Automatic Document Organization in a P2P Environment

Stefan Siersdorfer, Sergej Sizov
{stesi,sizov}@mpi-sb.mpg.de

Max-Planck Institute for Computer Science

Abstract. This paper describes an efficient method to construct reliable machine learning applications in peer-to-peer (P2P) networks by building ensemble based meta methods. We consider this problem in the context of distributed Web exploration applications like focused crawling. Typical applications are user-specific classification of retrieved Web contents into personalized topic hierarchies as well as automatic refinements of such taxonomies using unsupervised machine learning methods (e.g. clustering). Our approach is to combine models from multiple peers and to construct the advanced decision model that takes the generalization performance of multiple 'local' peer models into account. In addition, meta algorithms can be applied in a restrictive manner, i.e. by leaving out some 'uncertain' documents. The results of our systematic evaluation show the viability of the proposed approach.

1 Introduction

Motivation Text processing using machine learning algorithms (e.g. using supervised methods such as classification or unsupervised algorithms like clustering) is an important part of many Web retrieval applications. As an example, we may consider a focused crawler [7] that starts with the sport-related topics 'ball games', 'track and fields', and 'swimming' that are initially filled by the user with some training data. Based on these training data the parameters of a mathematical decision model can be derived, which allows the system to automatically classify web pages gathered during the crawl into the topic taxonomy. In the next step, a large crawl would populate the topics of interest. In the postprocessing phase, unsupervised machine learning methods (e.g. clustering) may be applied for automatic organization of the 'ball games' documents by partitioning this class into appropriate subtopics (say 'soccer', 'basketball', and 'handball').

The key to success for the classification step clearly lies in the selection of an appropriate amount of human labeled training samples, being the intellectual bottleneck of the system. The clustering step should provide high accuracy in the sense that whatever subclasses it forms should indeed be reasonably homogeneous.

In the context of a peer-to-peer (P2P) network that puts together multiple users with shared topics of interest, it is natural to aggregate their knowledge and

construct better machine learning models that could be used by every network member. The naive solution would be to share available data (training samples and/or results of the focused crawl) along a higher number of peers with others. However, the following reasons may prevent the peer from sharing all of its data with other members of the overlay network:

- significantly increased network costs for downloads of additional training data on every peer
- increased runtimes for the training of the decision models
- privacy, security, and copyright aspects of the user's personal information sources

Contribution To overcome the limitations of single-peer models, we propose the application of *meta methods*. Our objective is to combine multiple independently learned models from several peers and to construct the advanced decision model that utilizes the knowledge of multiple P2P users.

In addition we show how meta learning can be applied in a *restrictive* manner, i.e. leaving out some documents rather than assigning them to inappropriate topics or clusters with low confidence, providing us with significantly more accurate classification and clustering results on the remaining documents.

Related Work Focused Web exploration applications were intensively studied in the recent literature. The idea of focused crawling [7] was recently adopted for P2P systems. However, this work was mainly focused on sharing crawling results from particular peers (e.g. using distributed indexes) rather than improving the underlying crawler and its components.

On the other hand, there is a plethora of work on text classification and clustering using all kinds of probabilistic and discriminative models [7]. The machine learning literature has studied a variety of meta methods such as bagging, stacking, or boosting [5, 29, 19, 13], and even combinations of heterogeneous learners (e.g., [30]). There are also methods available for combining different clustering methods [26, 12, 24]. The approach of intentionally splitting a training set for meta classification has been investigated by [8, 25]. However, these techniques were not considered in the context of P2P systems.

Algorithms for distributed clustering are described in [16, 18], but here document samples must be provided to a central server, making these solutions inconsistent with our requirements. The distributed execution of k-means was discussed in [11]. However, this method requires multiple iterations that must be synchronized among the peers and causes a considerable amount of coordination overhead. Privacy-preserving distributed classification and clustering were also addressed in the prior literature: In [27] a distributed Naive Bayes classifier is computed; in [20] the parameters of local generative models are transmitted to a central site and combined, but not in a P2P system.

