TagLearner: A P2P Classifier Learning System from Collaboratively Tagged Text Documents

Haimonti Dutta*, Xianshu Zhu†, Tushar Mahule†, Hillol Kargupta†, Kirk Borne§, Codrina Lauth‡, Florian Holz¶ and Gerhard Heyer¶
*The Center for Computational Learning Systems, Columbia University, New York, USA
haimonti@ccls.columbia.edu
†Department of CSEE, University of Maryland, Baltimore County, MD, USA
{xianshu1,tusharm1}@umbc.edu, hillol@cs.umbc.edu
§George Mason University, Fairfax, VA, USA
kborne@gmu.edu
‡Fraunhofer Institute for Intelligent Analysis and Information Systems, Knowledge Discovery Group, Germany
codrina.lauth@iais.fraunhofer.de
¶Department of CS, University of Leipzig, Germany
{holz, heyer}@informatik.uni-leipzig.de

***Abstract*—The amount of text data on the Internet is growing at a very fast rate. Online text repositories for news agencies, digital libraries and other organizations currently store giga- and tera-bytes of data. Large amounts of unstructured text poses a serious challenge for data mining and knowledge extraction. End user participation coupled with distributed computation can play a crucial role in meeting these challenges.**

In many applications involving classification of text documents, web users often participate in the tagging process. This *collaborative* tagging results in the formation of large scale Peer-to-Peer (P2P) systems which can function, scale and self-organize in the presence of highly transient population of nodes and do not need a central server for co-ordination. In this paper, we describe TagLearner, a P2P classifier learning system for extracting patterns from text data where the end users can participate both in the task of labeling the data and building a distributed classifier on it. We present a novel distributed linear programming based classification algorithm which is asynchronous in nature. The paper also provides extensive empirical results on text data obtained from an online repository - the NSF Abstracts Data.

I. Introduction

An increasing number of websites are offering large unstructured text documents. Online newspapers (e.g. `www.cnn.com`, `www.bbc.com`), digital libraries (e.g. IEEE, ACM, Google digital libraries), product review sites (e.g. `www.cnet.com`)are some examples.Most of these sites do not offer any mechanism for automated text classification which is useful for personalization and indexing of documents in a large corpora. Traditionally, such categorization or indexing is either performed by an authority, such as a librarian, or else derived from the material provided by the authors of the documents [13]. However, since document repositories are constantly increasing in size, manual classification is impractical.

Collaborative tagging has grown in popularity because of its ability to recruit web users to tag shared text and images, especially when there are too many documents for a single authority to classify. Unfortunately, most online large text repositories are not designed to support such collaborative efforts. There exists little support for the users to annotate the data and more importantly develop data mining capabilities based on the collective effort.

This paper presents a P2P classifier learning system which allows users to collaboratively label web documents and learn from them. We leverage the benefit obtained from peer-to-peer architectures and describe mechanisms for collaborative tagging and efficient distributed knowledge discovery from these annotated data. In prior work ([5], [6], [4]) we have shown that distributed linear programming can be used effectively for resource management in Peer-to-Peer networks. Our contributions in this paper are as follows: (1) We present TagLearner, an architecture for building a P2P classification system for collaboratively tagged text data. (2) We describe the distributed *linear* programming algorithm for classification on a P2P networks. To the best of our knowledge, this is the first *distributed* algorithm for solving classification as a linear program in-network without constraint centralization. (3) We present extensive empirical analysis of the algorithm by illustrating its performance on the NSF abstracts database.

This paper is organized as follows. Section II presents related work; Section III gives the overview of the proposed P2P classifier learning system; Section IV shows how the problem of classification can be posed as a linear program; Section V presents the distributed algorithm; Section VI provides experimental results and finally Section VII concludes the paper.

II. Related Work

This section presents related literature on collaborative tagging and representation of data mining problems as linear programs.

