Document Classification Using KNN on GPU

S.G. Lade and Nikhil Vyawahare

Abstract—**Real-time and archrival data documents are increases as fast as or faster than computing power now a days. Document classification using k-nn classification algorithm takes more time in searching nearer neighbors in large training dataset, it include large number of computations. The time for classification increases in proportion to the number of documents. Therefore it is necessary to find alternative for implementation of KNN.The Graphics processing unit consists of multi-core processors and they can perform computations in parallel, in small amount of time. This paper describes implementation of parallel k-nn algorithm for document classification on NVIDIA graphics processors.**

Index Terms— **classification, k-nn, gpu, text mining.**

I. INTRODUCTION

Document classification is a well known problem that is focused on assigning predefined labels or categories to the documents found in the searched collection. Document classification may be applied into many areas such as information retrieval, personalized recommendation systems, and many others.

With the increasing number of documents become document classification even more significant. This task has many difficulties, because documents may have different nature, and it is an unstructured data. Many sequential algorithms are developed for solving document classification problem. But sequential algorithms usually take large execution time with increasing number of documents. One of such algorithm K-Nearest Neighbors (k-NN)[5] algorithm which is based on a majority vote of the k closest training examples in the feature space.

K-Nearest Neighbors (k-NN) algorithmtakes too much of timewhen the number of documents is large. Therefore it is necessary to find another algorithm or different style of programming. We select another possibility and focus on the graphics processing unit (GPU). Graphics processing unit is very promising technology, because it brings massive parallel architecture for reasonable price and becomes a new technical standard in writing application and algorithms.

In the previous paper [27] we proposed the parallel approach for K nearest neighbor algorithm. This paper describes modification in the previous algorithm and focus on the implementation, experiments and results of parallel algorithm on graphics processing unit.

The rest of paper is organized as follows. The section II describes related work done in previous paper, the section III describes the proposed algorithm. The section IV describes the implementation of algorithm on GPU; section V includes performed experiments and results. Last section contains conclusion of this paper.

***Mrs. S.G. Lade**, Department of Computer Engineering, Vishwakarma Institute Of Engineering, Pune, Maharashtra,, India.*

***Mr. Nikhil Vyawahare**, CSE-ITM.Tech Student, Vishwakarma Institute of Technology, Pune, Maharashtra, India.*

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 content bearing words 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 be converted into its canonical form using normalization 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 suffixes; it will be more useful for classification if these prefixes and postfixes are removed. When this 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 wayto 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 termt_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 is applied. Several approaches to feature selection were developed in [23] and [24]. One of the popular approaches is entropy weight scheme. The entropy weighting scheme computesthe weight of 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 follows,

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

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

Where N is number of documents 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. Graphics Processing Unit

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[2]. 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 applications.

CUDA is the hardware and software architecture that enables NVIDIA GPUs to execute programs written with C, C++, Fortran, OpenCL, DirectCompute, and other languages. A CUDA program calls parallel kernels. A kernel executes in parallel across a set of parallel threads. The programmer or compiler organizes these threads in thread blocks and grids of thread blocks. The GPU instantiates a kernel program on a grid of parallel thread blocks. Each thread within a thread block executes an instance of the kernel, and has a thread ID within its thread block, program counter, registers, per-thread private memory, inputs, and output results.[2]A thread block is a set of concurrently executing threads that can cooperate among themselves through barrier synchronization and shared memory. A thread block has a block ID within its grid. A grid is an array of thread blocks that execute the same kernel, read inputs from localmemory, write results to global memory, and synchronize between dependent kernel calls.

In the CUDA parallel programming model, each thread has a per-thread private memory space used for register spills, function calls, and C automatic array variables. Each thread block has a per-Block shared memory space used for inter-thread communication and data sharing. Grids of thread blocks share results in Global Memory space after kernel-wide global synchronization.[2]

IV. Proposed Algorithm

A. Let CPU to be the master, the other N cuda cores as slaves.

B. Master launches the kernel for calculating the term frequency tf_{ij}, inverse term frequency (IDF) and ifidf weight (W_{ij}) for each sample feature.

$$IDF_i = \log(N/d_{fi})$$
$$W_{i,j} = tf_{ij} X\ IDF_i$$

C. Each individual processor now computes the distance measures independently and storing the computes measures in a local array (using cosine similarity formula as stated bellow).

$$|\overrightarrow{d_j}| = \sqrt{\sum_{i=1}^{m} w^2{}_{i.j}}$$

$$sim(d_j, d_k) = \frac{\vec{d_j}.\vec{d_k}}{|\overrightarrow{d_j}|.|\overrightarrow{d_k}|} = \frac{\sum_{i=0}^{m} w_{ij}.w_{ik}}{|\overrightarrow{d_j}|.|\overrightarrow{d_k}|}$$

D. Master then notes the end of processing for the sender processor and acquires the computes measures by copying them into its own array.