2 System Architecture

The implementation of a peer in our distributed system consists of two layers. The *lower (network) layer* determines the communication among the peers. The peers form an autonomous agent environment: the exact way one particular peer solves its Web retrieval problem (e.g. crawling the Web, sending queries to 'Deep Web' portals, analyzing recent newsgroup discussions or publications in electronic journals, etc.) is not restricted in any way. We assume that all peers share the same thematic taxonomy such as *dmoz.org* [2]. The *upper (application) layer* is the distributed algorithm that utilizes results from particular peers to construct improved learning models (e.g. classification and/or clustering models) that can be used to continue the focused crawl with higher accuracy and to adjust the topics of a user-specific personalized ontology.

In our model, the peers use the epidemic-style communication [10]. Every peer maintains an incomplete database about the rest of the network. This database contains entries (e.g. addresses) on some other peers (neighbors) together with timestamps of the last successful contact to that neighbor. The neighbor list is refreshed using a push-pull epidemic algorithm.

To connect a new peer to the network one needs only one living address. The database of the new peer is initialized with the entry containing this living address only, and the rest is taken care of by the epidemic algorithm. Removal of a peer does not require any administration at all.

When new data becomes available, the peer initiates the building of a new meta learning method together with its direct neighbors as described in Section 3.1. With the next epidemic messages, it is broadcast to all neighboring peers.

3 Properties of the Application Layer

In this section we first describe a general framework for aggregating information from k peers in meta models, and then consider two typical applications for such a framework: classification and clustering for document collections.

3.1 Exchanging Data Models among Peers

In our framework we are given a set of k peers $P = \{p_1, \ldots, p_k\}$. Each peer p_i maintains its collection of documents D_i. In the first step, each peer p_i builds a model $m_i(D_i)$ using its own document set D_i. Next, the models m_i are propagated among the k peers as described in Section 2. To avoid high network load, it is crucial for this step that the models m_i are a very compressed representation of the document sets D_i. Each peer p_i uses the set of received models $M = \{m_1, \ldots, m_k\}$ to construct a meta model $Meta_i(m_1, \ldots, m_k)$. From now on, p_i can use the new meta model $Meta_i$ (instead of the 'local' model m_i) to analyze its own data D_i.

We notice that the dynamic nature of P2P overlay networks has no direct impact on the construction of meta models. If the participating nodes do not receive models $M^F = \{m_{f_1}, \ldots, m_{f_u}\}$ from some (failed) neighbors, they are still able to construct the meta model $Meta_i^*(M - M^F)$ on models obtained from the remaining live peers. When the number k of required models is explicitly given by estimators or tuning parameters of the framework (Section 3.2), the multicast capability of the network layer can be combined with advanced scheduling methods [14] in order to reach the desired number of live nodes in presence of failures.

3.2 Application to Automatic Document Organization

Meta Classifiers on k Peers In the context of classification algorithms, the introduced general approach 3.1 can be substantiated as follows. Each peer p_i contains a document collection D_i, consisting of a set of labeled training documents T_i and unlabeled documents U_i. The peer's goal is to automatically classify the documents in U_i. In the first step, every peer p_i builds its own feature vectors of topic labeled text documents T_i (e.g., capturing $tf \cdot idf$ weights of terms). The model m_i corresponds to the classifier obtained by running a supervised learning algorithm on the training set T_i.

Now, instead of transferring the whole training sets T_i, only the models m_i need to be exchanged among the peers. For instance, linear support vector machines (SVMs) [6] construct a hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ that separates the set of positive training examples from a set of negative examples with maximum margin. For a new, previously unseen, document $\boldsymbol{d}$ the SVM merely needs to test whether the document lies on the "positive" side or the "negative" side of the separating hyperplane. The classifiers m_i can represented in a very compressed way: as tuples $(\boldsymbol{w}, \boldsymbol{l}, b)$ of the normal vector $\boldsymbol{w}$ and bias b of the hyperplane and $\boldsymbol{l}$, a vector consisting of the encodings of the terms (e.g. some hashcode) corresponding to the dimensions of $\boldsymbol{w}$.

