IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4965

Distributed Data Mining and its Applications to Intelligent Textual Information Processing

Shibin Qiu
Department of Electrical and Computer Eng., University of New Mexico, sqiu@unm.edu

Mei Qiu
Emcore Corporation, Albuquerque, NM, mei_qiu@emcore.com

ABSTRACT

Textual information processing is of fundamental importance, due to the massive amount of documents, especially online textual information that we need to process every day. In this paper, we study data mining techniques applied to intelligent textual information processing in distributed environments, including text classification, information extraction (IE) and topic detection and tracking (TDT). These intelligent processing techniques will improve the quality and efficiency of information resource management and utilization. Their statistical models and computational algorithms challenge the researches in data mining and distributed/parallel computing. When successfully applied, they will help enhance and benefit applications in IT, digital library, and information retrieval. Specifically, we study the distributed computing of the following algorithms: naïve Bayes classifier combined with expectation-maximization (EM) for text classification, hidden Markov model for information extraction, and deterministic annealing with EM for topic detection and tracking. We also study the performances of the proposed algorithms and experiment on the improvements.

INTRODUCTION

Text classification is designed to classify documents into predefined classes. One example of its applications is the automatic classification of Internet news articles. Information extraction (IE) is the process of filling the fields in a database by automatically extracting sub-sequences of the documents. IE can be used, for example, to extract names of acquired companies and their prices in a newswire article regarding company takeovers. It can also be used to extract job titles from job posting documents. Topic detection and tracking (TDT, also called novelty detection) is a dynamic classification problem where topics are identified and clustered based on contents of documents and events are clustered on time to build temporal class hierarchy. For instance, when news reports on new events appear, they are different from existing classes and a new class in the hierarchy is added. These textual processing functions are called intelligent processing because they are different from traditional processing capacities such as text matching and searching. When these intelligent processing functionalities are integrated into an information system, they enable better utilization of text documents, provide more effective retrieval of digital documents, and promote new and creative uses of the resources to support knowledge-based inference and problem solving. They can also leverage much more services to end users with rich features and more efficient management of information resources for an enterprise.

Since documents in organizations are distributed over the Internet and over large geographical locations, distributed computing must be considered in order to apply these intelligent methods in practice. These textual processing algorithms involve theories from artificial intelligence, machine learning and data mining, and demand large amount of computational efforts. When distributed, they are even more challenging. We review related works in the following before we present our studies in detail.

Statistical algorithms based on Bayesian theorem are the most popular solution for text classification, though some other approaches such as support vector machine are also used [12, 14]. Naïve Bayesian classifiers are trained with labeled documents. When the available number of labeled documents is limited, naïve Bayesian algorithm is combined with the EM learning algorithm to estimate statistical parameters using unlabeled documents [14]. Due to the huge amount of documents and their rapid proliferation, naïve Bayesian combined with EM learning (NB-EM) have been extensively used in researches and production systems. Large amount of textual information also requires substantial processing power. When single processing node is not enough, parallel solution is the natural choice. When documents are not available in a centered site, distributed computing must be considered.

Classification based on data clustering has been parallelized and achieved substantial speedups [5][15]. A parallel classification algorithm based on decision tree was studied in [7] for mining large data sets. Parallelism for mining association rules has been studied [1]. Belief networks for probabilistic inference have been implemented in parallel [8]. A parallel implementation for the NB-EM algorithm was proposed and had achieved satisfactory speedups for small clusters [9]. Active learning using query-by-committee has been applied for text classification [11].

An IE algorithm based on hidden Markov model has been studied and experiments have been carried out to extract name and price fields for company takeovers and speaker, time, and location fields for seminar information with satisfactory precisions and recalls [6]. A graphical statistical model has been used for IE and web data extraction [3,13] with experiments on extracting job title and course schedules. A hierarchical, probabilistic TDT model has been studied and deterministic annealing was used to build the dynamic class hierarchy [2]. News articles from Reuters newswire service and closed-caption transcripts of CNN broadcast have been tested [2]. Activity monitoring and its applications to monitor and predict changes in a stock market have been studied [4].

