A MapReduce based Parallel SVM for Email Classification

Ke Xu 1, 2, Cui Wen 3*, Qiong Yuan 4, Xiangzhu He 5, and Jun Tie 1*

1. College of Computer Science, South-Central University for Nationalities, Wuhan, China
2. School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
3. Computer Information Engineering Department, LuoYang Institute of Science and Technology, LuoYang, China
4. School of Computer Science, China University of Geosciences, Wuhan, China
5. College of Computer Science, South-Central University for Nationalities, Wuhan, China

Email: cole.xu@gmail.com

Abstract—Support Vector Machine (SVM) is a powerful classification and regression tool. Varying approaches including SVM based techniques are proposed for email classification. Automated email classification according to messages or user-specific folders and information extraction from chronologically ordered email streams have become interesting areas in text machine learning research. This paper presents a parallel SVM based on MapReduce (PSMR) algorithm for email classification. We discuss the challenges that arise from differences between email foldering and traditional document classification. We show experimental results from an array of automated classification methods and evaluation methodologies, including Naive Bayes, SVM and PSMR method of foldering results on the Enron datasets based on the timeline. By distributing, processing and optimizing the subsets of the training data across multiple participating nodes, the parallel SVM based on MapReduce algorithm reduces the training time significantly.

Index Terms—Email Classification; Parallel SVM; MapReduce

I. INTRODUCTION

With the rapidly development of the Internet and computer technology, the quantity of electronic data is in exponential growth. Data deluge has become a salient problem should be solved. Text categorization has been a highly popular machine learning application in the past decade. Gretarsson et al. present a web-based system for visual and interactive analysis of large sets of documents using statistical topic models [1]. A variety of other problem domains have been explored, including categorization by events, communication and even by author’s gender [2, 3, 4].

Data mining is the process of discovering new patterns from large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics and database systems. This problem has been researched by many scholars in all kinds of application area for many years and many data mining methods have been developed and applied to practice. However, most classical data mining methods out of reach in practice in face of big data. Computation and data intensive scientific data analyses are increasingly prevalent in recent years. Support Vector Machines (SVMs) [5] are powerful classification and regression tools, but their compute and storage requirements increase rapidly with the number of training vectors, putting many problems of practical interest out of their reach. Efficient parallel algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such large scale data mining analyses.

Efficient parallel algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such large scale data mining analyses. Many parallel algorithms are implemented using different parallelization techniques such as threads, Message Passing Interface (MPI), MapReduce [6], and mash-up or workflow technologies yielding different performance and usability characteristics [7]. Several MapReduce architectures are developed now [8, 9], including distributed and parallel computing techniques. SVM decomposition is another widespread technique for improving the performance in SVM training. Decomposition approaches work on the basis of identifying a small number of optimization variables and tackling a set of fixed size problems. Another widespread and effective practice is to split the training data into smaller fragments and use a number of SVM’s to process the individual data chunks. This in turn reduces overall training time. Various forms of summarizations and aggregations are then performed to process the final set of global support vectors. The most famous is the Google, but the source code is not open. Hadoop is the most popular open source MapReduce software. It has been adopted by many huge IT companies, such as Yahoo, Facebook, eBay and so on. The MapReduce architecture in Hadoop [10] doesn’t support iterative Map and Reduce tasks, which is required in many data mining algorithms. It will be the popular MapReduce architecture in cloud computing and can be used in data intensive data mining problems.

Email classification can be applied to several different applications, including filtering messages based on priority, assigning messages to user-created folders, or identifying SPAM. We will focus on the problem of

© 2014 ACADEMY PUBLISHER
doi:10.4304/jnw.9.6.1640-1647
assigning messages to a user's folders based on that user's folder strategy. One major consideration in the classification is that of how to represent the messages. Specifically, one must decide which features to use, and how to apply those features to the classification. Three types of features are defined to consider in email: unstructured text, categorical text, and numeric data. Relationship data is another type of information that could be useful for classification [11].