In the next step, every peer p_j considers the set $M = \{m_1, \ldots, m_k\}$ of k binary classifiers with results $R(m_i, d)$ in $\{+1, -1\}$ for a document $d \in U_j$, namely, +1 if d is accepted for the given topic by m_i and -1 if d is rejected. These results can be easily combined into a meta result:

$$Meta(d) = Meta(R(m_1, d), \ldots, R(m_k, d)) \in \{+1, -1, 0\} \tag{1}$$

A family of such meta methods is the linear classifier combination with thresholding [25]. Given thresholds t_1 and t_2, with $t_1 > t_2$, and weights $w(m_i)$ for the k underlying classifiers we compute $Meta(d)$ as follows:

$$Meta(d) = \begin{cases} +1 \text{ if } \sum_{i=1}^{n} R(m_i, d) \cdot w(m_i) > t_1 \\ -1 \text{ if } \sum_{i=1}^{n} R(m_i, d) \cdot w(m_i) < t_2 \\ 0 \quad \text{otherwise} \end{cases} \tag{2}$$

The important special cases of this meta classifier family include voting [5] ($Meta()$ returns the result of the majority of the classifiers), unanimous decision

(if all classifiers give us the same result), and weighted averaging [28] ($Meta()$ weights the classifiers using some predetermined quality estimator, e.g., a leave-one-out estimator for each v_i).

The restrictive behavior is achieved by the choice of the thresholds: we dismiss the documents where the linear result combination lies between t_1 and t_2. For real world data there is often a tradeoff between the fraction of dismissed documents (the *loss*) and the fraction of correctly classified documents (the *accuracy*).

If a fixed set U of unlabeled documents (that does not change dynamically) is given, we can classify the documents with a user-acceptable loss of L as follows:

1. for all documents in U compute their classification confidence $\sum_{i=1}^{n} R(m_i, d) \cdot w(m_i)$
2. sort the documents into decreasing order according to their confidence values
3. classify the $(1-L)|U|$ documents with the highest confidence values according to their sign and dismiss the rest

In our experiments we assigned equal weights to each classifier, and instead of $R(m_i, d)$, we considered a "confidence" value $conf(m_i, d)$ for the classification of document d by the classifier. For SVM we chose the SVM scores, i.e., the distance of the test points from the hyperplane. A more enhanced method to map SVM outputs to probabilities is described, e.g., in [21].

Note that meta classifiers can be, similar as base classifiers, easily transferred between peers as tuples

$$(m_1, \ldots, m_k, w(m_1), \ldots w(m_k), t_1, t_2). \tag{3}$$

Meta Clustering Algorithms on k Peers Clustering algorithms partition a set of objects, text documents in our case, into groups called *clusters*. In the introduced scenario, each peer p_i contains a document collection U_i of unlabeled data. Every peer wants to cluster its unlabeled data. Analogously to the classification task every peer p_i can execute a clustering algorithm on its own data U_i to build the model m_i; a representation of the resulting clustering models m_i can be propagated to the other peers.

A simple, very popular member of the family of partitioning clustering methods is *k-means* [15]: k initial centers (points) are chosen, every document vector is assigned to the nearest center (according to some distance or similarity metric), and new centers are obtained by computing the means (centroids) of the sets of vectors in each cluster. After several iterations (according to a stopping criterion) one obtains the final centers, and one can cluster the documents accordingly. For the k-means algorithm, the clustering model m_i can be represented as $(\boldsymbol{z_1}, \ldots, \boldsymbol{z_l}, \boldsymbol{l})$, where the $\boldsymbol{z_i}$ are vector representations of the computed centroids, and $\boldsymbol{l}$ contains encodings of the feature dimensions as described above for the supervised case.