Related algorithms have been studied and tested on sequential bases and successes of variable degrees have been achieved. To apply these algorithms in real world, we study the issues related to implementations of these algorithms in distributed environments. The issues involved in our distributed systems, as in most other distributed systems, include performance, fault tolerance, security, synchronization, etc. In this study, we focus on performance issues by studying computing times and communication traffics to reduce costs and achieve optimal performances. We present the basic sequential algorithms and design their distributed schemes that are most appropriate for our applications. We test our algorithms with simulations on a Linux cluster. Other design issues will be studied in future work.

Section 2 describes the naïve Bayes text classification algorithm and its distributed computing. Due to space limit, we describe the classification algorithms in more detail and present the IE and TDT algorithms briefly. Section 3 presents the IE algorithm based on HMM

This conference paper appears in the book, Innovations Through Information Technology, edited by Mehdi Khosrow-Pour. Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

and its distributed scheme. Section 4 formulates the distributed TDT algorithm. Experiments are included in each section. Section 5 presents the integration algorithm that brings the three functions together to save time and communication costs. Section 6 concludes the paper.

TEXT CLASSIFICATION

In this section, we describe the naïve Bayes classifier, expectation-maximization algorithm applied to use unlabeled documents, and their distributed scheme.

Document description

Preprocessing on the documents including stemming, stop word removal is performed before they are used for training and classification. The attribute value presentation is used to represent document parameters, where each word is an index (or key), and the probability estimation (frequency) is the attribute [12][14]. Suppose our documents are divided into *M* classes, $C = \{c_1, c_2, ..., c_M\}$ and we have *N* training documents $D = \{d_1, d_2, ..., d_N\}$, each having a class label. We need a table of *N* rows and *V* columns (*V* is the size of the vocabulary) to store the frequencies of the words. This table can be prohibitively large and compact data structures such as hash table can be used to save space and speed up search. We also need a list to store the estimation of class distribution.

Naïve Bayes classifier

The naïve Bayesian classifier is based on the generative probabilistic model [12], which assumes that each document is generated from a mixture of components with probability distribution θ. If a document d_i was generated by a mixture component c_j, then we assign the label of c_j to d_i, $y_i = c_j$. The likelihood of document d_i is given by,

$$P(d_i \mid \theta) = \sum_{j=1}^{M} P(c_j \mid \theta) P(d_i \mid c_j; \theta), \quad (1)$$

where $P(c_j \mid \theta)$ is the class prior probability, with which document d_j is first created by selecting a mixture component. $P(d_i \mid c_j; \theta)$ is the distribution with which d_i is generated by the selected mixture component c_j.

If document d_i contains words $\{w_{i1}, w_{i2}, ..., w_{|d_i|}\}$, suppose conditional independence, then $P(d_i \mid c_j; \theta)$ can be expressed as,

$$P(d_i \mid c_j; \theta) = P(w_{i1}, w_{i2}, ..., w_{i|d_i|} \mid c_j; \theta),$$

$$= P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{ik} \mid c_j; \theta) \quad (2)$$

The conditional probabilities $\theta_{w_k|c_j} = P(w_k \mid c_j; \theta)$ and the prior class probabilities $\theta_{c_j} = P(c_j \mid \theta)$ are the parameters that we need to estimate. Collectively the parameters are represented as,

$$\theta = \{\theta_{w_1|c_1}, ..., \theta_{w_V|c_M}; \theta_{c_1}, ..., \theta_{c_M}\}. \quad (3)$$

The estimation $\hat{\theta}$ of θ can be accomplished as follows using Laplace smoothing. For the word probability estimate $\hat{\theta}_{w_k|c_j}$,

$$\hat{\theta}_{w_k|c_j} = \frac{1 + \sum_{i=1}^{N} N(w_k, d_i) P(y_i = c_j \mid d_i)}{V + \sum_{s=1}^{V} \sum_{i=1}^{N} N(w_s, d_i) P(y_i = c_j \mid d_i)} \quad (4)$$

where $N(w_k, d_i)$ is the number of times that word w_k occurs in document d_i and $P(y_i = c_j \mid d_i)$ is class label of d_i. And the class prior probabilities are estimated as,

$$\hat{\theta}_{c_j} = P(c_j \mid \hat{\theta}) = N(d_i, c_j) / N, \quad (5)$$

where $N(d_i, c_j)$ is the number of training document d_i that are classified to class c_j.