In this paper, we present a parallel scheme for scalable SVM training. We propose a parallel SVM based on MapReduce (PSMR) algorithm for email classification. We take the Enron email set as experiment dataset. The performance of the algorithm proposed in this paper is better than Naive Bayes (NB) classifier and one-by-one SVM classifier. We utilize a distributed computing framework, namely MapReduce using Hadoop's implementation.

II. RELATED WORKS

Computation and data intensive scientific data analyses are increasingly prevalent in recent years. Support Vector Machine based approaches have persistently gained popularity in terms of their application for text classification and machine learning [12, 13]. Classification in SVM based approaches is founded on the notion of hyperplanes [14]. Bickson et al [15] proposed a parallel implementation of an SVM solver using MPI. Zanghirati et al. [16] introduced a parallel implementation of SVM solver using MPI based on a decomposition technique that splits the problem into smaller quadratic programming sub problems. The outcomes of each sub problems are combined. That parallel solution can be used in peer-to-peer and grid environments, where there is no central authority that allocates the work. However, implementation in that paper using a synchronous communication model due to MPI's lack of support for asynchronous communication which could affect the speed of training time. SVM training is a computationally intensive process. Many SVM formulations, solvers and architectures for improving SVM performance have been explored and proposed including distributed and parallel computing techniques [17, 18]. A parallel SVM training algorithm was proposed in [19], and in which training multiple SVMs were performed using subsets of the training data, and then combined the classifiers into a final single classifier. The training data is then reallocated to the classifiers based on their performance and the process is iterated until convergence is reached. SVM is introduced the remote sensing extraction coastline in [20].

Email foldering is a rich and multi-faceted problem, with many difficulties that make it different from traditional topic-based categorization. Email users create new folders, and let other folders fall out of use. Email folders do not necessarily correspond to simple semantic topics, sometimes they correspond to unfinished to do tasks, project groups, certain recipients, or loose agglomerations of topics. It is also interesting to note that email content and foldering habits differ drastically from one email user to another, so while automated methods may perform well for one user, they may fail horribly for another. Furthermore, email arrives in a stream over time, and this causes other significant difficulties. Some email messages only make sense in the context of previous messages. Occasionally all messages in a thread should go to the same folder, but other times the topic in a thread drifts. The topic associated with a certain email folder can also shift over time. Reference [21] uses the Naive Bayes for constructing a real-world email foldering system that suggests three most appropriate folders for each incoming message.

Support Vector Machines are powerful classification and regression tools. Many SVM software models have been developed, such as lib-SVM, light-SVM, Is-SVM and so on. Lib-SVM is taken as the most efficient SVM model and widely applied in practice because of its excellent property [22]. The core of an SVM is a quadratic programming problem (QP), separating support vectors from the rest of the training data. For improving the training speed of SVM, many efforts have been done. Sequential Minimal Optimization (SMO) was used to select the work set to be optimized, which can simple the optimization problems markedly [23]. Graf et al proposed parallelization which splitting the problem into smaller subsets and training a network to assign samples to different subsets [8]. Reference [24] proposed a parallel SVM model based on hybrid MPI/OpenMP model for SVM training. A parallelization scheme was proposed where the kernel matrix is approximated by a block-diagonal [25]. Most of parallel SVM are based on MPI programming model. Little research work has been done with MapReduce work.

Based on current research work of SVM and MapReduce framework, the paper develops a parallel SVM model based on MapReduce. In this model, training samples are divided into subsections. Each subsection is trained with a SVM model. In this paper, lib-SVM is used to train each sub-SVM. The non-support vectors are filtered with sub-SVMs. The support vectors of each sub-SVM are taken as the input of next layer sub-SVM. The global SVM model will be obtained through iteration. The MapReduce based SVM model is encoded with Java language.