After propagating the models, every peer contains a set $M = \{m_1, \ldots, m_k\}$ of different clustering models. Document d is assigned to one of l clusters with labels $\{1, \ldots, l\}$ by each model: $m_i(d) \in \{1, \ldots, l\}$. In the case of k-means this

is the label of the centroid most similar to the document. The goal of meta clustering is now to combine the different clustering results in an appropriate way.

To combine the $m_i(d)$ into a meta result, the first problem is to determine which cluster labels of different methods m_i correspond to each other (note that cluster label 3 of method m_i does not necessarily correspond to the same cluster label 3 of method m_j, but could correspond to say cluster label 1). With perfect clustering methods the solution would become trivial: the documents labeled by m_i as a would be exactly the documents labeled by m_j as b. However, real clustering results exhibit certain fuzziness so that some documents end up in clusters other than their perfectly suitable cluster. Informally, for different clustering methods we would like to associate the clusters which each other which are "most correlated".

Formally, for every method m_i we want to determine a bijective function $map_i : \{1, \ldots, l\} \to \{1, \ldots, l\}$ which assigns all labels $a \in \{1, \ldots, l\}$ assigned by m_i a meta label $map_i(a)$. By these mappings the clustering labels of the different methods are associated with each other and we can define the clustering result for document d using method m_i as:

$$result_i(d) := map_i(m_i(d)) \tag{4}$$

One way to obtain the map_i functions is to take correlation of clusters from different clusterings into account. We want to maximize the correlation between the cluster labels. For sets $A_1..A_x$, we can define their *overlap* as

$$overlap(A_1, .., A_x) := \frac{|A_1 \cap .. \cap A_x|}{|A_1| + .. + |A_x| - |A_1 \cap .. \cap A_x|} \tag{5}$$

Now using

$$A_{ij} := \{d \in U | res_i(d) = j\} \tag{6}$$

we can define the *average overlap* for a document set U and the set of clustering methods M as

$$\frac{1}{l} \sum_{j=1}^{l} \frac{1}{\binom{k}{2}} \sum_{(i,m) \in \{1,\ldots,l\}^2, i<m} overlap(A_{ij}, A_{mj}) \tag{7}$$

We choose the mappings map_i which maximize the average overlap.

After having computed the mapping we are given a set $M = \{m_1, \ldots, m_k\}$ of k binary clustering methods with results $res_i(d)$. For simplicity we consider here the case of $k = 2$ clusters and choose $res_i(d) \in \{+1, -1\}$ for a document d, namely, +1 if d is assigned to cluster 1, and -1 if d is assigned to cluster 2. We can combine these results into a meta result: $Meta(d) = Meta(res_1(d), \ldots, res_k(d))$ in $\{+1, -1, 0\}$ where 0 means abstention. A family of such meta methods is the linear combination with thresholding [24]. Given thresholds t_1 and t_2, with $t_1 > t_2$, and weights $w(m_i)$ for the l underlying clustering methods we compute

$Meta(d)$ as follows:

$$Meta(d) = \begin{cases} +1 \text{ if } \sum_{i=1}^{k} res_i(d) \cdot w(m_i) > t_1 \\ -1 \text{ if } \sum_{i=1}^{k} res_i(d) \cdot w(m_i) < t_2 \\ 0 \quad \text{otherwise} \end{cases} \tag{8}$$

Thus by an intermediate meta mapping step we have a completely analogous situation to the one for the supervised case described in Section 3.2. Confidence values $conf(v_i, d)$ for the clustering of a document d by the base methods v_i can be obtained, say for k-means clustering, by computing the similarity (e.g., using the cosine measure) to the nearest centroid. The restrictive behavior can be obtained in exactly the same way as for the supervised case.

Estimators and Tuning For a restrictive meta classifier, we are interested in its behavior in terms of *accuracy* and *loss* (fraction of unclassified documents). A typical scenario could be a number of users in different peers accepting a loss up to fixed bound, to obtain a higher classification accuracy for the remaining documents. In [25] the tuning of the number k of classifiers for a user-acceptable loss threshold was described. We will not repeat this here and will instead focus on the P2P specific aspects.