Once we get the estimate $\hat{\theta}$, the naïve Bayesian classifier uses the generative model in a reverse way to infer the mixture component from which a given document is most likely to have been generated. Based on the maximum a posteriori (MAP) method, document d_i is classified to class c_{j^*}, if j^* achieves P^* by maximizing the following,

$$P^* = \arg\max_j P(y_i = c_j \mid d_i; \hat{\theta}), \quad (6)$$

and

$$P(y_i = c_j \mid d_i; \hat{\theta}) \propto P(c_j \mid \hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{ik} \mid c_j; \hat{\theta}). \quad (7)$$

Expectation-maximization learning

Naïve Bayesian classifier works fine if enough labeled training examples are available. When the labeled training data is not enough, however, it produces poor quality of classification. Labeled training is sometimes costly. But the number of unlabeled documents is abundant. The EM algorithm uses only a small number of labeled examples and produce satisfactory classifications by utilizing unlabeled data [12][14].

EM algorithm iteratively applies MAP estimations on incomplete data to get a local optimum. First, it learns the parameter estimation $\hat{\theta}$ using the naive Bayesian method on the labeled data D^l. Second, the naïve Bayesian algorithm, equation (7), is used to label each unlabeled document, from the unlabeled data set D^u, yielding a probability-weighted class label (soft label), $P(c_j \mid d_i; \hat{\theta})$. Third, based on the newly labeled and the originally labeled documents, the parameter of the data set $D = D^l \cup D^u$ is estimated and $\hat{\hat{\theta}}$ is obtained. Now the second step is repeated, but with $\hat{\theta}$ being replaced by $\hat{\hat{\theta}}$. Steps two and three are repeated until $\hat{\hat{\theta}}$ is close enough to $\hat{\theta}$. The MAP estimation of θ can equivalently be obtained by maximizing the log-posterior objective function,

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

$$\log P(\theta \mid D) =$$

$$\sum_{d_i \in D^l} \log(P(y_i = c_j \mid \theta) P(d_i \mid y_i = c_j; \theta)) +$$

$$\sum_{d_i \in D^u} \log \sum_{j=1}^{M} P(c_j \mid \theta) P(d_i \mid c_j; \theta) + \log P(\theta) \quad (8)$$

By introducing the class indicator variable *Z*, where $z_{ij} \in Z$ is one indicating that d_i comes from class c_j, and zero otherwise, (8) can be written as,

$$\log P_c(\theta \mid D; Z)$$

$$= \sum_{d_i \in D^l \cup D^u} \sum_{j=1}^{M} z_{ij} \log(P(c_j \mid \theta) P(d_i \mid c_j; \theta))$$

$$+ \log P(\theta) \quad (9)$$

We do not know the values of z_{ij}. Therefore, we use expected values of z_{ij} computed on current parameters in (9), and get an approximation of (8). The parameter estimation $\hat{\theta}$ thus obtained is a local maximum and can be used as substitute for the global optimal estimation. Let $\hat{Z}^{(k)}$ and $\hat{\theta}^{(k)}$ denote the estimates for θ and Z at iteration k. Based on data set *D*, the local maximum log-posterior can be found iteratively with the following two steps.

E-step: Set $\hat{Z}^{(k+1)} = E[Z \mid D; \hat{\theta}^{(k)}]$, (7), for soft labels;