E. After the master has claimed all distance measures from all processors, the following steps are performed:
   a. Sort all distance measures in ascending order
   b. Select top k measures
   c. Count the number of classes in the top k measures
   d. The test sample's class will assign the class having the higher count among top k measures.

V. Implementation On Gpu

Programming implementation of algorithm includes some sequential operation and parallel operation. Document preprocessing and Knn ranked based class prediction has to execute sequentially. Other operations like calculating term frequency, weight and similarity between test and train sample can be executed parallel. There are five CUDA kernels which performs these parallel operations. Following are the kernel configuration with the purpose of each kernel.

Kernel 1- This kernel is used to calculate df, tf for each feature/term. This kernel is executed for 1024 threads per block and number of blocks(grid size) depend on the number of features/terms.

Kernel 2- This kernel is used to calculate weight(tfidf) for each term in each document, this is two dimensional kernel, x dimension is the number of documents(D) while y dimension is the number of features(F).

Kernel 3- This kernel is used to calculate vector length, each document is considered as the vector and this kernel is executed for D grid size and 1000 block size.

Kernel 4 and 5-These kernels are used to calculate cosine Similarity for all samples.

VI. Experiments and Results

A. 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

B. Document Collection

The dataset is a collection of documents found in survey of different research areas in computer science. The papers or documents are published article related to different areas in computer science. There are twelve different domains' collected, these text documents are collected from the internet, like online published journals (for example Science Direct, ACM Digital library and IEEE explorer). All documents belong to only those domains called categories or class of that document. Then name of classes or categories are Artificial Intelligence (AI), Text mining (TM), Networking (NE), Data mining (DM), Image Processing (IP), etc. As shown in below table, there are total two thousand documents in the data set, thirteen hundred documents are used for training purpose and other seven hundred documents are used for testing.

C. Efficiency of classification

To evaluate efficiency of classification, we calculate precision, recall, and f1 measures for both C program and CUDA program results. Precision and recall are the most popular metrics in document classification. 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, the mathematical definitions are as follows:

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

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

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,

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

$$Accu = \frac{TP + TN}{TP + TN + FP + FN}$$

Following table shows the efficiency of the algorithm in terms of precision, recall and F1 for each categories or class. Table 2 shows accuracy for classification and table 3 shows results for recognition.

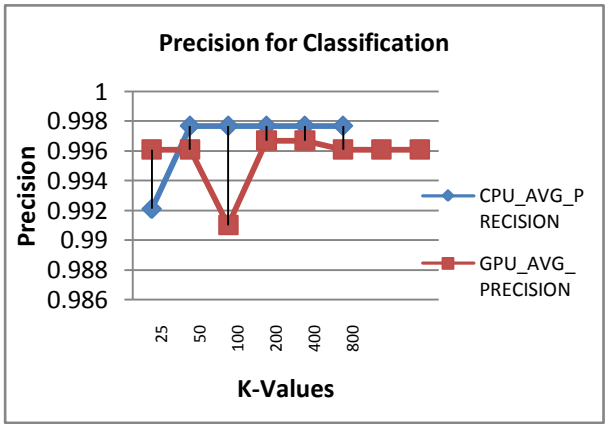
Table 2Accuracy for Classification

Class	Precision	Recall	F1-measure	Accuracy
AI	1	0.980392	0.990099	0.997692
AN	0.98	1	0.989899	0.998462
CG	1	1	1	1
CS	1	1	1	1
DM	0.993333	1	0.996656	0.999231
EM	1	1	1	1
HI	1	1	1	1
IP	1	1	1	1
NE	1	1	1	1
NF	1	1	1	1
SE	1	1	1	1
TO	1	1	1	1
Average	0.99211	0.992732	0.99694	0.998321