The main ingredients of the estimation and tuning process are:

1. estimators for base classifiers (based on cross-validation between the training subsets T_i)
2. estimators for the pairwise correlations between the base classifiers $\{m_1, \ldots, m_k\}$
3. probabilistic estimators for loss and error based on 1. and 2.

For the cross-validation, at least two peers, p_i and p_j, must cooperate: p_i sends a tuple $(m_i, IDs(T_i))$, consisting of its classifier m_i and a list of IDs (not contents!) of its training documents, to p_j. The peer p_j uses the list of submitted IDs to identify duplicates in both collections and performs cross-validation by m_i on $T_j - T_i$. (In the Web context, the IDs of T_i can be easily obtained by computing content-based 'fingerprints' or 'message digests' (e.g. MD5 [23])). The resulting error estimator (a simple numerical value) for m_i can be forwarded from p_j back to p_i or to other peers.

For the computation of pairwise covariance, at least three peers, p_i,p_j and p_m, must cooperate: p_i and p_j send their classifiers and document IDs to p_m and p_m cross-validates in parallel both classifiers on $T_m - T_i - T_j$. By this procedure we get also accuracy estimators.

Finally, the estimators for *covariance* and *accuracy* (numerical values) can be distributed among the peers and estimators for the overall meta classifier can be built. When the estimated quality of the resulting meta classifier does not meet the application-specific peer requirements (e.g. the expected accuracy is still below the specified threshold), the initiating peer may decide to invoke additional nodes for better meta classification. Note that for meta clustering, estimators *cannot* be built in the same easy way, because for the unsupervised case we cannot evaluate base methods by cross-validation.

4 Experiments

Setup To simulate different P2P Web retrieval scenarios (crawling the Web, sending queries to 'Deep Web' portals, analyzing recent newsgroup discussions or publications in electronic journals) we performed multiple series of experiments with real-life data from

1. The academic WebKB dataset [9] containing 8282 HTML Web pages from multiple universities, manually classified into the categories 'student', 'faculty', 'staff', 'department', 'course', 'project', and 'other'.
2. Newsgroups collection at [1]. This collection contains 17847 postings collected from 20 Usenet newsgroups. Particular topics ('rec.autos', 'sci.space', etc.) contain between 600 and 1000 documents.
3. The Reuters articles [17]. This is the most widely used test collection for text categorization research. The collection contains 21578 Reuters newswire stories, subdivided into multiple categories ('earn', 'grain', 'trade', etc.).
4. The Internet Movie Database (IMDB) at [3]. Documents of this collection are articles about movies that include the storyboard, cast overview, and user comments. The collection contains 6853 movie descriptions subdivided into 20 topics according to particular genres ('drama', 'horror', etc.).

We used the Porter stemming algorithm [22] in combination with stopword elimination to transform documents into the vector space model. In all discussed experiments, the standard bag-of-words approach [4] (using term frequencies to build L1-normalized feature vectors) was used for document representation.

Experiments with Supervised Learning Methods (Classification) For each data set we identified all topics with more than 200 documents. These were 20 topics for Newsgroups, 6 for Reuters, 12 for IMDB, 4 for WebKB. Among these topics we randomly chose 100 topic pairs from Newsgroups and all possible combinations for the others, i.e. 66 topic pairs from IMDB, 15 for Reuters, and 6 for WebKB. For each topic pair we randomly chose 200 training documents per class and kept - depending on the available topic sizes in particular collections - a distinct and also randomly chosen set of documents for the validation of the classifiers.

