ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X

International Journal of Advanced Research in Basic Engineering Sciences and Technology (IJARBEST)
Volume 2, Special Issue 19, October 2016

Data Mining using HADOOP on Bio-NER

M.Usharani
M.Phil Research Scholar
Vellalar College for Women (Autonomous)
Erode India
usharanimanick@gmail.com

Mrs. P. Anitha, M.C.A., M.Phil., B.Ed.,
Department of Computer Applications
Vellalar College for Women (Autonomous)
Erode India
anithavcw@gmail.com

Abstract

Biomedical named entity recognition (Bio-NER) is the critical step in text mining, where the data redundancy and performance of processing huge data is the challenging issue. Conditional Random Field is the conditional probability model used to overcome traditional FP-tree algorithm challenges, even in CRF achieving better performance is nontrivial due to internal sequential process. Here parallelism is introduced by combining and parallelizing the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) and Viterbi algorithms called parallel CRF or MRCRF (MapReduce CRF). The MRLB (Map Reduce LBFGS) algorithm and MRVtb (MapReduce Viterbi) algorithm enhance the parameter estimation and no data redundancy. MRCRF algorithm exhibits better performance improvement and information accuracy compared to traditional systems. Additionally the new IMRCRF (Improved Map Reduce CRF) shows better performance in terms of processing huge data from several nodes.

Keywords— Biomedical Named Entity Recognition, Conditional Random Fields, Map Reduce.

I.INTRODUCTION

In the 21st century, it is increasingly inseparable from the network, people visit dozens or even hundreds of pages, or upload photos or speech every day, which makes the data content on the network into a geometric growth, and the traditional technical architecture has become increasingly unable to meet the current needs of the vast amounts of data. Therefore, researching massive data processing and storage become more and more popular nowadays. Big data is a large data that it becomes difficult to process the conventional database systems. If the data is very large, moves very fast, or doesn't fit the structures of the database architectures. To gain value from this data, choose another way to process the data. Big Data in general is defined as high volume, velocity and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making. Big Data is the frontier of the firm's ability to store, process and access large volume of data it needs to operate effectively, make decisions, reduce risks, and serve customers. However, the amount of data generated can often be very large for a single computer to process in a reasonable amount of time. Furthermore, the data itself may be too big to store on a single machine. Therefore, in order to reduce the time taken to process the data, and to allocate the storage space for large files, it is necessary to write programs that can execute on multiple computers and distribute the workload among them.

II.HADOOP

Hadoop is the foundation for most big data architecture. Apache hadoop is an open source java programming framework for fast storing and fast processing large data sets with cluster of commodity hardware. Cluster is a set of machine in single LAN (Local Area Network). The Hadoop is mainly constituted by the underlying distributed file system HDFS (Hadoop Distributed File System) and MapReduce layer of parallel programming model engine. Hadoop is used by various universities and companies like Google, eBay, Facebook, IBM, LinkedIn and Twitter.

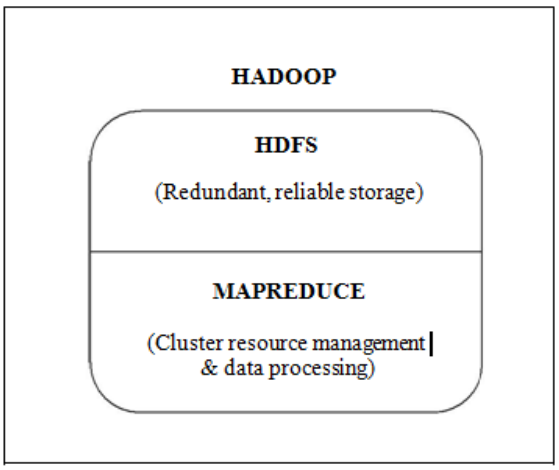

Fig. 1.HDFS and MapReduce

A. HDFS

HDFS is a reliable distributed file system that provides high-throughput and scalable access to data. MapReduce is a distributed framework for executing the work in parallel. Hadoop has the master/slave architecture for both processing and storage. Figure 1 shows the HDFS and MapReduce. HDFS is a specially designed file system for storing massive amount of data sets with cluster of commodity hardware with

steaming access pattern. Steaming access pattern means write once and read any number of times but don't change content of files in file system. HDFS differ from other file system by its significant. HDFS is a very large distributed file system which is highly fault-tolerant, provides high throughput access to the large data and deployed on low-cost hardware. HDFS is mainly used for storing data, and simply adding the number of servers can achieve growth in storage capacity and computing power.