III. PARALLEL SVM BASED ON MAPREDUCE

MapReduce is a generic framework and programming model intended to abstract large scale computation challenges. For more efficiency, eliminating non-support vectors early from the optimization is proved to be an effective strategy for accelerating SVM. Using this concept we developed a filtering process that can be parallelized efficiently. After evaluating multiple techniques, such as projections onto subspaces (in feature space) or clustering techniques, we opted to use SVM as filters. This makes it straightforward to drive partial solutions towards the global optimum, while alternative techniques may optimize criteria that are not directly relevant for finding the global solution.
A. Architecture of Parallel SVM

There are many parallel algorithms with simple iterative structures. Most of them can be found in the domains such as data clustering, dimension reduction, link analysis, machine learning, and computer vision. These algorithms can be implemented with iterative MapReduce computation. The parallel SVM is based on the cascade SVM model. The SVM training is realized through partial SVMs. Each sub-SVM is used as a filter. This makes it straightforward to drive partial solutions towards the global optimum, while alternative techniques may optimize criteria that are not directly relevant for finding the global solution. Through the parallel SVM model, large scale data optimization problems can be divided into independent, smaller optimizations. The support vectors of the former sub-SVM are used as the input of later sub SVMs. The sub SVM can be combined into one final SVM in hierarchical fashion. The parallel SVM training process can be described as in Fig. 1.

![Figure 1. Training flow of parallel SVM](image)

In the architecture, the sets of support vectors of two SVMs are combined into one set and to be input a new SVM. The process continues until only one set of vectors is left. In this architecture a single SVM never has to deal with the whole training set. If the filters in the first few layers are efficient in extracting the support vectors then the largest optimization, the one of the last layer, has to handle only a few more vectors than the number of actual support vectors. Therefore, the training sets of each sub-problem are much smaller than that of the whole problem when the support vectors are a small subset of the training vectors. In this paper, lib-SVM is adopted to train each sub-SVM.

B. Parallel SVM based on MapReduce (PSMR)

MapReduce was popularized by the latter and primarily motivated by the need to be able to parallelize the processing of Internet scale datasets. Programmatically inspired from functional programming, at its core are two primary features, namely a map and a reduce operation. From a logical perspective, all data is treated as a Key ($k$), Value ($v$) pair. Multiple maps and reducers can be employed. At an atomic level however a map operation takes a ($k_1$, $v_1$) pair and emits an intermediate list of ($k_2$, $v_2$) pairs. A reduce operation takes all values represented by the same key in the intermediate list and processes them accordingly, emitting a final new list. Whilst the execution of reduce operations cannot start before the respective map counterparts are finished, all map and reduce operations run independently in parallel. Each map function executes in parallel emitting respective values from associated input. Similarly, each reducer processes different keys independently and concurrently.

MapReduce is a parallel programming paradigm, originally introduced by Google [6], whose central focus is to simplify the processing of large datasets on inexpensive cluster computers. The map function takes as input a set of key-value pairs, designated as $k_1$ and $v_1$, provided directly from the user-defined input files. Within the map function, the user specifies what to do with these keys and values. The map function outputs another set of keys and values, designated as $k_2$ and $v_2$. The reduce function sorts the key value pairs by $k_2$. All of the associated values $v_2$ are reduced and emitted as value $v_3$. The map and reduce functions are as follows:

\[
Map(k_1, v_1) \rightarrow [(k_2, v_2)] \quad (1)
\]

\[
Reduce(k_2, [v_2]) \rightarrow [v_3] \quad (2)
\]

At the MapReduce run-time level, the map operations are distributed by the master-server to the chunk-servers. The scheduler makes an effort to schedule computation on the same node where the data is stored. Meanwhile, other chunk-servers assigned to the reduce phase begin to take the ($k_2$, $v_2$) value pairs and sort them by $k_2$. These sorted arrays of $v_2$ values are passed to the reduce functions on these same assigned nodes. These outputs are finally saved on the GFS [6]. It is quite common for an application to string together many simpler MapReduce operations.

From the parallel SVM architecture, the pseudo program code based on MapReduce is as follows.