Collaborative tagging: Collaborative tagging allows users to add semantically meaningful information in the form of tags to the shared documents [8]. It has been applied to improve web search result [14], recommendation system quality [12], [16] and clustering accuracy [1], [9]. In this paper, the user can only choose from predefined tags to label the documents. Moreover, tags created by users are stored on local machines and shared with only with their peers. There is no central server managing these tags. Finally, users work collaboratively to learn a classification model.

Distributed Linear Programming: Mathematical Programming has been used extensively in data mining for the purpose of feature selection and supervised and unsupervised learning ([10], [15]). One of the most popular algorithms for linear programming is the simplex algorithm [3]. The main competitors of simplex are a group of methods known as interior point methods ([7] and the references therein). Several other parallel implementations of the linear optimization algorithms exist – Stunkel and Reed [2] consider two different approaches of parallelization of the constraint matrices on the hypercube: (1) Column partitioned simplex and (2) Row partitioned simplex. Column partitioned simplex has been studied further in work done by Yarmish [18]. In all of the above mentioned techniques, the stress is on splitting a given problem on different machines to reach the solution faster by utilizing techniques for parallelization. Ours is an inherently distributed problem. Furthermore, to the best of our knowledge, the above parallel optimization algorithms have not been adapted for use in data mining algorithms.

III. TagLearner: A P2P Classifier Learning System

TageLearner, our P2P classifier learning system (Figure 1) allows users to label text and learn a classifier collaboratively. The system is composed of a service-provider and client-side browser plugins. The service-provider keeps track of the documents in the repository and can register a peer-to-peer classifier learning service by creating a tagging group. The client-side browser plugin provides two functionalities: (1) It provides an interface for users to join or leave the tagging group and annotate the web documents. (2) It implements the distributed classifier learning algorithm, which allows every user within the same tagging group to collaboratively classify the document. Users willing to learn the classifier can register on the service-provider, download the browser plugin and choose any tagging group to join. Registered users can start tagging documents using the predefined tags and communicating with their group members. The learnt model will be stored as a service and new incoming documents can be automatically classified.

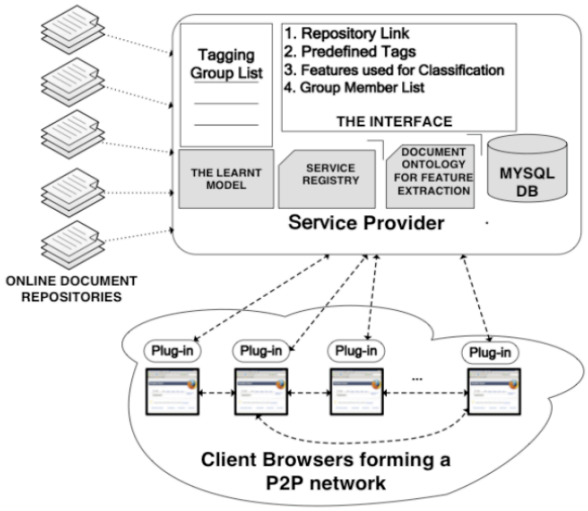

Figure 1. **The architecture of TagLearner.**

The components of TagLearner are as follows:

- The service-provider is composed of an interface and a MYSQL database. The interface allows users to register a classifier learning service The service-provider will maintain a *tagging* group for this service.
- Users wanting to participate in learning a classifier can download and install the browser plugin, create an account on the service-provider and login. They are allowed to join a tagging group of their choice and can select one of the following options:
  1) Website Link for the online repository
  2) Predefined Tags – Users can only use these tags to assign class labels to the document.
  3) Features for classification: The service-provider keeps ontologies for documents in different categories. The features can be obtained from these ontologies. Documents can be converted to feature vectors using these ontologies, which ensures that every user in a tagging group uses the same feature set.
- Users in the same tagging group can learn the classifier by running a distributed *linear* programming algorithm for classification.
- The learnt classifier will be available at the service provider.

