Collaborative Filtering with Maximum Entropy

Dmitry Pavlov, Eren Manavoglu, David M. Pennock, C. Lee Giles

***Abstract*— We describe a novel maximum entropy (maxent) approach for generating online recommendations as a user navigates through a collection of documents. We show how to handle high-dimensional sparse data and represent it as a collection of ordered sequences of document requests. Our representation and the maxent approach have several advantages: (1) we can naturally model long-term interactions and dependencies in the data sequences; (2) we can query the model quickly once it is learned, which makes the method applicable to high-volume Web servers; and (3) we obtain empirically high quality recommendations. Although maxent learning is computationally infeasible if implemented in the straightforward way, we explore data clustering and several algorithmic techniques to make learning practical even in high dimensions. We present several methods for combining the predictions of maxent models learned in different clusters. We conduct offline tests using over six months worth of data from ResearchIndex, a popular online repository of over 470,000 computer science documents. We show that our maxent algorithm is arguably one of the most accurate recommenders, as compared to such techniques as correlation, mixture of Markov models, mixture of multinomial models, individual similarity-based recommenders currently available on ResearchIndex, and even various combinations of current ResearchIndex recommenders.**

I. Introduction and Related Work

Recommender systems attempt to automate the process of "word of mouth" recommendations within a community. Typical application environments are dynamic in many respects: users come and go, users' preferences and goals change, items are added and removed, and user navigation itself is a dynamic process. Recommendation domains are also often high dimensional and sparse, with tens or hundreds of thousands of items, among which very few are known to any particular user.

Consider, for instance, the problem of generating recommendations in ResearchIndex (a.k.a., CiteSeer),[1] an online digital library of computer science papers, receiving thousands of user accesses per hour. The site automatically locates computer science papers found on the Web, indexes their full text, allows browsing via the literature citation graph, and isolates the text around citations, among other services [5]. Figure 1 gives a typical screen shot of a document details page in ResearchIndex. Shown are the title and authors of the paper, download options and a number of similarity-based recommenders predicting to the user the documents of possible interest based on the features of the current document.

The archive contains over 470,000 documents including the full text of each document, citation links between documents, and a wealth of user access data. With so many documents,

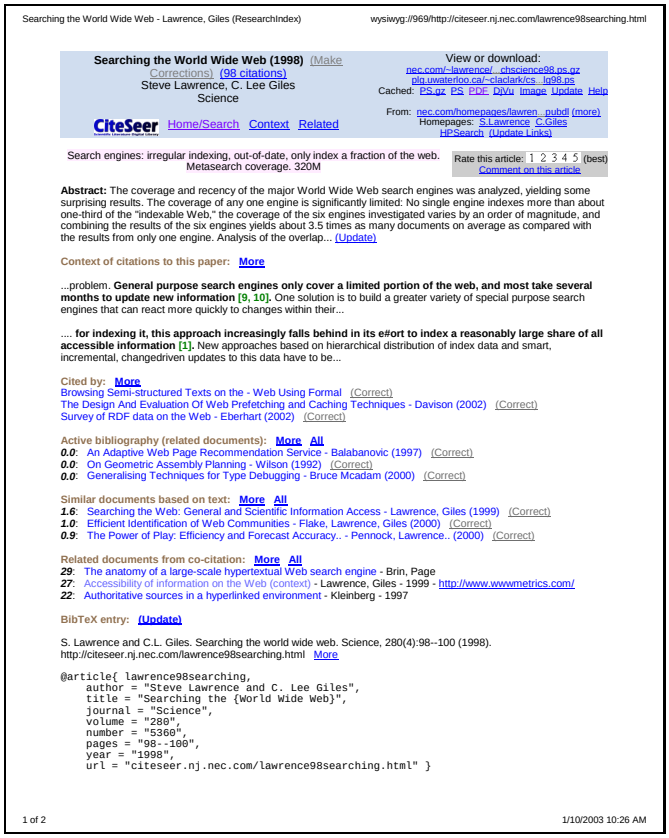

Fig. 1. A typical screen shot of a document details page in ResearchIndex. A user is given a wealth of information about the document as well as the recommendations for possible documents of interest.

and only eight accesses per user on average, the user-document data matrix is exceedingly sparse and thus challenging to model. In this paper, we work with the ResearchIndex data, since it is a rich data source with properties typical to many recommendation application areas [10]; we believe that our methods should work with little adaptation in other domains.

