International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Document Classification using Parallel Processing

Nikhil R. Vyawahare
MTech CSE-IT
Vishwakarma Institute of technology,
666, Upper Indira Nagar, Bibvewadi, Pune, Maharashtra
411037.

S. G. Lade
Assistant Professor
Vishwakarma Institute of technology,
666, Upper Indira Nagar, Bibvewadi, Pune, Maharashtra
411037.

***Abstract*— The wide availability of huge number of documents from different domains, real-time and archrivals data documents increase as fast as or faster than computing power. Many classical algorithms were developed to solve classification problem but many of them usually have large time complexity and with increasing number of documents it is necessary to find algorithm which are able to find solution in reasonable time. Because sequential processor can perform a process at a time. The parallel processing is a novel technique for scaling up the algorithms. Parallel processing for document classification can take the advantage of increasing availability of multi-processors or multi-cores processors. The Graphics processing unit is consists of multi-core processor and they can perform parallel process in small amount of time.**

I. INTRODUCTION

A large portion of the available information is stored in document database which consist of large collections of documents from various areas like news articles, research papers, books, e-books, e-mails, and Web pages and these can be known as document database. Document databases are rapidly growing due to the increasing amount of information available in electronic form. Nowadays most of the information in government, small or large industry, business, and other institutions are store in the form of documents. Document classification is process of assigning predefine label or categories to the documents which include unstructured and semi structured information. It is important because, with existence of a tremendous number of documents, it is tedious and essential to be able to automatically organize such documents into classes to facilitate document retrieval and subsequent analysis. Document classification is the primary process of retrieving, filtering, clustering and extracting documents. A general procedure for document classification is as follows: First, a set of pre-classified documents is taken as the training set. The training set is then analyzed in order to derive a classification scheme. Such a classification scheme often needs to be refined with a testing process. The so-derived classification scheme can be used for classification of other new documents.

The implementation of document classification over the large set of documents on sequential processor (CPU) takes large amount of time for learning and classification. Many classical algorithms were developed to solve classification problem but many of them usually have large time complexity and with increasing number of documents it is necessary to find algorithm which are able to find solution in reasonable time, Because CPU can perform a calculation one at a time. The parallel processing is a novel technique for scaling up the algorithms. Parallel processing for document classification can take the advantage of increasing availability of multi-processors or multi-cores processors. The Graphics processing unit is consists of multi-processor and they can perform parallel process in small amount of time. The GPU is not only the used by graphical application but also non-graphics application (like classification).

In this paper, we introduce the parallel process for document classification. For Implementation of such system we are using the NVIDIA Compute Unified Device Architecture, through a new API, an easy way to take advantage of the high performance of GPUs for parallel processing [2].

The rest of the paper is organized as follows. The section 2 describes process of document classification, section 3 introduce details about Graphic Processing Unit. Section 4 describes the algorithm for suggested in this paper for parallel processing. Section 5 contains description of experimental results. And remaining sections contains conclusion and references.

II. DOCUMENT CLASSIFICATION

Document classification is process of assigning the documents to predefined categories. Let, if a document $di \in D$ belongs to the category $ci \in C$ according to the knowledge of the correct categories known for subset $D_T \subset D$ of training documents, where *D* is the collection of all documents and *C* is collection of all categories. Generally, each document may belong to more than one category and each category may contain more than one document [1].

The document classification task can be solved by automatic classifier. The documents need to be in the form of input that classifier accept. Automatic document classifier needs several pre-processing steps, which must convert document into classification capable form. Pre-processing of documents follows several steps, the first step in pre-processing is preprocessing of words, a creation of vector representation of the documents. Each document is parsed out and list of used word with their frequency is extracted. Each word is compared with list of stop-words, which are useless and not take part in classification, because they are presents in most of the documents and it increase the length of document. Each word has to converted into it canonical form using

normalisation algorithm called as stemming, such as Porter Stemming Algorithm[23], it propose five rules to convert a word into its canonical form called as stem. A word is form by two components stem and its prefixes and postfixes, it will be more useful for classification if these prefixes and postfixes are removed. When these process is finished, the document collection is represented as *document term frequency matrix (Doc_i * $TF_{i,j}$)*, where Doc_i refer to the each document in the collection and $TF_{i,j}$ refer to the frequency of *j* term in the document *i*. In this representation, only relation of the term to the individual document is concerned, but the classification across the document collection must be computed, therefore we need to evaluate term importance in individual document according the importance of the term in collection and it will called as weights. One of the way we may define a weights to the terms is according to TF-IDF (term- frequency inverse frequency). The weights $w_{i,j}$ of the term *j* in document *i* is computed as :

$$w_{i,j} = t_{i,j} \times \log \frac{F}{f_i}$$

Where $t_{i,j}$ is the number of times that the j appears in document *i*,f_i is the number of times that the term t_j appears in the entire document database and F is the number of unique terms in document collection.