In the following section, we will show how the classification task can be posed as a linear program and solved in-network without constraint centralization.

IV. Classifier Design by Linear Programming

Let T be a data set with P instances. Each instance contains a vector of N features (denoted by **x**) and a

categorical or continuous valued target variable (denoted by **y**). It is assumed that there exists an underlying function **f** such that $\mathbf{y} = \mathbf{f}(\mathbf{x})$ for each instance (**x**,**y**) in the training set. The goal of a supervised learning algorithm is to find an approximation **H** of **f** that can be used to predict values of unseen instances in the test set. In order to find an approximation, one possibility is to obtain a *weighting* of the feature vector which is positive for patterns in the positive class and negative for patterns in the other class (assuming a binary classification problem) [15]. Thus, if the k^{th} instance is represented by $x_k = [x_{1k}, x_{2k}, \cdots, x_{Nk}], k = 1, 2 \cdots P$, and W is the weight vector of N weights represented by $W = [w_1, w_2, \cdots w_N]^T$, then we are interested in a $\hat{W}$ such that $x_k \hat{W} \geq 0$ for all instances in the positive category and negative for instances in the other class. Let e_k represent the error associated with instance x_k, π_k represent the weighting coefficient. Then total error over P instances can be expressed as: $\hat{e} = \Sigma_{k=1}^{P} \pi_k e_k$ and error function for a given instance is obtained by estimating $e_k = -(x_k W - d)$, when $x_k W < d$; and 0, when $x_k W \geq d$. In order to find the desired $\hat{W}$, the error function $\hat{e}$ should be minimized. Smith [15] shows that the above can be formulated as a linear programming problem, with the constraint matrix as follows: $\chi W + E = D + S$ where $\chi = [x_1 x_2 \cdots x_P]^T$ where T represents the transpose, $E = (e_1, e_2, \cdots e_P)^T$ is the error vector, $D = (d, d, \cdots, d)^T$ and $S = (s_1, s_2, \cdots s_P)$ is the vector of slack variables which allow the inequalities to be made equalities. The objective function of the linear program can be written as: $\hat{e} = \Pi^T E$ where $\Pi = (\pi_1, \pi_2, \cdots, \pi_P)$. To minimize $\hat{e}$, it is convenient to transform the minimization to a equivalent maximization problem: $z = \Pi^T D - \Pi^T E$ which can be further simplified as $z = \Pi^T \chi W - \Pi^T S$. The above linear program can be solved by using the Simplex Algorithm[1] [3]. Note that in this paper we are primarily concerned with problems that are linearly separable.

V. The Distributed Linear Programming Algorithm

In a Peer-to-Peer network, typically the nodes have their own local datasets comprising of the feature vectors and class labels. From these, local constraint matrices can be constructed as described in Section IV. However, solving the objective function based on local constraints does not ensure that constraints at other nodes are satisfied. Thus nodes in the network need to communicate with one another in order to ensure that the solution to the objective function takes into consideration all the constraints in the network. Our distributed linear classification algorithm has two main steps: (1) A pre-processing step for obtaining the canonical representation of a linear system and (2) Obtaining the solution to the objective function.

[1]We refrain from giving an extensive description of the simplex algorithm due to space restrictions.

Distributed Canonical Representation of the Linear System: An important pre-processing step before solving a distributed linear program is the development of an algorithm for obtaining the canonical representation. In order to do so, each node in the network needs to have access to the number of basic variables [3] that it should add which is equal to the total number of constraints in the system. We propose the following converge cast based approach: Let s be an initiator node which builds a minimum spanning tree on all nodes in the network. Following this, a message is sent by s to all its neighbors asking how many local constraints each node has. A neighbor on receiving this message, either forwards it to its neighbors (if there are any) or sends back a reply. At the end of this procedure, Node s has the correct value of the total number of constraints in the system, say T_c. Next, Node s sets a variable $count - Constraint$ to the number of its local constraints. It traverses the minimum spanning tree and informs each node visited of the number of constraints seen so far. Let T represent the value of $count - Constraint$ at node i. Then node i must add T_c basic variables to each of its constraints. At the end of this procedure, all nodes have added the relevant basic variables. Note that this procedure creates exactly the same canonical form as would have been obtained if all the constraints were centralized. It must be noted that the Distributed Canonical Representation algorithm needs to be run only once at the time of initialization. Thereafter, each node just updates its tableau [3] depending on the pivots chosen at that round of iteration.

