highest probability. This method of assigning values is always accurate because the probability of words in the new document belonging to the actual topic would be always higher than all other topics.

In this dataset we have five topics (news category) sports, entertainment, business, politics and technology. Now, for example if we get new document of sports category than topic describing words used in that document would be sports, game, player, strike etc. The probability of these words belonging to the sport cluster would be higher than any other cluster.

So when we find overall portability of entire document belonging to the sport cluster it would always be higher than any other cluster. Because of the probability of subparts (words of document) to that document belonging to the sport cluster is higher. Same way for technology related cluster probability of the words associated with technology would be higher than all other clusters.

This way we can also use LDA model to determine topic of the new document. This model has only one major disadvantage and that is, you have to give number of topic as input parameter if you don't know actual categories of data in your dataset, then it becomes very difficult to use LDA (Latent Dirichlet allocation) for clustering and in that case number of clusters has to be selected on the basis of trial and error.

So, you have to observe the results manually each time. It makes it hectic task to find the best number of clusters for your dataset. Though, currently LDA is the most advance and preferred method to perform clustering of data in text format.

VIII. CONCLUSION

Currently LDA is the most accurate model to text clustering. It handles major problem in clustering the textual data like sparsity, which conventional algorithms like K-Means can't handle well. It gives highly accurate results compared to conventional algorithms like K-Means.

Moreover, all major tools have added or are in process of adding support for LDA model due to its accuracy. It is a very good alternative to conventional way of text clustering which is to apply dimensionality reduction on the dataset. The results of LDA are even more accurate because there is no information loss when we use LDA model for clustering.

REFERENCES

[1] Data Mining: Concepts and Techniques by Han by Han (Author)
[2] Data Mining: Introductory and Advanced Topics by Margaret H. Dunham
[3] Introduction to Data Mining by Pang - Ning Tan
[4] Evan R Sparks, Ameet Talwalkar, Michael J. Franklin, Michael I. Jordan, and Tim Kraska. TuPAQ: An efficient planner for large-scale predictive analytic queries. arXiv:1502.00068, 2015.
[5] MLlib: Machine Learning in Apache Spark
[6] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, Ameet Talwalkar arXiv:1505.06807, 2015.
[7] http://spark.apache.org/news/spark-wins-daytona-gray-sort-100tb-benchmark.html