There are two conceptually different ways of making recommendations. *Content filtering* methods recommend solely based on the features of a document D (e.g., showing documents written by the same author(s), or documents textually similar to D). These methods have been shown to work well within ResearchIndex [1]. *Collaborative filtering* methods [9] work by assessing the similarities between users based on their overlap in document requests, then recommending to a given user documents that like-minded users accessed previously. Common measures of similarity between users include correlation, mean squared error, vector similarity, and Bayesian similarity measures. Other approaches include the application of statistical machine learning techniques, such as Bayesian networks dependency networks singular value decomposition

D. Pavlov is with Yahoo! Inc. This work was done at NEC Laboratories America.
D. Pennock is a Senior Research Scientist at Overture Services.
E. Manavoglu and C.L. Giles are with The Pennsylvania State University.
[1] `http://www.researchindex.com`

and latent class models.

Currently, ResearchIndex employs several *similarity-based* recommenders most of which are content-based and user-independent. On the other hand, most collaborative filtering algorithms, are user-specific, but are context and order independent: that is, the rank of recommendations does not depend on the context of the user's current navigation or on recency effects. Our objective was to design a superior (or at least complementary) model-based recommendation algorithm for ResearchIndex that (1) is tuned for a particular user at hand, and (2) takes recent documents into greater account, so as not to lead the user too far astray from his or her current search goal.

In this paper, we compare several probabilistic model-based collaborative recommenders to the standard correlation method, and to the similarity-based recommenders currently available in ResearchIndex. A probabilistic formulation of the recommending task can be posed as $P(D^{next}|H(U))$, where D^{next} is the next document requested by the user U and $H(U)$ is the history of document requests for the user U in the present session. This formulation allows for a number of choices to model P. We found one of the best to be a maximum entropy (maxent) approach that explicitly models time: each user is associated with a set of sessions, and each session is modeled as an ordered time sequence of document accesses. This approach seems well aligned with other recent efforts to model the recommendation problem as a sequential rather than a static problem. Our maxent model effectively estimates the probability of the next visited document given the most recently visited document (*"bigrams"*) and past indicative documents (*"triggers"*). To our knowledge, this is the first application of maxent for collaborative filtering, and one of the few published formulations that makes accurate recommendations in the context of a dynamic user session [11], [1]. We also implemented and evaluated two other probabilistic models: a mixture of Markov models and a mixture of multinomial models. In Section II we describe the maxent, Markov, and multinomial models in detail.

One of the simplest, yet accurate and often-used recommendation algorithms is correlation, which we employ as a baseline. We also compare against the similarity-based predictors already available in ResearchIndex, including merged combinations of these predictors that take users' navigation histories into account.

At present, ResearchIndex contains over 470,000 documents, with about 15% actively accessed, and slightly more than half requested more than once in our data set. Thus, we face a problem of fitting a distribution on a variable that takes on an order of 100,000 values. We are able to fit the Markov and multinomial mixture models to the available data directly, despite this huge scale. However, our maxent model cannot be feasibly learned over the entire data set, due to the high computational cost. We develop a greedy hierarchical approach to *clustering* that provides a feasible implementation. Clustering allows us to overcome sparsity and high dimensionality of the data, by reducing the original problem to a set of smaller subproblems corresponding to clusters. We then describe how to recombine the cluster-specific models in order to make recommendations across the entire data set and allow direct comparison with the non-clustering approaches (i.e., the mixture models, correlation, and the similarity-based recommenders). This contrasts with our previous work [6] which provided within-cluster recommendations only.

All the algorithms trade off accuracy, speed, and memory use. Our results demonstrate that the maxent approach offers one of the best performance profiles. Maxent is more accurate than the other model-based approaches, about as accurate as correlation, and orders of magnitude faster than correlation. Maxent even compares favorably against the current similarity-based recommenders in ResearchIndex. This is significant since the test data should be biased in favor of the similarity-based recommenders, as those recommenders were installed and running when the data was generated.