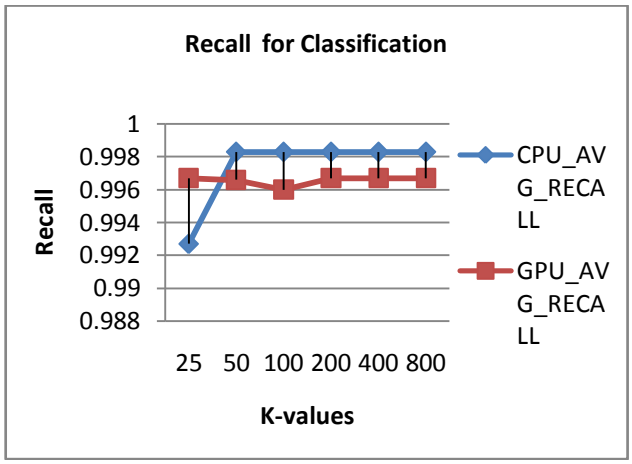
Table 3Accuracy for Recognition

Class	Precision	Recall	F1-measure	Accuracy
AI	0.495652	0.745098	0.5953	0.777937
AN	1	0.491803	0.659341	0.955587
CG	0.7	0.134615	0.225806	0.931232
CS	1	0.408163	0.57971	0.958453
DM	0.69	0.683168	0.686567	0.909742
EM	1	0.714286	0.833333	0.994269
HI	1	0.714286	0.833333	0.994269
IP	0.474359	0.948718	0.632479	0.938395
NE	0.791667	0.896226	0.840708	0.948424
NF	0.85	0.894737	0.871795	0.992837
SE	0.733333	0.468085	0.571429	0.952722
TO	0.5	0.465116	0.481928	0.938395
Average	0.743883	0.555333	0.650416	0.940516

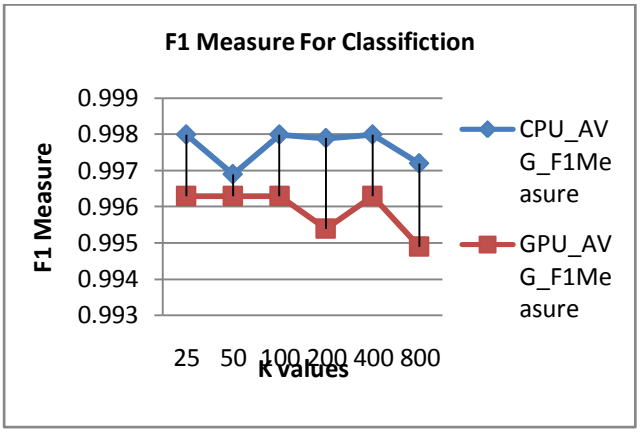
Accuracy of KNN classifier depends on the k value, for the experimentation we are executed classifier for different K values for the same dataset. Following graphs show precision, recall, f1 measure and accuracy for different K values on CPU and GPU.



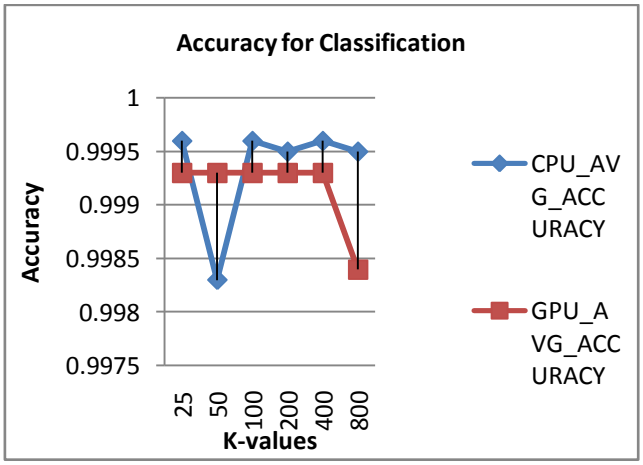
Graph 1 Average Precision for classification over different k-values