The previous paragraph described the process for conversion of document collection into (document – term-weight) matrix, but the number terms with non-trivial weight for each document is still large (tenths of thousands). This may cause problem with automatic classifiers because of the large number of terms to process. Therefore, feature/term selection were may be applied. Several approach to feature selection were developed in [23] and [24]. One of the popular approach is entropy weight scheme. The entropy weighting scheme is computed to the each term as a multiplication of the local weighting scheme L_{ij} and the global weighting scheme G_j of the document *i* and term *j*. The definition of the scheme is following,

$$L_{ij} = \begin{cases} 1 + \log TF_{ij}, TF_{ij} > 0 \\ 0, otherwise \end{cases}$$

$$G_k = \frac{1 + \sum_{k=1}^{N} \frac{TFij}{Fj} \log \frac{TF_{ij}}{F_k}}{\log N}$$

Where N is number of document in collection, TF_{ij} refers to the frequency of the *j* term in the document *i*, F_h is a frequency of term *k* in the entire document collection.

III. GPU

In GPU[12], the GPU takes advantage of a large number of execution threads to find work to do when some of them are waiting for long-latency memory accesses, thus minimizing the control logic required for each execution thread. Small cache memories are provided to help control the bandwidth requirements of these applications so multiple threads that access the same memory data do not need to all go to the DRAM. The GPUs are designed as numeric computing engines and they will not perform well on some tasks on which CPUs are designed to perform well therefore, one should expect that most applications will use both CPUs and GPUs, executing the sequential parts on the CPU and numerically intensive parts on the GPUs. This is why the [10] CUDA (Compute Unified Device Architecture) programming model, introduced by NVIDIA in 2007, is designed to support joint CPU/GPU execution of non graphics application.

IV. K-NEAREST NEIGHBOUR (KNN)

K-Nearest Neighbour algorithm(k-NN) is a classification technique in data mining. This algorithm is based on a majority vote of the k closest training samples in the feature space. If k=1, then the object is simply assigned to the class of its nearest neighbour. The k's value can be anything it is depending on what dataset is used for classification. Various measures (e.g. Euclidean distance, cosine measurement, KL) can be used to compute the distance between two data sample points in feature space, the most desirable distance metrics may differ in different applications.

K nearest neighbour or KNN[5] classification determines the decision boundary locally. For only one nearest neighbour we assign each document to the class of its closest neighbour. For KNN we assign each document to the majority class of its k closest neighbours where k is a parameter. The work of kNN classification is based on the contiguity hypothesis that we expect a test document *d* to have the same label as the training documents located in the local region surrounding *d*. The Voronoi tessellation of a set of objects space into Voronoi cells, each object's cell will consists of all points that are closer to the object than to other objects. In our case, the objects are documents. Then Voronoi tessellation partitions the plane into |*D*| set it seems like convex type of polygons, each containing its corresponding document , where a convex polygon is a convex region in 2-dimensional space bounded by lines. For general k ∈ N in KNN, consider the region in the space for which the set of k nearest neighbours is the same. This again is a convex polygon and the space is partitioned into convex polygons, within each of which the set of k nearest neighbours is invariant.

A. Parallel KNN

The parallelization of k-NN is not applied on training period but on the prediction of the unknown instances, which is given as follows:

1. Let D be the dataset, Partition the dataset *D* into *P* blocks like *D1*,D2, .. ,DP , each processor will handles roughly ///*D*///*P*
2. Processor *Pr* calculates the *k* nearest neighbours *Nr* with the local training samples *Dr*.
3. A global reduction computes the overall *k* nearest neighbours *Nglobal* from *N1*,...., *NP*, and then assign the object to the class which most common amongst *Nglobal*.

B. Algorithm

As describe in above, Parallel kNN, it partition data set D into P processors or cores and each core calculates k nearest neighbour for their local documents. The results from all these cores, gives the total k nearest neighbour for the test document, and this operation is perform by the CPU.