Firstly, computation nodes should be available. The program can be described as follows. Original large scale data $D$ should be split into smaller data sections $\{D_1, \ldots, D_n\}$. These data sections are put to computation nodes. Based on the available computation environment, Task-Initiation is used to configure the computation parameters, such as Map, Reduce, and Combine class names, number of Map tasks and Reduce tasks, partition file and so on. MapReduce-Driver will initiate the MapReduce task. Dynamic parameters will be transformed to each computation node through API interface.

In each computation node, Map tasks are operated. In the first layer of Fig. 1, sample data are loaded from local file system according to partition file. In the following layers, the training samples are support vectors of former layer. Lib-SVM is used to train each sub-SVM. In the Lib-SVM, Sequential Minimal Optimization is used to select the work set in decomposition methods for training support vector machines. C-SVC model is used to train classification SVM. Trained support vectors are sent to the Reduce tasks. In the Reduce task, all support vectors
of all Map tasks are collected together and feed back to client. Through iteration, the training process will stop when all sub-SVM are combined to one SVM.

Algorithm: PSMR pseudo-code
Preparation
Computation environment configuration;
Data partition and distribution to the computation nodes;
Create partition file;
Main class
TaskInitiation; /* configure the MapReduce parameters and class names */
MapReduceDriver; /* to initiate the MapReduce tasks */
While(condition) /* not combined to one SVM */
TaskInitiation; /* reconfigure the MapReduce parameters; */
MapReduceDriver; /* initiate new MapReduce tasks, broadcast combined support vectors to each computation node */
Get feedback results;
If(condition) break; /* if one SVM obtained, program finished */
End main class
Map class
If(the first layer SVM)
Load data from local file system;
else
Read data broadcasted by Main class;
End if
SvmTrain(); /* the parameters of the SVM model are transformed through taskInitiation */
Collector; /* sent the training result to Reduce task through message */
End Map class
Reduce class
Read data transformed from Map task;
Combine support vectors of each two subSVM into one sample set;
Collect; /* feedback all the trained support vectors */
End Reduce class

The time cost of SVM can be divided into following sections. The computation time complexity of libSVM is $O(n^2)$. The transformation time of data between Map and Reduce nodes is depend on the bandwidth of the connection network. The combination time cost of two SVMs is $O(n)$. When training data set is divided into m partitions, the computation cost is calculated as follows. The layers of cascade SVM is $N = \log_2 m$.

IV. EMAIL CLASSIFICATION USING PSMR

Email classification can be applied to several different applications, including filtering messages based on priority, assigning messages to user-created folders, or identifying SPAM. We will focus on the problem of assigning messages to a user’s folders based on that user’s foldering strategy.

A. Feature Extraction

As a special case of the general text categorization problem, let us define formally the email classification problem. Given a training set $D_c = \{(d_1, l_1), \ldots, (d_n, l_n)\}$ of labeled text documents, where each document $d_i$ belongs to a document set $D$, and the label $l_i$ of $d_i$ is selected from a predefined set of categories $C = \{c_1, \ldots, c_m\}$. The goal of text categorization is to induce a learning algorithm that, given the training set $D_c$, generates a classifier $\phi : D \rightarrow C$ that can accurately classify unseen documents.

So many design approaches for documents feature extraction in related research works. In this paper, we consider the traditional bag-of-words document representation that messages are represented as vectors of word counts. We use words as sequences of alphabetic, digit and underscore characters appearing anywhere in the email header or body. Words are sort descent, 50 most frequent words and words that appear only once in the training set are removed, and the remaining words are counted in each message to compose a vector.