B. MAP REDUCE

MapReduce can make full use of the computing resources of each server's CPU, which efficiently handles with the stored data and calculations. To address the above issues, Google developed the Google File System (GFS), which is a distributed file system architecture model for processing large amount of data and created the MapReduce programming model. The MapReduce programming model is for processing the massive amount of data in parallel. Hadoop is open source software which manage MapReduce framework, written in Java, originally developed by Yahoo.

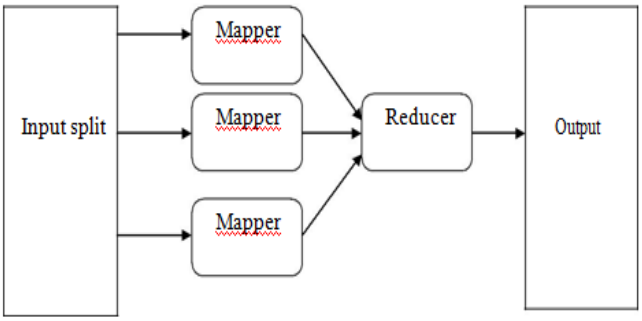

Fig .2. MapReduce Architecture

A MapReduce consists of two tasks namely the Map and Reduce task. Each Map task takes key-value pair as input and produce key-value pair as an output. The input data are split into various input splits. Based on the number of input splits Mapper will be assign. Record Reader is an interface between input split and Mapper which is used to convert record into key value pair. Mapper will read key value pair as an input and produce key value pair as an output. Now the Reducer will combine all the intermediate values associated with a particular key. Both input pairs of Mapper and Reducer are managed by the HDFS. The advantage of MapReduce is highly scalable, transparent fault-tolerant processing and automatic parallelization. Figure 2 shows the MapReduce architecture. MapReduce has been adopted by Google, Microsoft and Facebook.

III.CONDITIONAL RANDOM FIELDS

Conditional random fields (CRF), is a type of conditional probability model, has been widely applied in biomedical named entity recognition .The advantage of the CRF model is the ability to express long-distance-dependent and overlapping features. CRF has shown empirical success recently in Bio-NER, since it is free from the so-called label bias problem by using a global normalization. However, when facing large-scale data, the time efficiency of the CRF model with the traditional stand-alone processing algorithm is not satisfactory. For example, CRF takes approximately 45 hours (3.0GHz CPU, 1.0G memory, and 400 iterations) to train only 400K training examples. It is caused by the problem of CRF that the model parameter estimation cycle is long, because it needs to compute the global gradient for all features. The time complexity and space complexity of the whole algorithm show non-linear growth with the growth of the training data. To efficiently handle large-scale data, faster processing and optimization algorithms have become critical for biomedical big data. Hence, it is vital to develop new algorithms that are more suitable for parallel architectures. The CRF model needs to consider three key steps, i.e., feature selection, parameter estimation, and model inference. The parameter estimation step is very time-consuming because of the large amount of calculations especially when the training data set is large, which becomes the most important reason that degrades the performance of the CRF model. An optimization algorithm called Limited memory BFGS (L-BFGS) is a popular method that has been used to do parameter estimation of CRF. However, since it is an iterative algorithm, achieving high parallelism is not easy and demands considerable research attention for developing new parallelized algorithms that will allow them to efficiently handle large-scale data. It is a challenging task to parallelize such a dependent iterative algorithm. The task of making iterations independent of each other and thus leveraging and boosting parallel architectures is non-trivial. In this paper, we solve such an inter-dependent problem with an efficient strategy. Current methods of improving time efficiency of the CRF model focus on how to reduce the model parameter estimation time. However, the complexity of the model inference step increases quickly with the increase of constraint length of training data set as well. The model inference step can be performed using a modified Viterbi algorithm. The Viterbi algorithm within the MapReduce framework parallelizes the model inference step with a simple strategy.