In each experiment, the training data was distributed over 16 peers (data collections in sizes suitable for larger network experiments are hard to get for our scenarios) using equal-sized subsets with approximately 15% overlap (corresponding to peers that contain non-disjoint training data). Among these peers we randomly chose 1,2,4,8, and all 16 peers to simulate various P2P classification scenarios. The configuration with 1 peer corresponds to the 'local' classification that does not involve sharing of classifiers. As discussed in Section 3.2, we also compared the *restrictive* form of meta classification, where we dismissed at each peer exactly the same amount of documents with worst classification confidence using confidence values as discussed in Section 3. Our quality measure is the fraction of correctly classified documents (the *accuracy*) among the documents

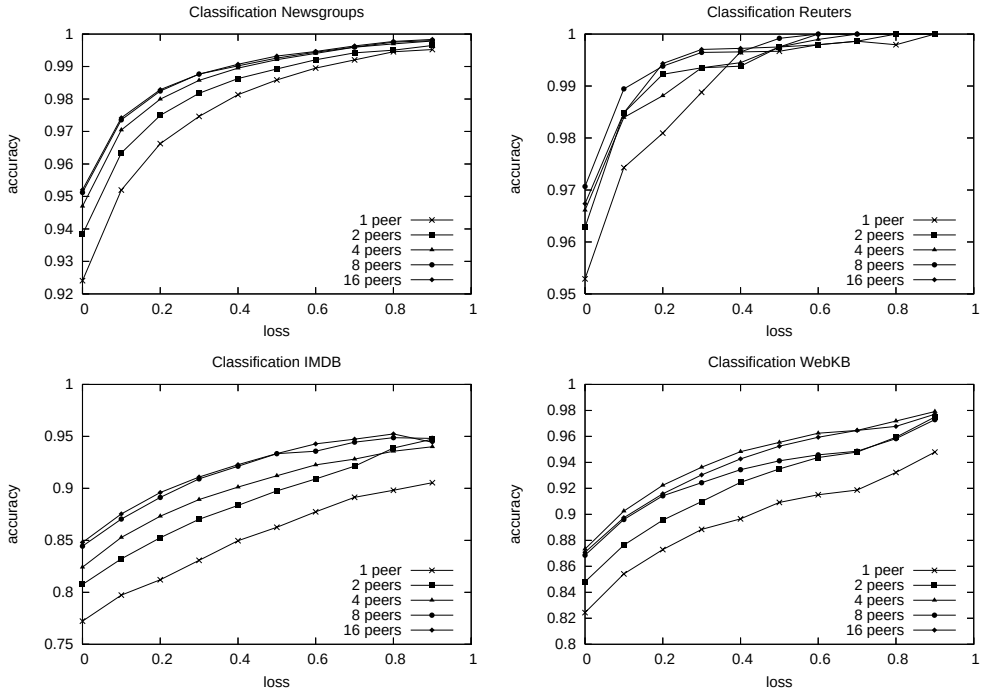

Fig. 1. Results of Restrictive Meta Classification

not dismissed by the restrictive algorithm. The *loss* is the fraction of dismissed documents.

Finally, we computed micro-averaged results along with their 95% confidence intervals for all groups of topic pairs. Figure 1 shows the observed dependencies between the numbers of cooperating peers, the induced loss, and the resulting accuracy for various reference collections. It can be observed that the meta classification and restrictive meta classification by multiple cooperating peers clearly outperforms the single-peer solution for all settings of the user-defined *loss*, including the non-restrictive meta classification with $loss = 0$. The quality of the meta algorithm clearly increases with the number of participating peers. In general, the difference between the one-peer solution and the meta solution is statistically significant for 4 and more participating peers and all values of the induced loss. The only exceptions are the results for Reuters with $loss > 0.7$ (the accuracy of all peer combinations, including one-peer experiment, becomes nearly 1.0) and the WebKB collection (due to the very limited number of possible topic combinations).