Training:

1. Document Collection into D dataset.
2. Tokenization of documents, it convert documents into set of tokens.
3. Remove the stop words from documents.
4. Partition D dataset into T threads, each thread. handles ||D||/T number of documents.
5. Perform stemming for all threads.
6. Entropy feature selection.

Testing:

Following are the steps perform by each core to calculate K nearest neighbour.

1. Calculate term frequency TF.
2. Calculate inverse term frequency IDF, $IDF_i = \log(N/df_i)$
3. Calculate TFxIDF document vector, it also called as a weight, $W_{i,j} = tf_{ij}$ X IDF_i
4. Calculate document vector length
$$|\overrightarrow{d_j}| = \sqrt{\sum_{i=1}^{m} w^2_{i.j}}$$
5. Calculate document cosine similarities, using
$$sim(d_j, d_k) = \frac{\overrightarrow{d_j}.\overrightarrow{d_k}}{|\overrightarrow{d_j}|.|\overrightarrow{d_k}|} = \frac{\sum_{i=0}^{m} w_{ij}.w_{ik}}{|\overrightarrow{d_j}|.|\overrightarrow{d_k}|}$$
6. Collect k samples nearer to the test sample.

V. EXPERIMENT AND RESULTS

This section contains description of performed experiments on several data sets.

I. Hardware configuration

In our implementation we used a system with Intel core i5 processor @2.30 GHz with windows 7 operation system, nVIDIA GeForce 520M which has 96 CUDA cores and 2 GB RAM.

Table 1 Technical Specification (NVIDIA GeForce 520M)

Technical specifications	**Compute capability** (version) **2.X**
Maximum x-, y-, or z-dimension of a grid of thread blocks	65535
Maximum dimensionality of thread block	3
Maximum number of threads per block	1024
Warp size	32
Maximum number of resident blocks per multiprocessor	8
Number of 32-bit registers per multiprocessor	32 K
Maximum number of 32-bit registers per thread	64
Maximum amount of shared memory per multiprocessor	48 KB
Number of shared memory banks	32
Amount of local memory per thread	512
Constant memory size	64 KB

VI. DOCUMENT COLLECTION

In our implementation, we selected five categories with high number of documents. There are 2000 text documents. These text documents are collected from the internet. All documents are belongs to the only that five categories. The categories are research domains/areas in computer science and engineering stream. These documents are related to those categories. The classes or categories are Artificial Intelligence, Text mining, Networking, Data mining, Image Processing. There are twelve hundred documents used for training purpose and other eight hundred documents are used for testing.

We experimented, this data set for several different values of k. The accuracy of the classifier is depends on the value of k. we experimented for k values with two to twenty and results are improved with greater value of k. Accuracy can be calculated by using one of the most popular metrics in document classification is precision and recall. Precision (*Pr*) is defined as a probability that selected document is classified correctly and recall (*Re*) is defined as a probability that randomly selected document is assigned to the correct category. Mathematical definitions are as follows:

$$\mathrm{Pr} = \frac{TP}{TP + FP}$$

$$\mathrm{Re} = \frac{TP}{TP + FN}$$

Where TP (true positive) is count of correctly classified documents, FP (false positives) is count of documents incorrectly not assigned into category. And combination of the precision and recall is F1 measure, it can be calculated as,

$$F1 = \frac{2 \times \mathrm{Pr} \times \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}}$$

Following table shows the efficiency of the algorithm in terms of precision, recall and F1 for each categories or class.

Table 2 Efficiency of document classification for collected dataset

Class	Pr	Re	F1
AI	0.271	0.323	0.294
TM	0.033	0.064	0.435
DM	0.703	0.391	0.502
IP	0.865	0.937	0.899
NE	1.000	0.869	0.929

VII. CONCLUSION

This paper described a document classification algorithm using KNN and implemented on GPU. The parallel processing is a novel technique for scaling up the algorithms. Parallel classification algorithm is developing to take advantage of the increasing availability of multi-processor. GPU is good for parallel processing operations and faster than CPU. Implementation of document classification reduces time complexity for learning and testing a huge set of documents.

ACKNOWLEDGEMENT

We are thankful to the computer department of Vishwakarma Institute of Technology, Pune for their valuable support.