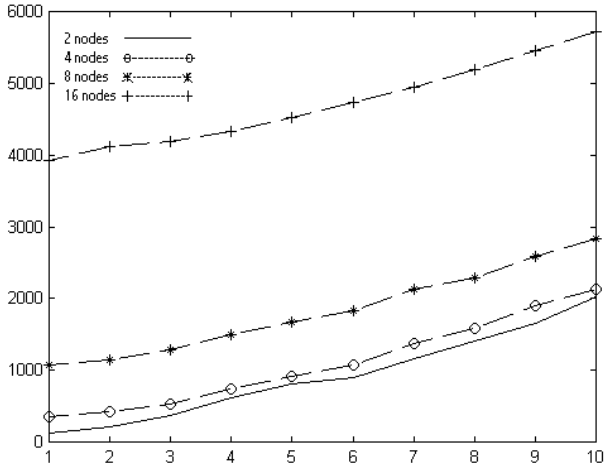
M-step: Set $\hat{\theta}^{(k+1)} = \arg\max_{\theta} P(\theta \mid D; \hat{Z}^{(k+1)})$, to estimate $\hat{\theta}$, with (4) and (5) using soft labels. When the parameters are trained classification can be done with (7).

Distributed classification

In a distributed environment, an organization (such as an international company) has documents across large geographical locations. We need to train the classifier accounting for all the documents so that classification in the organization is consistent. One possible solution is for each distributed site to collect and make available all training documents and train on them to have global classifier. This scheme would make the training on each site simple. But synchronizing the training documents consumes too much communication cost. The second approach is for each site to train on its local documents and all the sites combine their local parameters to obtain a global classifier. Parameters have less volume compared with training documents and hence have less communication cost.

Still another method is hierarchical where distributed sites are grouped based on geographical proximity and classifiers are synchronized in each group before global classifier is synchronized. One advantage of this hierarchical method is that local classifiers can be computed and made available for classification before global parameters are reached. Another advantage of this grouped scheme is that in case that the central server is down, the group based classifier is still available. When each group is connected via a high speed connection (in the case where they are located in the same metro area), local computation can be parallelized and can be sped up substantially. We tested the performance through a simulation on a Linux cluster with newsgroup data set [12,14]. Figure 1 shows the times used to train the classifier parameters

Figure 1. Times spent in distributed classification



including times of training on each node, sending them to the central server, and combining them on the central server to derive the global classifier. Average node-to-node delay is 300 ms (the delay between Albuquerque, USA and Barcelona, Spain). The x-axis is the total volume of documents in the training set on each node in 100 mega bytes and the y-axis denotes the times spent in seconds. It can be seen that when the number of nodes increases, the time consumed increased (the vertical distances between the curves became larger) almost quadratically. This is because that the parameters are two dimensional and combining them takes quadratic times.

INFORMATION EXTRACTION

In this section, we study how to apply HMM for information extraction in distributed systems. A HMM (hidden Markov model) is a Markov model whose states are not directly observable and can only be inferred from observations. HMM has been applied successfully to solving a number of language related problems, including speech recognition, named entity extraction, and text segmentation.

IE with HMM

Assume we use an HMM to extract a company's acquisition price from an article reporting the related takeover event, we can model the price tokens (words) as from the target model and the rest as from the background model. Thus a given document can be viewed as being generated by a stochastic process that first emits some tokens from the background model, then emits tokens from the price model (target model), and then switches back to tokens from the background model [6]. We extract the symbols associated with the optimal state sequence and fill in the field in the database. Specifically, we need to find the state sequence $s = \{s_s, s_2, ..., s_t\}$ that maximizes the probability,

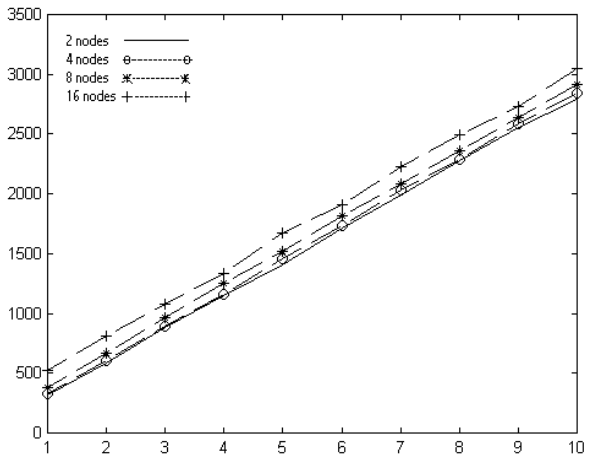
$$P(s_1, s_2, ...s_t \mid w, \theta), \quad (10)$$

where $w = \{w_1, w_2, ...w_{|d_i|}\}$ is the word observation, θ is the model parameter.