IV.CONDITIONAL RANDOM FIELDS USING MAPREDUCE

Nowadays, FIM is most significantly employed by researchers as a result of it's wide applied in planet to search out the frequent itemsets. As a volume of information will increase day by day, the issues of measurability and potency become a lot of severe. As an answer to the current downside, we have a tendency to style a parallel mining of frequent itemset mistreatment CRF formula on MapReduce framework. during this paper we have a tendency to incorporate CONDITIONAL RANDOM FIELDS (CRF), instead of ancient FP-Tree. CRF has major four blessings over ancient FP-tree like; it involves solely 2 spherical of scanning that minimizes I/O overhead. Then the CRF may b e a extremely improved thanks to

partition a information, that significantly reduces the search area.

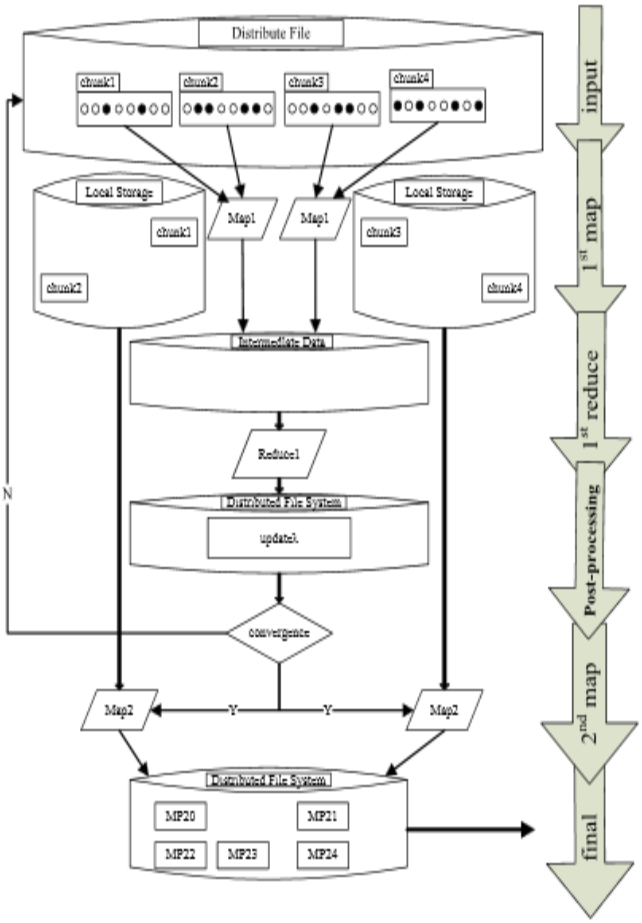

Fig. 3. Flow of the MapReduce CRF

Next is the frequent items in each transaction are inserted as nodes into the CRF for compressed storage. At last all frequent itemsets are generated without traversing the tree recursively by checking the leaves of each CRF which significantly reduces computing time. The sequences of steps followed are,

i. Datasets Partitioning
ii. MRLB
iii. MRViterbi

i. Datasets Partitioning

The CRF partitions the info set into M smaller sets and allocates every partitioned off subset to one map task. within the case of the Viterbi formula, the output of every map operate may be a partial state sequence for the native partition. Hence, we have a tendency to don't would like a combined output, and that we will save the scale back stage. The output of map that isn't any longer the intermediate result are going to be directly output and becomes the ultimate result. In MapReduce, the info set is split into several subsets, whose size depends on the amount of map tasks which will be run in parallel. to confirm the context {of every|of every} word in each sentence of Bio-NER, one sentence can't be split into 2 map tasks. Additionally, so as to realize optimum resource utilization and minimize the necessity for replication, we are going to develop a load reconciliation technique to partition an outsized dataset.

$$D_i = \begin{cases} \lceil N/M \rceil, & \text{if } R \neq 0,\ i = 1, ..., R; \\ \lfloor N/M \rfloor, & \text{if } R \neq 0,\ i = R+1, ..., M; \\ N/M, & \text{if } R = 0,\ i = 1, ..., M; \end{cases}$$

Where M denotes the amount of map tasks, and R resembles N mod M. we will divide the coaching information into M random subsets with about equal size. If N mod M = zero, each map tasks has one input split with [N/M] sentences. If N Mod M = zero, R map tasks have the input split with [N/M] sentences et al have the input split with [N/M] sentences.

ii. MRLB

Parameter estimation for giant dataset, the model can hugely increase the time consumption. Concerning ninetieth of the full computation time of L-BFGS is employed for the parameter estimation. If the parameter estimation is accelerated, time consumption can slow down sharply. Therefore, the most a part of parallelization of the L-BFGS formula is parallelized objective operate gradient calculation.