The overall contributions of this paper can be summarized as follows:

- We represent the data as a collection of ordered sequences of document requests, which allows us to model long-term interactions and dependencies in the data sequences.
- We present a novel maximum entropy approach for generating online recommendations and suggest a document clustering approach for speeding up training of the model.
- We compare the maximum entropy approach to (1) a mixture of multinomial models, (2) a mixture of Markov models, (3) correlation—one of the best and most widely used collaborative filtering recommenders, (4) similarity-based recommenders currently available in ResearchIndex, and (5) combinations of the similarity-based recommenders. In offline experiments on over 6 months worth of data from ResearchIndex, we show that maxent allows for fast model querying, is relatively compact, and on the whole performs as well or better than its competitors in terms of accuracy.

The rest of the paper is organized as follows. In Section II, we give descriptions and of the probabilistic models, motivate our choice of using maximum entropy, and describe the features used for learning. Section III presents our greedy algorithm for clustering documents and describes how the clustering helps to decompose the original prediction task in order to render maxent learning computationally feasible. Experimental results and comparisons are given in Section IV. In Section V, we draw conclusions and ponder future work.

II. The Probabilistic Models

We assume that we are given a data set consisting of ordered sequences in some fixed alphabet. We refer to individual items in the alphabet as documents or, simply, items. For each document we define its history H as a so-far observed (ordered) subsequence of the current sequence. D^{prev} is the last observed document in H. Our task is to predict the next document D^{next} given the history H. We intend to accomplish this task by learning a probabilistic model $P(D^{next}|H)$ from the available training data.[2] Below we explore several alternatives for P.

[2]To be absolutely precise, we should write $P(D^{next}|H, Data)$ but for simplicity we omit $Data$ in conditioning.

A. Mixture of Multinomials

The *mixture of multinomials* essentially prescribes to disregard the sequential information and treat the available document accesses in the history H as a "bag of items":

$$\begin{aligned} P(D^{next}|H) &= \sum_{k=1}^{N_c} \alpha_k P(D^{next}|H,k); \\ P(D^{next}|H,k) &\propto \prod_t \theta_{kt}^{n_t}, \end{aligned} \quad (1)$$

where N_c is the number of mixture components, θ_{kt} is the probability that the document number t is accessed and n_t is the number of times the document number t was accessed in H. Fitting the model can be done using the Expectation-Maximization (EM) algorithm as described in [2]. The number of parameters of the model is linear in the number of documents and the number of mixture components.

B. Mixture of Markov Models

In the first order Markov model, the main assumption is that the current document depends on the history H only through the last observed document in H, i.e. D^{prev}. The equations defining the *mixture of Markov models* are as follows:

$$\begin{aligned} P(D^{next}|H) &= \sum_{k=1}^{N_c} \alpha_k P(D^{next}|H,k); \\ P(D^{next}|H,k) &\propto \theta_{0,k} \prod_{h=1}^{|H|} \theta_{(h\to h+1),k}, \end{aligned} \quad (2)$$

where the first equation is just a standard equation for the mixture, while the second equation uncovers how each component is modeled; $\theta_{0,k}$ is the probability of observing H_0 as a first document in the history, and $\theta_{(h\to h+1),k}$ is the probability of observing a transition from document number h to document number $h+1$ in the history. For $h = |H|$, document with index $h+1$ is D^{next}. This model can also be learned by using the EM algorithm. The number of parameters is quadratic in the number of documents and linear in the number of components. When $N_c = 1$, the model reduces to a regular Markov model (single component mixture). Note that the regular Markov model only depends on the so-called *bigrams* or first order Markov terms, i.e. the frequencies of pairs of consecutive documents.

C. Trigger Maximum Entropy Model

The distribution $P(D^{next}|H)$ can also be modeled as a maximum entropy distribution. Our main motivation for using maximum entropy estimation is to combine the attractive features of the Markov and multinomial models. In particular, we believe that the document D^{prev} requested immediately prior to D^{next} has the most influence on what D^{next} is. For this reason it is essential to model sequence information at least for the pair $D^{prev}D^{next}$, as the Markov model does. On the other hand, we also believe that the documents in H beyond D^{prev} also have influence on D^{next}. We want something more sophisticated than the multinomial model which is essentially a zeroth-order model. A higher (than the first) order Markov model would probably suffice but we cannot afford to build it because of the curse of dimensionality. Thus, we have to restrict ourselves to models that can be reliably estimated from the low-order statistics but still look at the whole H, and this naturally lead us to consider the maximum entropy model.