We build one model for each field to be extracted. The transition and emission probabilities of a model can be trained using the labeled documents. Laplace smoothing is applied to calculate the frequencies, as in (4). The Viterbi's algorithm is used to solve the training and inferences in polynomial time. Shrinkage and EM can also be applied to improve model quality with a smaller number of labeled documents. To test a given document, we treat the tokens in the document as observations and find the state sequence that has most probably generated the observations (10).

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 2. Time costs for distributed IE



Distributed IE

In a distributed environment, each node has partial documents for training and extraction. We let each node extract and fill in part of a database table, and send them to a central server that combines the partial tables to form the whole table. Both central server and grouped hierarchy can be used as in the case of text classification (section 2). Figure 2 shows the total times spent by the distributed IE algorithm, simulated on a cluster. Average node-to-node delay is also 300 ms. The x-axis is the total volume of documents in the training set on each node in 100 mega bytes and the y-axis denotes the total times spent in seconds. It can be seen that when the number of nodes increases, the time consumed increased about linearly. This is due to the fact that combining the partial tables of a database needs approximately linear time with respect to the number of nodes, faster than combining the two dimensional parameters for classification and TDT.

TOPIC DETECTION AND TRACKING

Topic detection and tracking (TDT), also called novelty detection, is a dynamic classification problem. It combines clustering and classification, and accounts for both content and temporal relationships among documents. In this, section we study the TDT algorithm and its distributed computing.

TDT algorithm

A set of topics in a collection of documents is described by a hierarchy of classes that clusters based on content and contains temporal information representing changes of events over time. To build the hierarchy, we can use the classification algorithm (7) in section 1. A better way is to use deterministic annealing (DA), which usually converges to a better optimum [2,17]. DA views document clustering as a random process where the class is chosen randomly according to $P(c_j \mid d_i)$. It optimizes the parameters of the random process and maintains the entropy of the distribution $P(c_j \mid d_i)$. DA thus smoothes the surface on which EM is hill-climbing. The generalized posterior given by DA is,

$$P(c_j \mid d_i;\hat{\theta}) \propto P(c_j \mid \hat{\theta})\left(\prod_{t=1}^{|V|} P(w_t \mid c_j;\hat{\theta})^{N(w_t,d_i)}\right)^{1/T}, \quad (11)$$

where T is the temperature.

To build the temporal relationship in the topic hierarchy, each document is clustered separately in time within each topic as a mixture of Gaussian distribution. We denote the probability with which a document belongs to a certain topic $P(c_j \mid d_i)$, as ρ_{ji}, which is held constant during our temporal clustering process. The probability that a document in a class belongs to a time slot can be formulated as

$$P(e_{jk} \mid d_i, c_j) \propto P(e_{jk} \mid c_j) P(\tau_{d_i} \mid e_{jk})^{1/T} \quad (12)$$

$$P(\tau_{d_i} \mid e_{jk}) \propto \frac{1}{\sqrt{2\pi}\sigma_{jk}} e^{-(\tau_{d_i}-\mu_{jk})^2/2\sigma_{jk}^2}$$

where τ_{d_i} is the time stamp of document d_i, and τ_{jk} is the k^{th} event of class c_j. The parameters can be estimated using the EM method. In the E-step, $P(e_{jk} \mid d_i, c_j)$ is calculated. In the M-step, $\hat{\xi}$, $\hat{\mu}$ and $\hat{\sigma}$ are computed according to (13), (14) and (15) based on estimates of $P(e_{jk} \mid d_i, c_j)$.

$$\hat{\xi}_{jk} = P(e_{jk} \mid c_j) \propto \sum_{i=1}^{|D|} \rho_{ji} P(e_{jk} \mid d_i, c_j) \quad (13)$$

$$\hat{\mu}_{jk} \propto \sum_{i=1}^{|D|} \rho_{ji} P(e_{jk} \mid d_i, c_j) \tau_{d_i} \quad (14)$$

$$\hat{\sigma}_{jk} \propto \sum_{i=1}^{|D|} \rho_{ji} P(e_{jk} \mid d_i, c_j)(\tau_{d_i} - \mu_j)^2 \quad (15)$$

After the documents have been temporally clustered using (12), we can compute the total probability that a document belongs to a specific event within a topic as,

$$P(e_{jk} \mid d_i) = \alpha_{ijk} P(e_{jk} \mid d_i, c_j), \quad (16)$$

where α_{ijk} is the probability of the parent topic of event e_{jk}.