We can extract the factor as follows,

$$\frac{\partial L(\vec{\lambda})}{\partial \lambda_k} = \sum_{i=1}^{N} \left(\sum_{t=1}^{T} f_k(y_t^i, y_{t-1}^i, x_t^i) - \sum_{y} p(y^{i'}|x^i) f_k(y_t^{i'}, y_{t-1}^{i'}, y_t^i) \right) - \frac{\lambda_k}{\sigma^2},$$

iii. MRViterbi

The MRViterbi partitions the info set into M smaller sets so as to balance the load and allocates every partitioned off subset to one map task, every map optimizes a partition in parallel. Within the Viterbi formula, the output of every map operate may be a partial state sequence for the native partition. Hence, no have to be compelled to mix the output and scale back method time is saved. The output of map that isn't any longer the intermediate result are going to be directly output and becomes the ultimate result.

V.FREQUENT ITEMSET MINING

Frequent items are an item that occurs frequently in the dataset. Frequent itemset mining (FIM) is a one of the core data mining operation. Frequent itemset mining is mainly used for market basket analysis. Consider an example a set of items

that contains bread and butter which always occurs frequently together. A traditional frequent itemset mining algorithms are Apriori and FP-growth algorithm. Apriori algorithm is a level-wise iterative approach were k items are used to generate the k+1 items. Apriori algorithm consists of two steps join step and prune step. Initially candidate items are generated by joining process after that by checking the minimum support count frequent items will be generated. The process will be repeated until all k frequent items generation. However it has a disadvantage that many candidate items should generate which increases the computing time. To overcome that a pattern growth approach algorithm is proposed which significantly reduce the size of candidate sets. FP-Growth algorithm adopts a divide and conquers strategy for finding frequent itemsets. It also has some disadvantage that frequent items are generated by repeated scanning of database and recursive traversing of tree.

i. Generating one Itemsets and K Itemsets

ii. Generating Frequent K Itemsets

i. **Generating one Itemsets and K Itemsets**

Phase1 consists of two round of scanning the database. At the first round of scanning the database frequent one item will be generated based on the minimum support count. At the second round of scanning the database all k-items will be generated by pruning the infrequent items from each transaction.

ii. Generating Frequent K Itemsets

Phase2 consists of a two process decompose each 'h' itemsets into 'k' itemsets. After decomposing process the repetitive construction of K-CRF-Tree and all 'k' frequent itemsets are generated by checking the leaves of CRF-Tree where 'k' is from M down to 2. After decomposing process 'k' itemsets are generated that are used for the construction of K CRF Tree. Initially the root is labelled as null.

Then each 'k' itemsets are inserted into the tree. If first frequent item exists as one of the children of the root, then it denote the child as a temporary 1^{st} root, if it is not exist then add a new node for this item as a child of the root node and denote it as temporary 1^{st} root. Then the s^{th} frequent item of the k itemset, where 's' is from 2 to k - 1, check if the s^{th} frequent item exists as the children of the temporary $(s-1)^{th}$ root, then denote the child as a temporary s^{th} root. If it does not exist, then add a new node under this item as a child of the temporary $(s-1)^{th}$ root and denote it as a temporary s^{th} root. This process is repeated until K-CRF Tree is constructed. By checking the leaf node all k frequent items will be generated.

VI. CATEGORICAL DATA

Three groups of key words in MEDLINE by using GoPubMed are,

i. First group is biological-process and disease,
ii. Second group is cellular-component and disease,
iii. Third group is molecular-function and disease.