Notation and Preliminaries: Let $P_1, P_2, \cdots P_\eta$ be a set of nodes connected to one another via an underlying communication tree such that each node P_i knows its neighbors N_i. Each node P_i has its own local constraints which may change from time to time depending on the resources available at that node. The constraints at node i have the form $A^i X^i = b^i$ where A^i represents an $m \times n$ matrix, X^i is a $n \times 1$ vector and b^i is a $m \times 1$ vector. Thus at each node, we are interested in solving the following linear programming problem: Find $X^i \geq 0$ and Min z^i satisfying $c_1 x_1 + c_2 x_2 + \cdots c_n x_n = z^i$ subject to the constraints $A^i X^i = b^i$. The global linear program (if all the constraint matrices could be centralized) can be written as follows: Find $X \geq 0$ and Min z satisfying $c_1 x_1 + c_2 x_2 + \cdots c_n x_n = z$ subject to constraints $AX = B$ where $A = \bigcup_{i=1}^{\eta} A^i$ and $B = \bigcup_{i=1}^{\eta} b^i$. Next, we present an exact algorithm for solving linear optimization using the simplex method. Our assumption is that each node contains different sets of constraints, but has knowledge of the global objective function.

The Algorithm: At the beginning of iteration l, a node P_i has its own constraint matrix and the objective function. The column pivot, henceforth referred to as $col - pivot^i$, is that column of the tableau corresponding to the most

negative indicator[2] of $c_1, c_2, \cdots, c_n$. Following this, each node forms the row ratios ($r_j^i, 1 \leq j \leq m$) for each row i.e. it divides $b_j^i, 1 \leq j \leq m$ by the corresponding number in the pivot column of that row. Let minimum of r_j^i's be presented as $row - pivot^i$. This is stored in the history table of node P_i corresponding to iteration l. Now the node must participate in the distributed algorithm for determination of the minimum row ratio i.e. Minimum $(row - pivot^i), i \in N_i$. We describe a simple protocol called *Push-Min* for computing this. At all times t, each node maintains a minimum $m_{t,i}$. At time t=0, $m_{t,i} = row - pivot^i$. Thereafter, each node follows the protocol given in Algorithm 1. When the protocol Push-Min terminates, each node will have the exact value of the minimum $row - pivot^i$ in the network [11]. Once the Push-Min protocol converges, the node containing

Algorithm 1 Protocol Push-Min

1. Let $\{\hat{m}_r\}$ be all the values sent to i at round $t-1$.
2. Let $m_{t,i} = \min(\{\hat{m}_r\}, row - pivot^i)$
3. Send $m_{t,i}$ to all the neighbors.
4. $m_{t,i}$ is the estimate of the minimum in step t

the minimum $row - pivot^i$ (say P_{min}) will send its row in the simplex tableau to all other nodes in the network. Next node P_i updates its local tableau with respect to the extra row it received from node P_{min}. The algorithm, Constraint Sharing Protocol is described in Table 2. Completion of one round of the CS-Protocol, ensures that one iteration of the distributed simplex algorithm is over.

Termination: In a termination state, two things should happen: (1) No more messages traverse in the network (2) Each local node has all its $c_i > 0$. Thus the state of the network can be described by information possessed by each node. In particular, each node will have a solution to the linear programming problem. Note that this solution converges exactly to the solution if all the constraints were centralized. If the distributed system were dynamic with nodes joining and leaving on an ad-hoc basis, the constraint matrix for nodes currently in the network will change and so will the objective function to be solved. We are currently investigating approaches to solve this problem using dynamic linear programming approaches.