We select two flavors of low-order statistics or features. *Bigrams*, or first order Markov terms, are one type and *triggers* are another. In order to introduce long term dependence of D^{next} on the documents that occurred in the history of the session, we define a trigger as a pair of documents (a, b) such that $P(D^{next} = b|a \in H)$ is substantially different from $P(D^{next} = b)$. To measure the quality of triggers and in order to rank them, we compute mutual information between events $E_1 = \{D^{next} = b\}$ and $E_2 = \{a \in H\}$. We then discard 10% of low scoring triggers but retain all bigrams. Note that the quantity and quality of selected triggers depends on the length of H. Since the average transaction only has about 8 document requests (see Table II), we choose 10 to be the maximum length of the history for the purposes of finding informative triggers.

The set of features—bigrams and triggers in our case—together with maximum entropy as an objective function, can be shown (as done in [4]) to lead to the following form of the conditional maximum entropy model:

$$P(D^{next}|H) = \frac{1}{Z_\lambda(H)} exp[\sum_{s=1}^{S} \lambda_s F_s(D^{next}, H)], \quad (3)$$

where F_s are the features and $Z_\lambda(H)$ is a normalization constant ensuring that the distribution sums to 1:

$$Z_\lambda(H) = \sum_D exp[\sum_{s=1}^{S} \lambda_s F_s(D, H)]. \quad (4)$$

The set of parameters $\{\lambda\}$ needs to be found from the following set of equations that restrict the distribution $P(D^{next}|H)$ to have the same expected value for each feature as observed in the training data:

$$\sum_H \sum_D P(D|H) F_s(D, H) = \sum_H F_s(D(H), H), \quad s = 1, \ldots, S, \quad (5)$$

where $D(H)$ is the document following H in the training data. The LHS in Equation 5 represents the expectation (up to a normalization factor) of the feature $F_s(D, H)$ with respect to the distribution $P(D|H)$ and the RHS is the frequency (up to the same normalization factor) of this feature in the training data. There exist efficient algorithms for finding the parameters $\{\lambda\}$ (e.g. generalized, improved and sequential conditional iterative scaling algorithms) that are known to converge if the constraints imposed on P are consistent.

Under fairly general assumptions, the maximum entropy model can also be shown to be a maximum likelihood model. Employing a Gaussian prior with a zero mean on parameters

λ yields a maximum aposteriori solution that has been shown to be more accurate than the related maximum likelihood solution and other smoothing techniques for maximum entropy models. We use Gaussian smoothing in our experiments with a maximum entropy model.

III. Dimensionality Reduction Via Clustering

As we mentioned in Section I, models such as the mixture of Markov or multinomial models, can be fit to high-dimensional data directly using the EM algorithm. However, depending on the number of selected features, learning the maximum entropy model directly on our raw training data could take months of computation. Goodman [3] arrives at a similar conclusion. As we have shown in [6], the problem can be solved by employing clustering. Here we discuss a simple and fast greedy clustering algorithm. We further show that a nice feature of our algorithm is that it finds topically related clusters. Finally, we discuss how to combine maximum entropy models learned in different clusters in order to provide global recommendations.

A. Clustering Based On User Navigation

The goal of clustering based on user navigation patterns is to maximize the probability that once a document D from the cluster c is requested, the user session will not leave c (i.e., will consist only of documents in c). Our intuition is that within a relatively small time frame most of the users are interested in a single topic, and if we succeed in finding topically related clusters, we should achieve the goal. Here, instead of looking at the contents of individual document or user queries, we take a purely collaborative approach, and cluster the documents solely based on user navigation sequences.

In order to come up with such a clustering, we scanned the processed training data, and, for each pair of consecutively requested documents, we computed its count. The gathered statistics are nothing else but the bigram counts. Even though the bigram counts are fairly sparse, i.e. for every document there is not much incoming or outgoing traffic, clustering such data is still challenging.