There are two effective parallel implementations currently, i.e., the CRF based on MPI (Message Passing Interface) and GPU (Graphics Processing Units). MPI and GPU are not suitable for large volumes of data in data-intensive applications. The drawback of MPI is communication delay in a big data environment for data-intensive applications, because a large amount of data are exchanged between a large number of nodes, and network communications will spend long time, such that the MPI method shows low performance. Due to the capacity limits of global memory and the bottleneck of data transmission for data intensive applications in a big date environment, the GPU method also shows low performance. Hadoop, an implementation of MapReduce, has a master-slave file system HDFS, which is the underlying support for the MapReduce data processing function. Hadoop can easily realize the computation of data storage migration computation", thus greatly improve the computational efficiency of the system. MapReduce deals with huge amount of data, for data-intensive applications. Virtual machine instances are used in a public cloud to run Hadoop applications and the CPU instructions, memory space within a virtual machine have to be translated and mapped to its physical machine host. Therefore, the intermediate operation degrades the efficiency of running Hadoop jobs, and deploys them on physical machines directly. Meanwhile virtual machine templates enables public cloud running in Hadoop applications and more execution nodes can be instantiated. Therefore, the scalability capacity will be much better, but this is not the focus of this paper. To analyze the speed in a efficient way, a local cluster interacts with the virtualization hypervisor, reveals the real performance of Hadoop jobs.

VII. RESULTS AND DISCUSSIONS

The experiment dataset is collected from different groups of key words in MEDLINE by using GoPubMed. The first group is biological process and disease, the second group is cellular-componentand disease, and the third group is molecular-function and disease. The unparallel CRF was carried out on a single machine.There are two effective parallel implementations currently, i.e., the CRF based on Message Passing Interface and Graphics Processing Units. However, they are not suitable for large volumes of data in data-intensive applications.

A. Message Parsing Interface and GPU

The strongest weakness of MPI is communication latency in a big data environment for data-intensive applications, because a large amount of data are exchanged between a large number of nodes, and network communications will spend long time, such that the MPI method shows low performance. Due to the capacity limits of global memory and the bottleneck of data transmission for data-intensive applications in a big data environment, the GPU method also shows low performance. Hence, we have the proposed algorithm compared with the sequential CRF algorithm, but not compared with other parallel implementations of the algorithm.

B. Hadoop to overcome weakness of MPI and GPU

Hadoop 2.6.0**,** YARN (Yet Another Resource Negotiator) or MRv2 is a Next generation of map reduce, fundamental idea is to split up the two major functionalities Job Tracker, resource management and job scheduling into separate deamons. The idea is to have a global ResourceManager (RM) and per application ApplicationMaster(AP). The ResourceManager and per-node slave, the NodeManager (NM), cast the data-computation framework. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is a framework specific library, engaged with the NodeManager(s) to execute and monitor the tasks and it negotiate resources from the ResourceManager.

Hadoop, an implementation of MapReduce, has a master slave file system HDFS, which is the underlying, support for the MapReduce data processing function. With the HDFS, Hadoop can easily realize "computation to the data storage migration", thus greatly improve the computational efficiency of the system. MapReduce can deal with huge amount of data, especially for data-intensive applications.Recognition of biomedical named entity using conditional random fields in this paper is a data-intensive application in the big data environment, so the Hadoop method is a suitable method.

Virtual machine instances are usually used in a public cloud to run Hadoop applications. The CPU instructions and memory space within a virtual machine need to be translated and mapped to its physical machine host. Therefore, this intermediate operation degrades the efficiency of running Hadoop jobs and deploy them on physical machines directly. Meanwhile running Hadoop applications on a public cloud can be enabled by virtual machine templates and more execution nodes can be instantiated. Therefore, the scalability capacity will be much better, but this is not the focus of this paper.

C. MRCRF implementation

MRCRF is a combination of LBFGS and Virterbi algorithms where the dataset is divided into different chunks and the infrequent items are removed and merge the resultant from different chucks into single. Frequent item set mining is the process under these process and the non duplicate record means the not highly refereed or the biomedical field not been discussed or the documents not available for particular disease or molecular combination etc., all over the system.To analyze the speedup of IMRCRF in a more efficient way, a local cluster with less interaction with the virtualization hypervisor reveals the real performance of Hadoop jobs. A document of 100000 records uploaded respectively, the dataset is divided into different chunks for mapreduce process. Minimum four chucks are used to achieve the better performance.