Distributed TDT

The distributed TDT computing we designed is for each node to build its own topic and event hierarchy and send it to a central server for combining. Hierarchies are represented by tree structures. The combined hierarchy shows a whole picture of the news stories with temporal relationships across the global organization. Its time performance shows similar patterns as that of distributed classification.

INTEGRATION

The three processing functions of text classification, information extraction and topic detection and tracking can be integrated together to save training time and communication costs. For each document we need to train, the parameters of the three sets of models can be computed one after the other before the document file is closed. When we send the parameters to a central server, we can send them together with the rows of the extracted table in one packet to save traffic. The integrated algorithm is shown in Figure 3.

Combining the tables in a database can be implemented via database system functions (e.g., those provided by SQL Server) or using proprietary network software. Figure 4 shows the time performance of the integrated processing. It shows that the total time is less the sum of the parts.

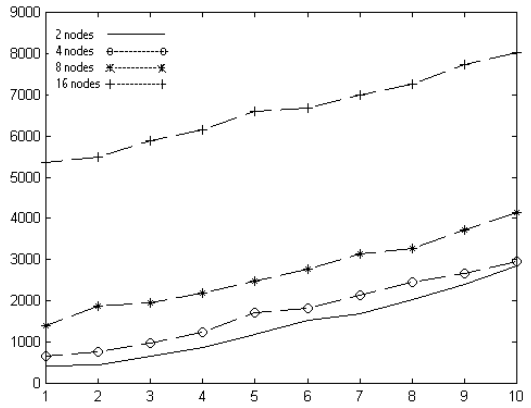
Using a central server exhibits a single point of failure. We can provide fault tolerance by deploying a backup server back to back with

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

Figure 3. Integrated processing

```
Integrated_processing()
{
 For each document in training set
          Train parameters for classifier, IE and TDT;
 Build classifier parameters, P;
 Extract fields, T ;
 Build topic hierarchy, H;
 Send P, T and H to central server;
 Central server combines for global T, global H, global P;
 Central server sends global P and H to all sites;
}
```

Figure 4. Performance of integrated pro



the central server. We thus have a simple fault tolerance solution.

The algorithms have been tested with variable degree of accuracies. The text classification has been tested and achieved 85% accuracy on the newsgroup dataset [12,14]. The information extraction algorithm has achieved 71% accuracy, on average, with the seminar information and company takeover articles [6]. The TDT has worked with precision and recalls of about 81% on the Reuters newswire and close-caption from CNN broadcast news [2].

CONCLUSIONS

To better manage and utilize information resource for an organization, we have studied intelligent text processing algorithms including text classification, information extraction, and topic detection and tracking. In order to cope with the distribution of information resources across large geographical locations, we have studied distributed computing of the algorithms. Experiments have showed performance improvements through simulation on a computing cluster. The proposed distributed algorithms are economical with respect to communication cost and computing time. They also provide easy means of implementation for fault tolerance.

For our future work, we will explore more statistical models for intelligent text processing to improve processing quality in distributed environments.

REFERENCE

[1] Agrawal, R., and Shafer, J. C., Parallel mining of association rules. *IEEE Transaction on Knowledge and Data Engineering*, 1996.

[2] Baker, L., D., Hofmann, T., McCallum, A., K., Yang, M., A hierarchical probabilistic model for novelty detection in text, *Proc. of 16th Int'l Conf. On Machine Learning*, 1999.

[3] David M. Blei, J. Andrew Bagnell, Andrew K. McCallum, Learning with scope, with application to information extraction and classification, *Proc. UAI 2002*, pp. 53-60, Aug., 2002, Canada.