B. Classification Accuracy Measure

As assumption above, the experiments consider the traditional classification accuracy, ratio of correctly classified instances to the total number of instances in the test set, as the macro-averaged performance (MAP), which is a conventional metric for evaluating classification method indicate text documents with labels compare to which category it belong to [4]. The system made decisions on each email document or message in dataset belongs to which group with respect to a specific category $l_i \in C = \{c_1, \ldots, c_m\}$, can be divided into four groups: True Positions ($TP_i$), False Positions ($FP_i$), True Negatives ($TN_i$) and False Negatives ($FN_i$), respectively. The corresponding evaluation metrics are defined as:

Global Precision:

$$P = \frac{\sum_{i \in C} TP_i}{\sum_{i \in C} (TP_i + FP_i)}$$

Global Recall:

$$R = \frac{\sum_{i \in C} TP_i}{\sum_{i \in C} (TP_i + FN_i)}$$

Macro-averaged performance:

$$MAP = \frac{2PR}{P + R}$$

V. EMAIL CLASSIFICATION EXPERIMENT

A. Email Dataset

We present experimental results of application parallel SVM based on MapReduce (PSMR) algorithm proposed in section 3 on the Enron datasets. A large set of email

Figure 2. Distribution of emails message per user
messages, the Enron corpus, was made public during the legal investigation concerning the Enron Corporation. The raw corpus is currently available on the web at http://www-2.cs.cmu.edu/~enron/. The dataset is provided by CMU after major clean-up and removal of attachments. The dataset version we use was released on February 3, 2004.

For cleaning the corpus for use in these experiments by removing certain folders from each user, we remove the non-topical folders "alldocuments", "calendar", "contacts", "deleted_items", "discussion_threads", "inbox", "notes_inbox", "sent", "sent_items" and "_sent_mail". In our cleaned Enron corpus, there are a total of 200,399 messages belonging to 158 users with an average of 691 messages per user. This is approximately one third the size of the original corpus. Fig. 2 shows the distribution of emails per user. The users in the corpus are sorted by ascending number of messages along the x-axis. The number of messages is represented in log scale on the y-axis. The horizontal line represents the average number of messages per user (691).

We then flatten all the folder hierarchies and remove all the folders that contain fewer than three messages. Since our goal in this paper is to explore how to classify messages as organized by a human, these folders would have likely been misleading. Although the size of the dataset is large, many users' folders are sparsely populated. We use the email directories of six former Enron employees that are especially large. Considering their personal privacy, we use anonymous names instead of their names follows as: User1, User2, User3, User4, User5 and User6. Each of these users has several thousand messages, with User1 having more than one hundred folders. We also remove the X-folder field in the message headers that actually contains the class label. Table 1 shows statistics on the seven resulting datasets.

### B. Experiment Results

As discussion above, this paper consider the macro-averaged performance (MAP) defined in section 4 as our evaluation method. However, in a general case, email messages can with certain probability belong to multiple folders, so it would be beneficial not to directly assign a message into one folder but rather to rank folders according to a confidence measure of the message belonging to each of these folders.

We apply Naive Bayes (NB), Support Vector Machine (SVM), and parallel SVM based on MapReduce (PSMR) Algorithm to each of the Enron datasets.

SVM was used first to classify the folder of each email based solely on a particular field of data from the email. The fields used were "From", "Subject", "Body", and "To, CC". The date field was not used, as it is not text information, and the problem of how to apply date information to email classification has not been fully explored. Next, SVM was used on each email treated as a single bag-of-words. This approach is labeled "All" in the analysis below. For this representation, the fields used in the previous experiments were concatenated and used in the classification. Thus, if the same term appears in both the subject and body of a message, it is considered to be multiple occurrences of the same feature. For the final approach, labeled "linear combination" below, the SVM scores from the "From", "Subject", "Body", and "To, CC" classifiers were combined linearly. The weights for each section were learned for each folder of a particular user, using ridge regression on the training data.

In order to create training and testing sets, the data for each user was sorted chronologically, and then split in half. The earlier half of the messages was used for training, while the later half was used for testing. Standard text parsing routines were applied to each of the fields in the email to produce the list of terms. Stemming was also performed on the body of the message. The terms were then given weights using the standard "ltc" formula, and given to SVM, using the one-vs-rest method for multi-class classification. For binary decisions, optimal thresholds were found for each folder (category). In many previous experiments, binary decisions were based on choosing only the highest ranked folder for each message, making precision the only relevant evaluation metric.