We tried the PageGather [8] algorithm for clustering the bigrams. The idea behind PageGather is to create the attribute interaction graph based on bigrams and identify the cliques this graph using one of the well-known techniques, such as join-tree clustering. The algorithm returned us one large clique and a lot of smaller ones, whereas we would prefer a set of clusters with more balanced sizes. Thus, we decided to employ a very straightforward hierarchical greedy algorithm, the pseudocode for which looks as follows:

Input: Bigrams counts $B[i,j]$; Number of Clusters C;
Output: Set S of C Clusters.
Algorithm:

0. $n_C = 0$;
1. set $n = argmax_{i,j} B[i,j]$ // most frequent bigram
2. for all docs i, j such that $B[i,j] == n$ do
   // all docs with n transitions
3. if (i and j are unassigned and $n_C < C$)
4. $S[n_C] \leftarrow \{i, j\}$;
5. n_C++; // new cluster for i and j
6. else if (i is unassigned)
7. $S[n_C] \leftarrow S[n_C] \bigcup \{i\}$;
8. else if (j is unassigned)
9. $S[n_C] \leftarrow S[n_C] \bigcup \{j\}$;
10. end if
11. $B[i,j] = -1$;
12. end for
13. if $(n \geq 2)$ goto 1
14. Return S

The algorithm starts with empty clusters and then cycles through all documents picking the pairs of documents that have the current highest joint visitation frequency as prompted by a bigram frequency (lines 1 and 2). If both documents in the selected pair are unassigned, a new cluster is allocated for them (lines 3 through 5). If one of the documents in the selected pair has been assigned to one of the previous clusters, the second document is assigned to the same cluster (lines 6 through 10). The algorithm repeats for a lower frequency n, as long as $n \geq 2$.

After the clustering, we can assume that if the user requests a document from the i-th cluster $S[i]$, he is considerably more likely to prefer a next document from $S[i]$ rather than from $S[j]$, $j \neq i$, i.e. $P = P(D^{next} \in S[i] | D^{prev} \in S[i], Data) \gg 1 - P$. This assumption is reasonable because by construction clusters represent densely connected (in terms of traffic) components, and the traffic across the clusters is small compared to the traffic within each cluster.

Our previous results described in [6] were obtained with this clustering. We broke individual user sessions down into subsessions, where each subsession consisted of documents belonging to the same cluster. The problem was thus reduced to a series of prediction problems for each cluster.

We also studied the clusters by trying to find out if the documents within a cluster are topically related. We ran code previously developed at NEC Labs that uses information gain to find the top features that distinguish each cluster from the rest. Table I shows the top features for some of the created clusters. The top features are quite consistent descriptors, suggesting that in one session a ResearchIndex user is typically interested in searching among topically-related documents, at least during one session (as defined above). Thus, a straightforward greedy clustering algorithm is successful in finding such attractive topically-related clusters.

B. Combining Predictions from Different Clusters

In order to be able to make predictions on the raw test data we need an ability to combine predictions of the learned maximum entropy models for different clusters into a predictor that would work on the unclustered data.

We used the following idea borrowed from Goodman [3]. Consider an arbitrary clustering C of a set of documents, such that a document belongs to one, and only one cluster. Then the probability $P(D^{next}|H)$ can be represented as

TABLE I

TOP FEATURES FOR SOME OF THE CLUSTERS.

Cluster 1	$agent$, $agents$, $behavior$, $good$, $autonomous$, the_agent, an_agent, ...
Cluster 2	$training$, $clustering$, $distance$, $classification$, $kernel$, svm, $support$, ...,
Cluster 3	web, $documents$, $query$, the_web, $queries$, $pages$, web, $engines$, ...
Cluster 4	$packet$, $fast$, $routing$, $address$, $the\_network$, ip, $packets$, $speed$, ...
Cluster 5	$transform$, $channel$, $coding$, $rate\_compression$, $images$, $coefficients$, ...
Cluster 6	$detection$, $agents$, $security$, $intrusion\_detection$, $host$, ...
Cluster 7	$traffic$, $rate$, $packet$, $long$, $wide\_scheduling$, $service$, qos, ...
Cluster 8	$mobile$, $wireless$, $protocol$, $service$, $services$, $location$, ...

$$P(D^{next}|H) = \sum_C P(D^{next}|C,H)P(C|H) = P(D^{next}|C(D^{next}),H)P(C(D^{next})|H). \quad (6)$$

The first equality is just a rule of full probability, while the second equality holds since each document has a unique cluster assignment.

Our scheme of Section III-A for greedy clustering is directly applicable to this setting. We use a maximum entropy model to represent $P(D^{next}|C(D^{next}),H)$ in each cluster. The only remaining question is how to learn the weights $P(C(D^{next})|H)$. Goodman [3] suggests learning a separate maximum entropy model to represent the weights. However, we found that the following ad hoc approximation scheme works reasonably well in practice.