D. PARAMETERS FOR EVALUATION

The performance for proposed methods can be evaluated by using the following parameters. Parameters which are considered for evaluating the experiments are:

i. Minimum support
ii. Scalability

i. Minimum Support Count

Minimum support count plays the important role in mining frequent itemsets. When we increase the minimum support threshold the running time of the proposed algorithm reduces. A small minimum support slows down the performance of the evaluated algorithms. This is because an increasing number of items satisfy the small minimum support when the minsupport is decreased; it takes an increased amount of time to process the large number of items.

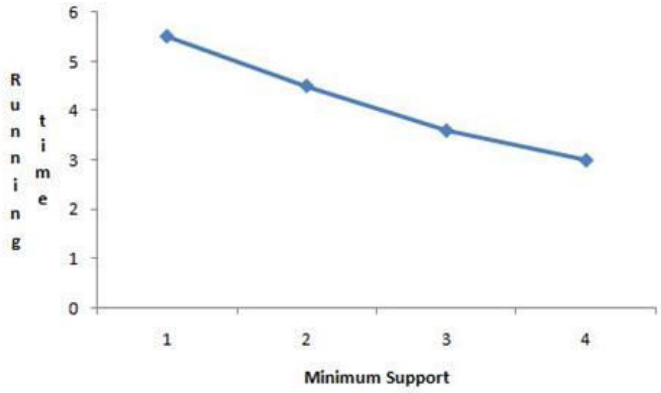

Fig 4 Execution time of four different minimum support counts.

ii. Scalability

In this experiment, we evaluate the scalability of the proposed algorithm when the size of input dataset grows dramatically. The parallel mining process is slowed down by the excessive data amount that has to be scanned twice. The increased dataset leads to a long scanning time. An output of the second MapReduce job are distributed and stored in intermediate files based on the length of itemset; these files are accessed by the third MapReduce job as an input. Further, the decomposed results are written into these external files. In summary, the scalability of the proposed algorithm is higher when it comesto parallel mining of an enormous amount of data.

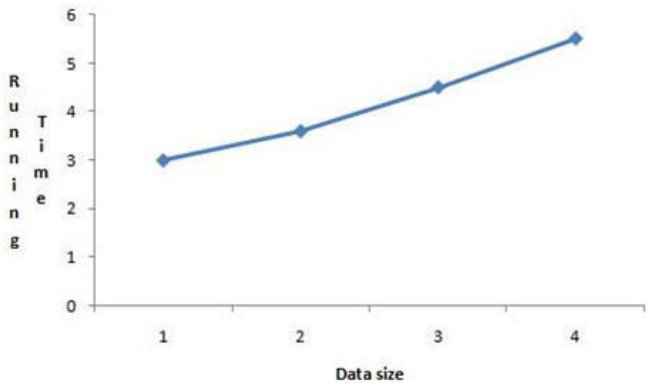

Fig 5 Running time of different sized datasets.