We report on experiments on Timeline: we calculate the accuracy for each training/test split and plot the accuracy curve over the number of training messages in the splits. If a test set contains messages of a newly created folder, so that no messages of this folder have been seen in the training data, then such test messages are ignored in the accuracy calculation.

Results on the six users Enron datasets are reported as the accuracy over the timeline in Fig. 3, where the x-axis is the training set size. Interestingly, results on some datasets do actually show a tendency of improvement over time. Such dataset is User3, and such are to some extent User1 and User5. The User6 dataset is a special case that is discussed in Section 4. The User5 phenomenon is presented there as well.

As shown in Fig. 3, those six sub-graph indicate accuracy over those six users on the timeline measure. In Fig. 3(a), compare to the other two methods, the parallel SVM based on MapReduce (PSMR) algorithm shows more well performance except some cases. The similar

| Table 1. Statistics on Enron datasets. After removing non-topical and small folders. |
|---------------------------------|-----|----------------|----------------|----------------|----------------|----------------|----------------|
| **User name** | **Number of folders per user** | **Message size of largest folder** | **Message size of smallest folder** | **Number of messages per user** | **Words size of largest message** | **Words size of smallest message** |
| User1 | 31 | 132 | 3 | 1892 | 2851 | 43 |
| User2 | 22 | 1019 | 5 | 3309 | 3270 | 41 |
| User3 | 42 | 571 | 4 | 4139 | 48296 | 46 |
| User4 | 11 | 1259 | 6 | 2598 | 4564 | 45 |
| User5 | 30 | 409 | 4 | 1196 | 19531 | 56 |
| User6 | 17 | 1298 | 3 | 2697 | 2278 | 50 |
results show in the other five sub-graph. The performance of Naive Bayes (NB) is much better than that of SVM in many case in Fig. 3(c) and Fig. 3(f). Tendency of accuracy of those three algorithms are similar as shown in Fig. 3(a), Fig. 3(b) and Fig. 3(d). As shown in Fig. 3(f), accuracy of those three algorithms is nearly to 1.0 after the training set size over 1400.

Table 2 shows the final results on Enron datasets - accuracies averaged over all the training/test splits, with the standard error of the mean. Depending on the level of complexity and homogeneity of each test set, the classification performance can significantly vary, causing relatively large standard errors for the averaged results.

<table>
<thead>
<tr>
<th>Enron user name</th>
<th>Naive Bayes</th>
<th>SVM</th>
<th>PSMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>31.2 ± 2.1</td>
<td>57.3 ± 1.9</td>
<td>48.6 ± 1.8</td>
</tr>
<tr>
<td>User 2</td>
<td>63.9 ± 1.6</td>
<td>74.9 ± 1.5</td>
<td>78.4 ± 1.2</td>
</tr>
<tr>
<td>User 3</td>
<td>36.5 ± 2.6</td>
<td>59.3 ± 1.6</td>
<td>55.6 ± 1.5</td>
</tr>
<tr>
<td>User 4</td>
<td>75.2 ± 1.3</td>
<td>82.6 ± 1.1</td>
<td>83.4 ± 1.6</td>
</tr>
<tr>
<td>User 5</td>
<td>57.6 ± 5.8</td>
<td>72.0 ± 4.2</td>
<td>73.4 ± 4.5</td>
</tr>
<tr>
<td>User 6</td>
<td>84.8 ± 2.2</td>
<td>93.1 ± 2.6</td>
<td>94.8 ± 1.8</td>
</tr>
</tbody>
</table>