Recall that every document has an associated cluster. If we scan the history H, we can accumulate the frequency with which each cluster (represented by a document in the H) has occurred in H. For instance, if all documents in H were from the same cluster c, then only c would get a non-zero weight of 1, and we would only have to make predictions within c. If there were half the documents in H from c_1 and half from c_2, then we would make predictions in each of these clusters and weigh them equally. Our experimental results below use this technique for combining predictions.

IV. EXPERIMENTAL RESULTS AND COMPARISONS

A. Preprocessing the ResearchIndex data

We obtained a log file that recorded over 6 month worth of ResearchIndex data that can roughly be viewed as a series of requests $< \text{time}, \text{user}, \text{document} >$. We chronologically partitioned the log into roughly 5 million training requests (covering 159 days) and 1.7 million test requests (covering 50 days). We preprocessed the training and test sets as follows. Each document indexed in ResearchIndex is assigned a unique document id (DID). Whenever a user accesses the site with a cookie-enabled browser, (s)he is identified as a new or returning user and all activity is recorded on the server side with a unique user id (UID) and a time stamp (TID).

In the first processing step, we aggregated the requests by the UID and broke them into sessions. For a fixed UID, a session is defined as a sequence of document requests, with no two consecutive requests more than T seconds apart. In our experiments we chose $T = 120$, so that if a user was inactive for more than 120 seconds, his next request was considered to mark a start of a new session.

The next processing step included heuristics, such as identifying and discarding the sessions belonging to robots (they obviously contaminate the browsing patterns of human users), collapsing all same consecutive document accesses into a single instance of this document (our objective was to predict what interests the user beyond the currently requested document), getting rid of all documents that occurred just once in the log and finally discarding sessions containing only one document.

After the preprocessing step, our data was represented as a collection of ordered sequences of document requests, possibly with multiple sequences per user. All models and recommenders took this data as an input. The properties of the data are described in Table II.

B. Parameters of Competing Models

We performed the clustering of the training data set using the greedy algorithm described in Section III-A. A maximum entropy model was learned separately for each cluster, with all bigrams and 90% of the top triggers retained for evaluation as discussed in Section II-C. Global predictions of the maximum entropy model on the test data were obtained by combining individual cluster maximum entropy models as described in Section III-B.

We compared our maximum entropy approach with the following models: a single-component Markov model, a mixture of Markov models (30 components), a mixture of multinomials (60 components), the correlation method, the individual similarity-based predictors, and the ResearchIndex merged similarity-based predictor described below. For computation reasons, we did not optimize the adjustable parameters of the models (such as the number of components for the mixture or the variance of the prior for maximum entropy models) or the number of clusters (1000).

Following is the list of similarity recommenders available in ResearchIndex. More detailed descriptions can be found in [5]:

1) **Active Bibliography** finds the documents that make similar citations and are also close in the sense of word frequencies;
2) **Sentence Similarity** displays documents with significant number of identical sentences;
3) **Text Similarity** recommends documents that are similar to the current one in terms of word occurrences;

TABLE II
STATISTICAL PROPERTIES OF THE TRAINING AND THE TEST DATA USED IN THE EXPERIMENTS.

	Training Set	Test Set
Total Number of Users	567,472	212,339
Total Number of Documents	263,156	233,361
Average Requests per User	8.6785	8.0977
Standard Deviation of Requests per User	31.9832	27.6857
Minimum Requests per User	2	2
Maximum Requests per User	4486	3421

4) **Users Who Viewed** displays documents requested by the users who also visited the current document;
5) **On Same Site** shows other documents listed on the same Web site as a current document;
6) **Cited By** lists the documents that cite the current one;
7) **Co-citation** recommends documents that are often cited together with the current document;
8) **Cites** displays the documents cited by the current document.

The lists of recommendations for individual similarity-based predictors were populated by taking the information currently stored in the system, which is in most cases 3-5 top recommendations per document per recommender. We call **ResearchIndex Merge** the recommender that is obtained by pulling all similarity-based recommenders together—essentially, this corresponds to the currently available recommending system in ResearchIndex.