REFERENCES

[1] Vandana Korde,C Namrata Mahender,"Text Classification And Classifiers: A Survey", International Journal Of Artificial Intelligence & Applications (IJAIA), Vol.3, No.2, March 2012.
[2] David B. Kirk and Wen-mei W. Hwu,"Programming Massively Parallel Processors A Hands-on Approach"
[3] Jan Platos, Vaclav snasel, Tomas Jezowicz, Pavel, Ajith Abraham "A PSO-Based Document Classification Algorithm accelerated by the CUDA Plateform" in IEEE International Conference on Systems, Man and Cybernetics, October 14-17, 2012, COEX, Seoul, Korea.
[4] Han Xiao "Towards Parallel and Distributed Computing in Large-Scale Data Mining: A Survey" April 8, 2010.
[5] M. Connor and P. Kumar.” Parallel construction of k-nearest neighbour graphs for point clouds. In Eurographics Symposium on Point-Based Graphics”, 2008.
[6] Jesse St. Charles, Robert M. Patton, Thomas E. Potok, And Xiaohui Cui "Flocking-Based Document Clustering On The Graphics Processing Unit" published in U.S. Department of Energy Journal of Undergraduate Research, 2009.
[7] Thanh-Nghi Do, Van-Hoa Nguyen, and François Poulet "Speed Up SVM Algorithm for Massive Classification Tasks", ADMA 2008, LNAI 5139, pp. 147–157, 2008.
[8] Joseph M. Cavanagh, Thomas E. Potok, Xiaohui Cui "Parallel Latent Semantic Analysis using a Graphics Processing Unit", GECCO’09, July 8–12, 2009, Montréal Québec, Canada.
[9] Yongpeng Zhang, Frank Mueller, Xiaohui Cui and Thomas Potok "GPU-Accelerated Text Mining" in EPHAM’09, March 22-25, 2009, Seattle, Washington.
[10] Bryan Christopher Catanzaro, Narayanan Sundaram and Kurt Keutzer "Fast Support Vector Machine Training and Classification on Graphics Processors" UCB/EECS-2008-11.
[11] T.L. Griffiths, M. Steyvers, D.M. Blei, and J.B. Tenenbaum.” Integrating topics and syntax. Advances in neural information processing systems”, 17:537–544, 2005.
[12] S. Manavski and G. Valle.” CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. BMC bioinformatics”, 9(Suppl 2):S10, 2008.
[13] D. Newman, A. Asuncion, P. Smyth, and M. Welling. “Distributed inference for latent dirichlet allocation”. Advances in Neural Information Processing Systems, 20:1081–1088, 2007.
[14] Y.Wang, H. Bai, M. Stanton,W.Y. Chen, and E.Y. Chang. “ PLDA: Parallel Latent Dirichlet Allocation for Large-Scale Applications”. AAIM, June, 2009.
[15] Reza Farivar, Daniel Rebolledo, Ellick Chan, Roy Campbell "A Parallel Implementation of K-Means Clustering on GPUs", 2009.
[16] Erik Lindholm, John Nickolls, Stuart Oberman “Nvidia Tesla:Aunified Graphics And Computing Architecture” Published by the IEEE Computer Society.0272-1732/2008 IEEE.
[17] J. Montrym and H. Moreton, ‘‘The GeForce 6800,’’ IEEE Micro, vol. 25, no. 2, Mar./ Apr. 2005, pp. 41-51.
[18] J. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In Proceedings of the International Conference on Very Large Data Bases, pages 544–555. Citeseer, 1996.
[19] B. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In Proceedings of the 25th international conference on Machine learning, pages 104–111. ACM, 2008.
[20] E.Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. Psvm: Parallelizing support vector machines on distributed computers. Advances in Neural Information Processing Systems, 20, 2007.
[21] R. Jin and G. Agrawal. A middleware for developing parallel data mining implementations. In Proceedings of the first SIAM conference on Data Mining. Citeseer, 2001.
[22] Wei Zhanga, Feng Gao,"An Improvement to Naive Bayes for Text Classification", doi : 10.1016 / j.proeng . 2011 . 08 . 404.
[23] Ms. Anjali Ganesh Jivani,"A Comparative Study of Stemming Algorithms",IJCTA | NOV-DEC 2011.
[24] Y. Saeys, I. Inza, and P. Larraaga, “A review of feature selection techniques in bioinformatics”, Bioinformatics, vol. 23, no, 19, pp. 2507-2517, 2007.
[25] Y. Yang and J. Pedersen, Feature selection in statistical learning of text categorization, Morgan Kaufmann, 1997, pp, 412-420.