From Table 2, we see that Naive Bayes, SVM and parallel SVM based on MapReduce (PSMR) differently on all the Enron datasets. On all the Enron datasets, PSMR demonstrates the highest accuracies, followed by SVM, then Naive Bayes. Despite that, the User 1 dataset is the only dataset on which the prevalence of PSMR over all the other classifiers is statistically significant. In two cases (User 1 and User 3), SVM outperforms PSMR—by notable 9% in the case of User 1, but the difference is not statistically significant. On the User 6 dataset, Naive Bayes shows the best performance, but again the difference is not statistically significant. The performance of PSMR is similar to that of SVM in three cases (User 1, User 4 and User 3). On two cases (User 1 and User 3), the difference of averaged accuracies of Naive Bayes and SVM is not statistically significant though the size of training set is different.
In eleven out of the six cases, the Naive Bayes classifier is significantly inferior to the other two classifiers. It is widely believed that Naive Bayes is not the optimal solution for text categorization [19]. However, the performance of Naive Bayes could likely be improved by applying feature selection and/or a more sophisticated smoothing method than Laplace.

Generally speaking, the timeline curves show unstable, spiky behavior. One might expect the classification performance improve when the training set size increases, but this rarely happens in practice. It can be explained by the observation that email is usually related to other recently received email, rather than to email received long ago. Thus, old email in the training set probably does not affect the classification procedure. The classification results are surprisingly low. They show the real complexity of the task of categorizing email, as opposed to the regular topical text classification. They are obtained on the realistic time-based evaluation setup. If we applied random training/test splits (method that is standard for the regular text classification), we would obtain much higher results.

VI. CONCLUSION

There are many more ways to email classification model than the methods attempted in this paper. More research needs to be done into using the relationships between emails to reinforce knowledge about a particular message. SVM is taken as a most efficient classification and regression model. The computation cost of SVM is square proportion to the number of training data. Classical SVM model is difficult to analyze large scale practical problems. Parallel SVM can improve the computation speed greatly. In this paper, parallel SVM model based on iterative MapReduce is proposed.

In this paper we have presented a parallel SVM algorithm for email classification. By splitting the training set and applying distributed computing techniques such as MapReduce we can improve the training time considerably. However, this has varying yet noticeable degrees of accuracy degradation. In our work, we employ ontology based semantics to improve the accuracy of the parallel SVM based on MapReduce (PSMR). Email foldering is a rich and interesting task. It differs from the standard (topical) text classification in its highly subjective and non-monotonic folding schema. Folders are constantly being created and abandoned, becoming more active and less active, and even their major common topics are changing over time. In addition, users tend to folder some messages by sender, some by event, and some by date and in many cases the logic behind a certain foldering choice can be difficult to discern. Therefore, an email foldering system should be adaptive to the working style of individual users.

One of these relationships is thread membership. While a small amount of research has been done into how to detect threads, no one has studied how to use the threads for the task of email classification. Time information was also left out of these experiments, while it seems clear that it could be useful. Time cannot be used in the same way as other fields though, obviously, so work must be done to determine how time affects the foldering strategies of a user. For future work, we intend to research appropriate schemes to extract additional intelligence from annotated instances and employ this within the machine learning, parallel SVM feedback loop process. We believe that accuracy can be also further improved via automated annotation.

ACKNOWLEDGEMENTS

This research is supported by “the Fundamental Research Funds for the Central Universities”, South-Central University for Nationalities (CZY14010, CYZ12008), and is also supported by National Science and Technology Project of the Ministry of Education of China (MCM20121061).

REFERENCES

[13] A. Blanco, A. M. Ricket, and M. Martin-Merino, “Combining SVM Classifiers for Email anti-spam Filtering,” Computational and Ambient Intelligence,


Ke Xu (1981- ) is a PHD Candidate in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. He is currently an instructor in the College of Computer Science, South-Central University for Nationalities. His research interests include data mining, cloud computing and information retrieval.

Wen Cui is an instructor in the Computer Information Engineering Department, LuoYang Institute of Science and Technology. He is one of the Corresponding author.

Qiong Yuan is a PHD Candidate in the School of Computer Science, China University of Geosciences, Wuhan, China. She is currently an instructor in the College of Computer Science, Donghu Institute. She is one of the Corresponding author.

Jun Tie is currently an associate professor in the College of Computer Science, South-Central University for Nationalities. He is one of the Corresponding author.