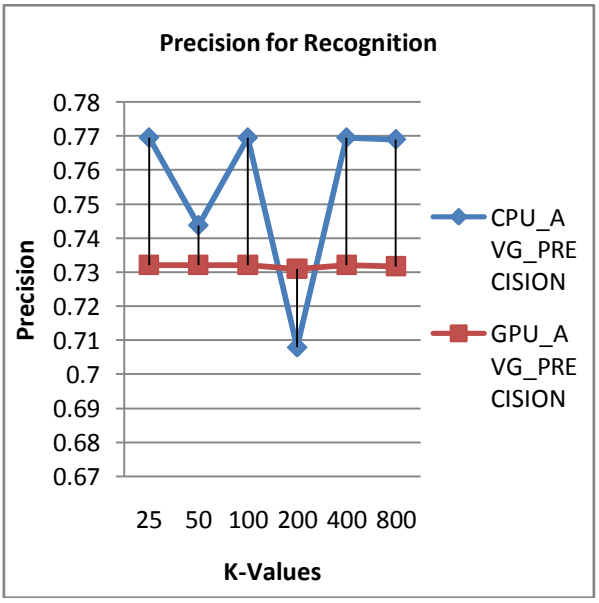
Graph 2 Average Recall for classification over different k-values



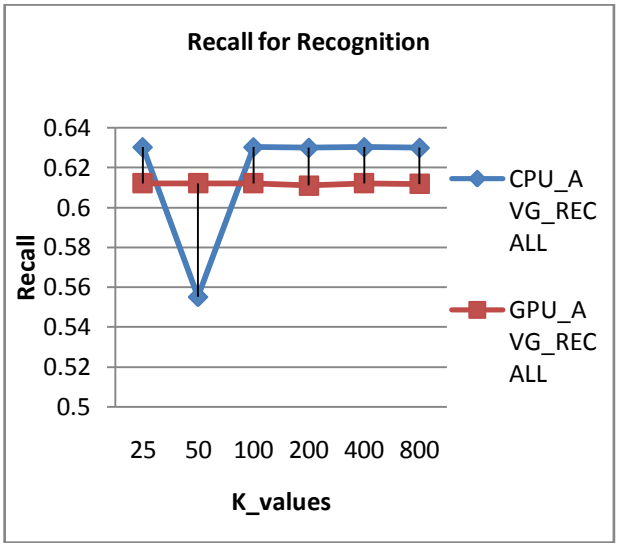
Graph 3 Average F1 Measure for classification over different k-values



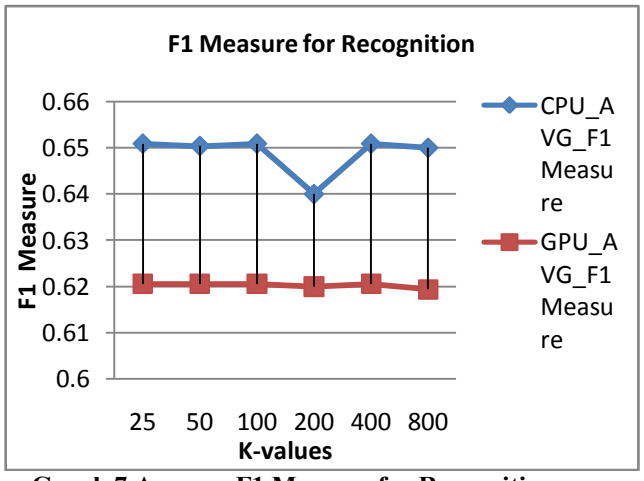
Graph 4 Average Accuracy for classification over different k-values



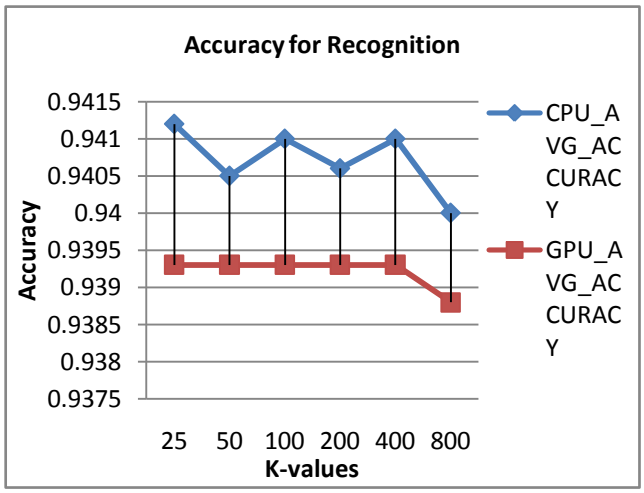
Graph 5 Average Precision for Recognition over different k-values