[4] Fawcett, T., Provost, F., Activity Monitoring: Noticing Interesting Changes in Behavior, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 53-62, 1999.

[5] Forman, G., and Zhang, B., Linear speed-up for a parallel non-approximate recasting of center-based clustering algorithms, including k-means, k-harmonic means, and EM. *KDD Workshop on Distributed and Parallel Knowledge Discovery*, 2000.

[6] Dayne Freitag, Andrew K. McCallum, Information extraction with HMMs and shrinkage, *Proc. AAAI-99 workshop on machine learning for information extraction*, July, 1999, Orlando, Florida, USA.

[7] Joshi, M.V., Karypis, G., and Kumar, V. ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets. *In Proceedings of International Parallel Processing Symposium*, 1998.

[8] Kozlov, A., V., Singh, J., P., A parallel Luatitzen-Spiegelhalter algorithm for probabilistic inference, *Proc. of 1994 Conf. Supercomputing*, Washington, D.C., 1994.

[9] Kruengkrai, C., Jaruskulchai, C., A parallel learning algorithm for text classification, *The 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, July 23 - 26, Edmonton, Alberta, Canada, 2002.

[10] McCallum, A., and Nigam, K. A comparison of events models for naive Bayes text classification, *Proc. of 98 the AAAI Workshop*, pages 41-48, 1998.

[11] Andrew K. McCallum, Kamal Nigam, Employing EM and pool-based active learning for text classification, *Proc. ICML 1998*, pp. 350-358, Wisconsin, USA.

[12] Mitchell, T. *Machine Learning* (McGraw-Hill,1997).

[13] Tom Mitchell, Kamal Nigam, Sean Slattery, Learning to extract symbolic knowledge from the world wide web, *Proc. 1998 National Conference on Artificial Intelligence*, July 1998, Madison, Wisconsin, USA.

[14] Nigam, K., McCallum, A., Thrun, S., Mitchell, T., Text classification from labeled and unlabeled documents using EM, *Machine Learning* (Kluwer Academic Publishers, 2000).

[15] Patanè, G., and Russo, M., Distributed Unsupervised Learning Using the MULTISOFT Machine, *Information Sciences*, vol. 43 no.1-4, 2002.

[16] Rabiner, L., A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE 77(2).*, Feb 1989, pp.257-286.'

[17] Rose, K., Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of IEEE*, 86(11), pp.2210-2239, 1998.

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.

0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/proceeding-paper/distributed-data-mining-its-applications/32375

Related Content

Continuous Assurance and the Use of Technology for Business Compliance
Rui Pedro Figueiredo Marques (2018). *Encyclopedia of Information Science and Technology, Fourth Edition (pp. 820-830).*
www.irma-international.org/chapter/continuous-assurance-and-the-use-of-technology-for-business-compliance/183795

Complexity Analysis of Vedic Mathematics Algorithms for Multicore Environment
Urmila Shrawankar and Krutika Jayant Sapkal (2017). *International Journal of Rough Sets and Data Analysis (pp. 31-47).*
www.irma-international.org/article/complexity-analysis-of-vedic-mathematics-algorithms-for-multicore-environment/186857

Massive Open Online Courses (MOOCs) and the Technologies That Support Learning with Them
Jeremy Riel and Kimberly A. Lawless (2015). *Encyclopedia of Information Science and Technology, Third Edition (pp. 7529-7537).*
www.irma-international.org/chapter/massive-open-online-courses-moocs-and-the-technologies-that-support-learning-with-them/112454

Modeling Rumors in Twitter: An Overview
Rhythm Walia and M.P.S. Bhatia (2016). *International Journal of Rough Sets and Data Analysis (pp. 46-67).*
www.irma-international.org/article/modeling-rumors-in-twitter/163103

Covering Based Pessimistic Multigranular Approximate Rough Equalities and Their Properties
Balakrushna Tripathy and Radha Raman Mohanty (2018). *International Journal of Rough Sets and Data Analysis (pp. 58-78).*
www.irma-international.org/article/covering-based-pessimistic-multigranular-approximate-rough-equalities-and-their-properties/190891