Experiments with Unsupervised Learning Methods (Clustering) The same collections and topics were used to evaluate distributed meta clustering. All documents from randomly combined selections of 3 or 5 topics were consid-

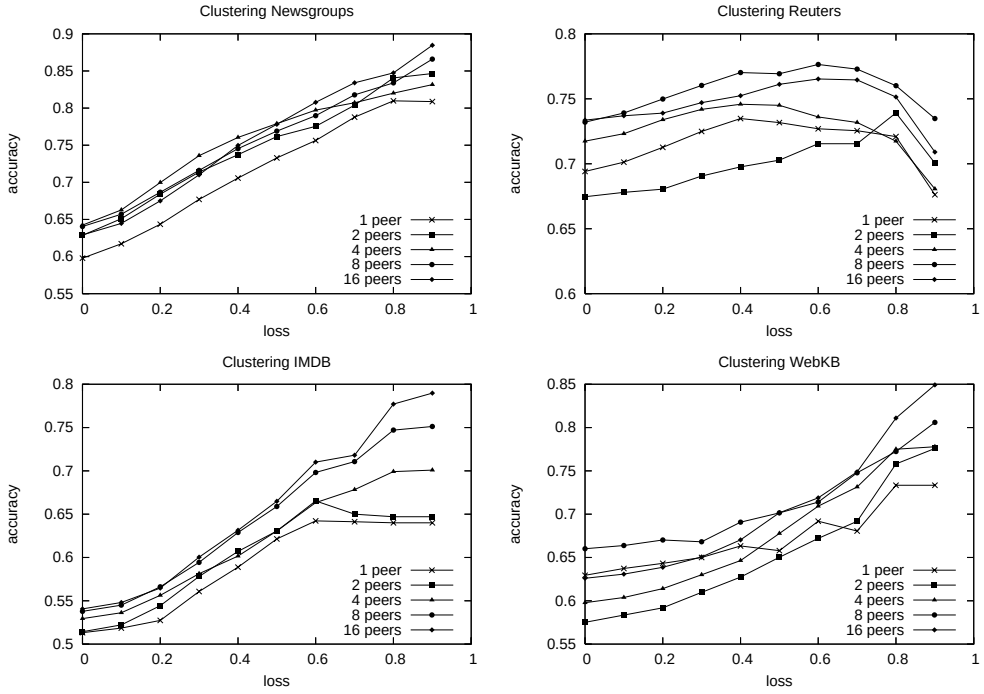

Fig. 2. Results of Restrictive Meta Clustering, k=3 Clusters

ered as unlabeled data and distributed among peers analogously to classification experiments from the previous section, with approximately 15% overlap. The goal of the clustering algorithm was to reproduce the partitioning into topics on each peer with possibly high accuracy. Our quality measure describes the correlation between the actual topics of our datasets and the clusters found by the algorithm. Let k be the number of classes and clusters, N_i the total number of clustered documents in $class_i$, N_{ij} the number of documents contained in $class_i$ and having cluster label j. We define the clustering accuracy as follows:

$$accuracy = \max_{(j_1,\ldots,j_k)\in perm((1,\ldots,k))} \frac{\sum_{i=1}^{k} N_{i,j_i}}{\sum_{i=1}^{k} N_i} \tag{9}$$

The *loss* is the fraction of documents dismissed by the restrictive algorithm.

For all peers, k-means was used as the underlying base method. We compared the one-peer clustering (i.e. clustering that can be executed by one peer on its local dataset without cooperation with others) with meta clustering, exchanging centroids from cooperating peers and correlation-based mapping (Section 3.2) of the final clusters. Analogously to classification experiments, we considered

restrictive meta clustering, dismissing exactly the same number of documents with the worst clustering confidence [24] on each peer.

The results are summarized in Figure 2. The main observations are similar to the ones discussed for the supervised case:

- The quality of the meta clustering results is consistently higher than for isolated one-peer solutions.
- The quality of the meta algorithm tends to increase with the number of participating peers and is in almost all cases statistically significant. For the Reuters collection, the difference between one-peer solution and the meta result is statistically significant for 8 and more participating peers and all values of the induced loss. For the IMDB and Newsgroups collections, the difference between the one-peer solution and the meta result is statistically significant for 4 and more participating peers and all loss values.