C. Evaluation Metrics

All models that were trained on the training data and evaluated on the test data. Evaluation was performed by scanning the sequences of document requests document by document and for each document in the sequence predicting the identity of the next document.

We used the *average height of predictions* on the test data as the main evaluation criteria. The height of a prediction corresponds to the rank of the requested document within the recommendation list of a given model or recommender. The formal definition is as follows. Assuming that the probability estimates $P(D^{next}|H)$ are available from a model P for a fixed history H and all possible values of D^{next}, we first sort them in the descending order of P and then find the distance in terms of the number of documents to the actually requested D (which we know from the test data) from the top of this sorted list. The height tells us how deep into the list the user must go in order to see the document that actually interests him. The height of a perfect prediction is 0, i.e. the requested document is the first one on the list. For ResearchIndex recommenders we used the order of predictions as stored in the system. If for a given document a specific recommender did not have any predictions, then for all occurrences of this document in the test data the height was set to infinity.

To make comparisons among different recommenders easier, we binned the heights of predictors. We used bins with intervals $[0, K]$ for $K = 1, \ldots, 5, 10$. For all recommenders and all bins we computed two statistics:

1) **Hits** is the total number of recommendations falling within the bin normalized by the total number of recommendations made;
2) **Height** is the average height of predictions within the bin.

The best performing model would place most of the predictions inside the bins with smaller values of K and within those bins the average heights would be as low as possible. Note the hits statistics is primary for comparison, while the height statistic only plays a role for ranking recommenders with similar hits ratios.

The primary goal of our experiments was to establish which of the competing recommending strategies provides the best accuracy in terms of hits and height statistics. We also compared the recommenders with respect to the average online time and the total memory taken to generate a prediction. Time is an essential characteristic: the current implementation of ResearchIndex ignores any recommendations that take more than 0.01 seconds to make. The memory is less important since in principle an attractively accurate and reasonably fast recommending system can reside on a separate computer, with all its memory resources available for recommending.

Our expectation is that our models will not be competitive with similarity-based recommenders with respect to memory, since the latter only use small lists of predictions per document, while the former model attribute interactions and can be quite expensive. For a simple example consider the case of modeling $250,000$ attributes which is roughly how much we have. All bigrams take quadratic memory in the number of attributes, which would result in roughly $\frac{1}{2}250,000^2 * 4Bytes \approx 116Gigabytes$. Luckily, as we will see below many bigrams have 0 counts and do not need to be stored. On the other hand, if one keeps only 7 numbers per document (as would be the case for most similarity-based recommenders), this would only result in $250,000 * 7 * 4Bytes \approx 6.6Megabytes$.

The time taken to generate recommendations for all D^{next} using a probabilistic model $P(D^{next}|H)$ could be quite high. Indeed, this operation amounts to sequentially generating $P(D^{next}|H)$ for all D^{next} and sorting the resulting list.[3] An important role in reducing the time requirements for probabilistic models thus belongs to selecting a limited set of candidates for D^{next} depending on H. Since selecting the candidates set can be non-obvious for certain models, and straightforward for the other, we review our choices here:

[3]A notable exception is dependency networks, capable of generating predictions for all D^{next} simultaneously.

- Markov mixtures: the candidate set was generated naturally by assessing the bigrams $D^{prev}D^{next}$ for each document D^{prev}.
- Multinomial mixtures: in view of the absence of other obvious choice, the candidate set was chosen to be a set of all similarity-based recommendations available for D^{prev}.
- Correlation: no candidate set (and tremendous time requirements as a result).
- Maximum entropy: the candidate set is naturally defined by the set of documents belonging to the clusters extracted from H as described in Section III-B.

Thus, we do expect our models to be fast enough to fit into the recommending time limit requirements mentioned above and accurate enough to compete with existing methods.