VIII.RELATED WORKS

There has been some prior works proposed in the literaturefor accelerating CRF. These methods essentially gain acceleration by omitting important information of labels and losing accuracy. Pal et al. proposed a Sparse Forward Backward (SFB) algorithm, in which marginal distribution is compressed by approximating the true marginal using Kullback-Leibler (KL) divergence [25]. Cohn proposed a Tied Potential (TP) algorithm which constrains the labeling considered in each feature function, such that the functions can detect only a relatively small set of labels [2]. Both of these techniques efficiently compute the marginal with significantly reduced runtime, resulting in faster training and decoding of CRF. Although these methods could reduce computational time significantly, they train CRF only on a small data set. In order to handle large data, Jeong et al. proposed an efficient inference algorithm of CRF for large-scale natural language data which unified the SFB and TP approaches [11]. Lavergne et al. addressed the issue of training very large CRF, containing up to hundreds output labels and several billion features. Efficiency stems here from the sparsity induced by the use of penalty term [15]. However, none of these works described so far explore the idea of accelerating CRF in a parallel or distributed setting and thus their performance is limited by the resources of a single machine. Given that CRF is weak in processing massive data, the idea of parallelization is introduced into the algorithms. Xuan-Hieu et al. proposed a high-performance training method of CRF on large-scale data by using massively parallel computers [38]. In [19], a novel distributed training method of CRF is proposed by utilizing the clusters built from commodity computers. The method employs Message Passing Interface (MPI) and improves the time performanceon large datasets. Recently, in [21], an efficient parallel inference on structured data with CRF based on Graphics Processing Units (GPU) is introduced and it is testified that the approach is both practical and economical on very large data sets. These methods achieve significant reduction in computational time without losing accuracy. However, they are not suitable for a distributed cloud environment, where usually the communication cost is higher. In our approach,we overcome this limitation by a parallel implementation of CRF based on MapReduce which is suitable for huge data sets [32]. MapReduce is an excellent model for distributed computing on large data sets, which was introduced by Google in 2004. It is an abstraction that allows users to easily create parallel applications while hiding the details of data LI ET AL.: HADOOP RECOGNITION OF BIOMEDICAL NAMED ENTITY USING CONDITIONAL RANDOM FIELDS 3041 distribution, load balancing, and fault tolerance. At present, it is popular in text mining of various applications, especially natural language processing (NLP) [8], [31], [37].Laclavik et al. presented a pattern of annotation tool based on the MapReduce architecture to process large amount of text data [13]. Lin and Dyer discussed the processing method of data intensive text based on MapReduce, such as parallelization of EM algorithm and HMM model [18]. Palit and Reddy proposed two parallel boosting algorithms, i.e., ADABOOST.PL and LOGITBOOST.PL, scalable and parallel boosting with MapReduce [26].

IX.CONCLUSION

To solve the scalability and efficiency in the existing parallel mining algorithms for frequent itemsets for frequent itemsets, we applied the parallel mining of frequent itemsets using Frequent Itemset Ultrametric Tree on MapReduce framework. We incorporate the Frequent Itemset Ultrametric Tree rather than conventional FP trees, thereby achieving compressed storage and avoiding the necessity to build conditional pattern bases. The proposed algorithm integrates three MapReduce jobs to accomplish parallel mining of frequent itemsets. At the end of the third MapReduce job all frequent K itemsets are generated. To evaluate the performance of the proposed MRCRF algorithm on MapReduce framework we use synthetic datasets in our experiments. The future research direction is the distributed cache technique is used to store the intermediate result of each MapReduce job which will significantly improves performance of parallel mining of frequent itemsets using MRCRF on MapReduce framework.

X.REFERENCES

[1] Chang E.Y., Li H., Wang Y., Zhang D. and Zhang M. (2008), 'PFP: Parallel FP-growth for query recommendation', in Proc. ACM Conf. Recommend.Syst., Lausanne, Switzerland, pp. 107–114.

[2] Chang W.L., Chen P.L. and Lin K.W. (2011), 'A novel frequent pattern mining algorithm for very large databases in cloud computing environments', in Proc. IEEE Int. Conf. Granular Comput. (GrC), Kaohsiung, Taiwan, pp. 399–403.

[3] Han J., Mao R., Pei J. and Yin Y. (2004), 'Mining frequent patterns without candidate generation: A frequent-pattern tree approach', Data Min. Knowl. Disc., vol. 8, no. 1, pp. 53–87.

[4] Hsueh S.C., Lin M.Y. and Lee P.Y. (2012), 'Apriori-based frequent itemset mining algorithms on MapReduce', in Proc. 6th Int. Conf. Ubiquit. Inf. Manage. Commun. (ICUIMC), Danang, Vietnam, pp. 76:1–76:8.

[5] Hsu T.J., Tsay J.Y. and Yu J.R. (2009), 'FIUT: A new method for mining frequent itemsets', Inf. Sci., vol. 179, no. 11, pp. 1724–1737.

[6] Li E., Liu L., Tang Z. and Zhang Y. (2007), 'Optimization of frequent itemset mining on multiple-core processor', in Proc. 33rd Int. Conf. Very Large Data Bases, Vienna, Austria, 2007, pp. 1275–1285.

[7] Tang P. and Turkia P.M. (2006), 'Parallelizing frequent itemset mining with FP-trees', in Proc. 21st Int. Conf. Comput. Appl., Seattle, WA, USA, pp. 30– 35.