In the experiments with the Reuters dataset, the accuracy decreases for high loss values (greater 0.7). Possibly this can be explained by the fact that the Reuters topics - unlike the other considered reference collections - are very different in size (e.g. the topics '*earn*' and '*grain*' contain about 3900 and 280 documents, respectively). The in-depth analysis of such artifacts is subject of our future work.

5 Conclusion

In this paper, we proposed a new methodology to construct distributed machine learning meta models for P2P Web exploration applications. The results of the evaluation clearly show the advantages of cooperation between nodes for building meta decision models. Our method does not require the comprehensive exchange of private data collections between peers and thus provides substantial advantages for aspects of privacy, network bandwidth, storage, and computational expense. Furthermore, in terms of accuracy our restrictive meta methods clearly outperform the models that can be separately built on training sources of isolated peers and - more importantly - also the restrictive variant of such one-peer solutions with the same induced loss.

References

1. The 20 newsgroups data set. *http://www.ai.mit.edu/ jrennie/20Newsgroups/.*
2. dmoz - open directory project. *http://dmoz.org/.*
3. Internet movie database. *http://www.imdb.com.*
4. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.
5. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
6. C. Burges. A tutorial on Support Vector Machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.
7. S. Chakrabarti. *Mining the Web.* Morgan Kaufmann, 2003.

8. P. Chan. An extensible meta-learning approach for scalable and accurate inductive learning. *PhD thesis, Department of Computer Science, Columbia University, New York*, 1996.
9. M. e. a. Craven. Learning to extract symbolic knowledge from the World Wide Web. *15th National Conference on Artificial Intelligence (AAAI)*, 1998.
10. A. e. a. Demers. Epidemic algorithms for replicated database management. *6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, 1987.
11. I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, pages 245–260, 2000.
12. A. Fred and A. K. Jain. Robust data clustering. In *Proc. Conference on Computer Vision and Pattern Recognition, CVPR*, 2003.
13. Y. Freund. An adaptive version of the boost by majority algorithm. *Workshop on Computational Learning Theory*, 1999.
14. Gorunova,K. and Merz,P. Reliable Multicast and its Probabilistic Model for Job Submission in Peer-to-Peer Grids. *WISE, New York, USA*, 2005.
15. J. Hartigan and M. Wong. A k-means clustering algorithm. *Applied Statistics, 28:100-108*, 1979.
16. H. Kargupta, W. Huang, K. Sivakumar, and E. L. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.
17. D. D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318. Defense Advanced Research Projects Agency, Morgan Kaufmann, Feb. 1991.
18. T. Li, S. Zhu, and M. Ogihara. Algorithms for Clustering High Dimensional and Distributed Data. *Intelligent Data Analysis Journal*, 7(4), 2003.
19. N. Littlestone and M. Warmuth. The weighted majority algorithm. *FOCS*, 1989.
20. S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *International Conference on Data Mining (ICDM'03), Melbourne, FL*, 2003.
21. J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers, MIT Press*, 1999.
22. M. Porter. An algorithm for suffix stripping. *Automated Library and Information Systems*, 14(3).
23. R. Rivest. The MD5 message digest algorithm. *RFC 1321*, 1992.
24. S. Siersdorfer and S. Sizov. Restrictive Clustering and Metaclustering for Self-Organizing Document Collections. In *SIGIR*, 2004.
25. S. Siersdorfer, S. Sizov, and G. Weikum. Goal-oriented methods and meta methods for document classification and their parameter tuning. In *CIKM, Washington, USA*, 2004.
26. A. Strehl and J. Gosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research 3, pp. 583-617*, 2002.
27. J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *SDM*, 2004.
28. H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. *SIGKDD*, 2003.
29. D. Wolpert. Stacked generalization. *Neural Networks, Vol. 5, pp. 241-259*, 1992.
30. H. Yu, K. Chang, and J. Han. Heterogeneous learner for Web page classification. *ICDM*, 2002.