VI. Experimental Results

This section presents the experimental results for the performance of the distributed classifier learning algorithm from collaboratively tagged text documents using NSF abstracts text collections. The experiments were carried out using the Distributed Data Mining Toolkit(DDMT[3]). We assume that a number of users already joined in the same tagging group.

[2]Note that if no negative indicator is found, then this is the final simplex tableau

[3]www.umbc.edu/ddm/wiki/

Algorithm 2 Constraint Sharing Protocol (CS-Protocol)

1. Node P_i performs protocol Push-Min until there are no more messages passed.
2. On convergence to the exact minimum, the minimum $row - pivot^i$ is known to all nodes in the grid.
3. All the nodes use the row obtained in Step 2 to perform Gauss Jordan elimination on the local tableau.
4. At the end of Step 3, each node locally has the updated tableau and completes the current iteration of the simplex algorithm.

They have installed the browser plugin and have tagged some text documents. The tagged text documents have been converted to feature vectors. The DDMT toolkit will only simulate the distributed classifier learning process. Since to the best of our knowledge no prior work exists for solving linear programming using local, asynchronous, distributed algorithms in P2P environments, we are unable to compare this work with other distributed algorithms.

A. Dataset Description

NSF Abstracts data set is available from the UCI Machine Learning Repository and is called the "NSF Research Awards Abstracts 1990-2003". It contains 129000 abstracts describing NSF awards for basic research. Abstracts are labeled based on the PI's department. We consider abstracts belonging to only Earth and Mathematical sciences. There are 7218 instances in Earth sciences and 9781 instances in Mathematical sciences.

B. Representing Text using Feature Vectors

Abstracts are represented using a bag of words, extracted using Indexer Tool [17]. We create a m-dimensional feature vector for each abstract. "Important" terms are selected from the abstracts which correspond to features for our feature vector. Here, a "important" [17] term is defined as a word that not only appears frequently in one single abstract but also appears frequently in the document collection.

Note that the features used for classification do not rely on collaboratively generated annotations. As mentioned in section III, users in the same tagging group will be given the same feature set from the P2P server.

C. Network Topology

The distributed environment is simulated using the DDM Toolkit. The network topologies are generated using BRITE[4]. The Waxman model is used for generating the network structure *i.e.* the probability that nodes u and v have an edge between them is given by the formula $P(u,v) = \alpha e^{-d/(\beta L)}$ where $0 < \alpha, \beta \leq 1$, d is the Euclidean distance from node u to node v, and $\overline{L}$ is the

[4]http://www.cs.bu.edu/brite

$NetworkSize$	$TrainSetSize$	$TestSetSize$	$Error\%$
30	300	150	19.8
50	500	250	15
80	800	400	12.6
100	1000	500	11.2

Table I
CLASSIFICATION ERRORS OBTAINED IN P2P NETWORKS OF VARYING SIZES FOR THE NSF DATA.

maximum distance between any two nodes. Each peer in the network can tag a portion of the abstract locally. We assume that there is an appropriate browser plugin installed in end users' browser which allows a distributed classifier learning algorithm running on each peer.

D. Results and Discussion

To measure the performance of the distributed algorithm, the amount of data transferred in bytes was calculated for both the centralized[5] and distributed scenarios. Let c be the total number of constraints, v be the number of variables per constraint equation and assume that each real number is represented by 4 bytes, the amount of data that needs to be transferred for centralization is calculated as: $c*(v+1)*4$. However, in the distributed case, both messages transferred for estimating the pivot and the final pivot row at the end of an iteration of simplex algorithm should be considered *i.e.* If μ number of messages are exchanged to estimate the row pivot, then the total communication cost per simplex iteration will be: $4*(\mu+v+1)$. We first vary the number of nodes in the network between 10 and 100 for NSF dataset, keeping the number of constraint equations and variables fixed at 10. Figure 2 show that the distributed algorithm has less communication cost compared to the centralized version.