Graph 6 Average Recall for Recognition over different k-values



Graph 7 Average F1 Measure for Recognition over different k-values



Graph 8 Average Accuracy for Recognition over different k-values

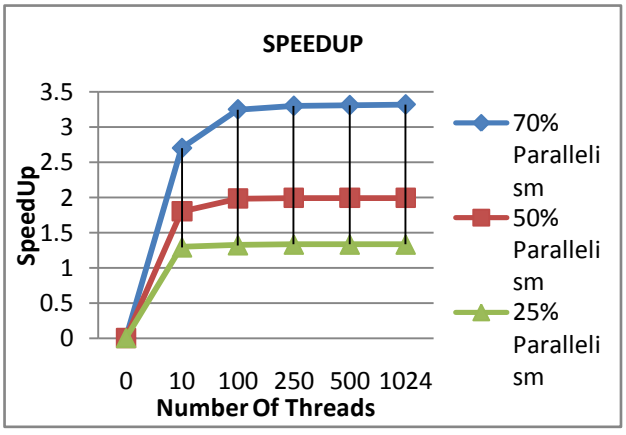
D. Speedup achieved by the GPU

The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program. Amdahl's law [26] is a model for the relationship between the expected speedup of parallelized implementations of an algorithm relative to the serial algorithm, under the assumption that the problem size remains the same when parallelized. The theoretical speedup that can be achieved by executing a given algorithm on a system capable of executing n threads of execution is:

$$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1)(B + \frac{1}{n}(1-B))} = \frac{1}{(B + \frac{1}{n}(1-B))}$$

Where,n-is number of threads,
B - The fraction of the algorithm that is strictly serial,
T(n) -The time an algorithm takes to finish when being executed on n thread(s) of execution.

A parallelized implementation of an proposed algorithm can run 70% of the algorithm's operations arbitrarily quickly (while the remaining 30% of the operations are not parallelizable), Amdahl's law states that the maximum speed up of the parallelized version is 1/(1 – 0.7) = 3.33 times as fast as the non-parallelized implementation..Parallel tasks include process of calculating term frequency, weight and cosine similarity. Sequential tasks include document preprocessing and the process assigning class test documents based on Knn ranking. Graph 9 shows the speed up for three portions of parallelism. The graph shows that speed up is not increased further even though thenumber of threads is increasing.



Graph 9 Speed up

(25% parallelism includes operations like calculating term frequency, 50% parallelism also includes process of calculating weight, and 70% parallelism includes operation for calculating term frequency, weight and similarity)

VII. CONCLUSION

For large datasets, the document classification on GPU reduces the execution time than on CPU. A parallel document classification based on KNN algorithm is faster for the k value of 50 & gives accuracy of 0.9993 tested for 2000 documents.

The execution time for GPU is approximate 800 seconds while on CPU it takes 2000 seconds. The execution time varies for varying K values for training & testing data set. Experiments on mentioned dataset show that the maximum speedup achieved by GPU implementation is 3.32.

ACKNOWLEDGMENT

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

.

REFERENCES

[1]VandanaKorde,CNamrataMahender,"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] RezaFarivar, Daniel Rebolledo, Ellick Chan, Roy Campbell "A Parallel Implementation of K-Means Clustering on GPUs", 2009.
[16] ErikLindholm, John Nickolls, Stuart Oberman "NvidiaTesla: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 Bayesfor 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.
[26]Amdahl, Gene M. (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". AFIPS Conference Proceedings (30): 483–485.
[27] Nikhil R. Vyawahare, S. G. Lade ." Document Classification using Parallel Processing ", Vol. 3 - Issue 2 (February - 2014), International Journal of Engineering Research & Technology (IJERT) , ISSN: 2278-0181 , www.ijert.org

Sangita G. Lade is working as Associate Professor in Vishwakarma Institute of Technology, affiliated to Pune University, Pune. She obtained his MTech degree from University of Pune. She has published extensively in International Journals, in the area of Data mining and text mining. She had guided M.Tech theses in Computer Science.

Nikhil R. Vyawahare, CSE-IT M.Tech Student atVishwakarma Institute of Technology, Pune, Maharashtra, India.