For estimating the classification error in-network, we run the distributed simplex algorithm on a given training set T_{train} and solve the global weight vector (say W). Let T_{test} represent the test set. Using W, we estimate $T_{test}*W$ and threshold[6] the values at 0.5. The Table I illustrates the classification error for different network sizes and varying number of train / test instances; in general it is observed that the larger the training set, the higher the classification accuracy. Note that this distributed classification algorithm is an exact algorithm and hence the classification accuracies for both centralized and distributed versions are same.

In order to study the effect of varying the number of variables and constraints per node in the network, we set the network size as 50 nodes and vary the number of edges between 50 - 200. Figure 4 shows the effect of variation of

[5]Note that this is a hypothetical scenario. We assume that the constraints from all nodes in the network can be centralized and use it as a baseline for comparing the performance of the distributed algorithm.

[6]Values greater than 0.5 get class label 1 and less than 0.5 gets class label 0.

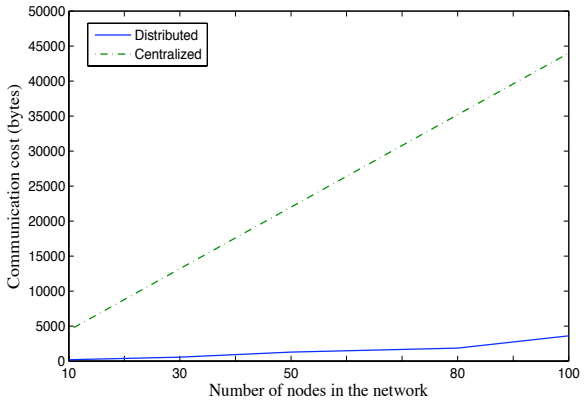

Figure 2. Communication cost versus the number of nodes in the network

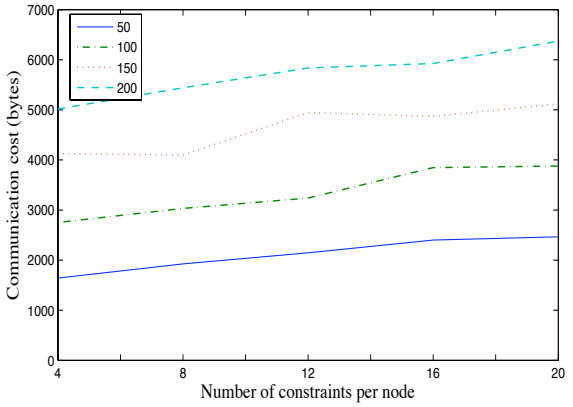

Figure 3. Communication cost versus the number of constraints per node

the number of variables in a constraint equation at a node. With the increasing number of variables, the slight increase in the graph can be attributed to the larger size of the row transmitted at the end of the simplex iteration. Figure 3 shows the effect of varying the number of constraints per node. As the number of constraints increase, only the local computation at each node to find the row pivot increases. This does not affect the number of messages being passed in the network, due to which the communication cost curve has a relatively flat nature.

These experiments provide an insight into the scalable nature of the distributed linear programming-based classification algorithm.

VII. CONCLUSIONS AND FUTURE WORK

A large amount of unstructured text data is available on the Internet and online repositories. Mining such data to find interesting patterns can be done efficiently in a distributed way and with the help of user participation. Peer-to-Peer(P2P) networks form an ideal platform for collaborative activities where end user participation plays an important

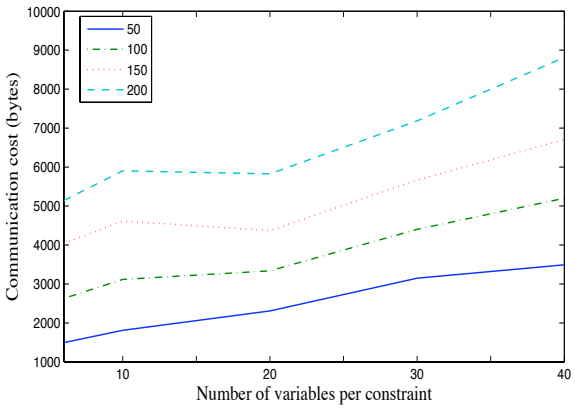

Figure 4. Communication cost versus number of variables per constraint

role. This paper presents a P2P classifier learning system prototype and a scalable distributed classification algorithm based on linear programming. To the best of our knowledge, this is the first distributed classification algorithm based on Linear Programming. We present empirical results for testing the accuracy and scalability of the algorithm on NSF abstracts data. Our results indicate that increasing the number of variables at a node does not affect communication cost significantly, but increase in constraints can increase the communication cost, depending on how hard the optimization problem becomes. Future work includes extension of the classification algorithm for quadratic programming, multi-class problems and dealing with dynamic network structures.

VIII. Acknowledgments

This work is supported by the NASA (NNX07AV70G) and AF (MURI 2008-11) grants.

References

[1] Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM.

[2] S. Craig and D. Reed. Hypercube Implementation of the Simplex Algorithm. In *Association of Computing Machinery (ACM)*, pages 1473–1482, 1988.

[3] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, NJ, 1963.

[4] H. Dutta and A. Matthur. Distributed optimization strategies for mining on peer-to-peer networks. In *Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on*, pages 350–355, Dec. 2008.

[5] Haimonti Dutta. *Empowering Scientific Discovery by Distributed Data Mining on the Grid Infrastructure*. PhD thesis, University of Maryland, Baltimore County (UMBC), 2007.

[6] Haimonti Dutta and Hillol Kargupta. Distributed linear programming and resource management for data mining in distributed environments. In *ICDM Workshops*, pages 543–552, 2008.

[7] R. M. Freund and S. Mizuno. Interior point methods: Current status and future directions. In H. Frenk, editor, *In High Performance Optimization*, pages 441–446. Kluwer Academic Press, 2000.

[8] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems, Aug 2005.

[9] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based social interest discovery. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 675–684, New York, NY, USA, 2008. ACM.

[10] O. L. Mangasarian. Linear and nonlinear seperation of patterns by linear programming. *Operations Research*, 13(3):444 – 452, May 1965.

[11] B. Mayank, G. M. Hector, G. Aristides, and R. Motwani. Estimating aggregates on a peer-to-peer network. Technical report, Computer Science Department, Stanford University, 2004.

[12] Satoshi Niwa, Takuo Doi, and Shinichi Honiden. Web page recommender system based on folksonomy mining for itng '06 submissions. In *ITNG '06: Proceedings of the Third International Conference on Information Technology: New Generations*, pages 388–393, Washington, DC, USA, 2006. IEEE Computer Society.

[13] J.E. Rowley and J. Farrow. *Organizing knowledge*. Gower Aldershot, Hampshire, England, 2000.

[14] L. V. S. Lakshmanan S. A. Yahia, M. Benedikt and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. In *Proc. VLDB*, pages 710–721, 2008.

[15] F. W. Smith. Pattern classifier design by linear programming. *IEEE Transactions on Computers*, C-17(4):367 – 372, April 1968.

[16] Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999, New York, NY, USA, 2008. ACM.

[17] H.F. Witschel. Terminology extraction and automatic indexing – comparison and qualitative evaluation of methods. In *Proc. of Terminology and Knowledge Engineering (TKE)*, 2005.

[18] G. Yarmish. *A Distributed Implementation of the Simplex Method*. PhD thesis, Computer and Information Science,Polytechnic University, 2001.