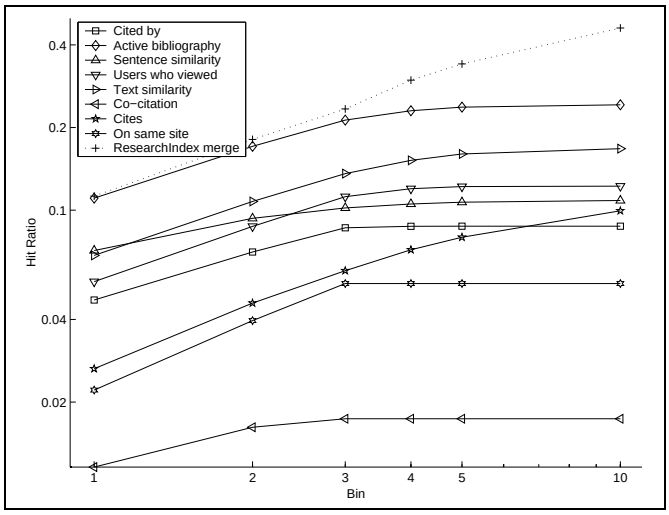

Fig. 2. Ratio of hits to the total number of document requests for the bins $1, \ldots, 5, 10$ and various similarity-based recommenders currently used in ResearchIndex.

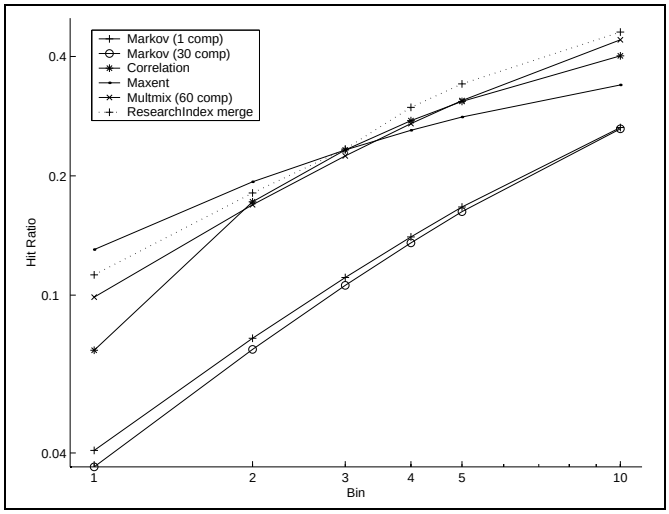

Fig. 3. Ratio of hits to the total number of document requests for the bins $1, \ldots, 5, 10$ and different probabilistic models. Also shown is correlation and the ResearchIndex Merge recommender.

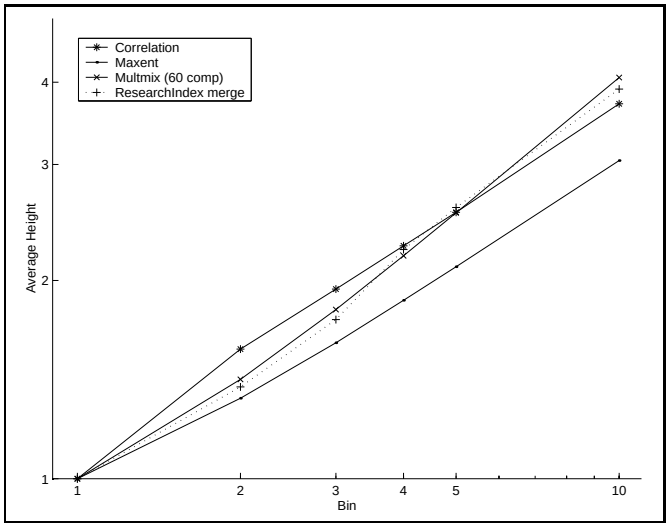

Fig. 4. Average heights of predictions for the bins $1, \ldots, 5, 10$ and four best recommenders with respect to hits ratio (see Figure 3).

D. Experimental Results

In Figure 2 we present the graph showing the number of hits for each bin for all recommending methods currently used in ResearchIndex. As follows from the plot and in line with our expectations, the ResearchIndex Merge recommender is the best similarity-based recommender. In Figure 3 we show the plot similar to that of the Figure 2 but for all probabilistic collaborative filtering models, correlation and the ResearchIndex Merge recommender. Maximum entropy outperforms all recommenders, including ResearchIndex Merge for $K < 3$. It is also one of the top predictors for all other values of K. The mixture of multinomial models appears to be one of the top models for $K \geq 5$. Notice that this is not surprising, since the list of candidates for the mixture of multinomials was created from the predictions of the individual similarity-based recommenders. Performance of a single component Markov model improves for larger bins. Interestingly, the mixture of Markov models fails to improve over the performance of a single component model, probably due to overfitting. The Co-citation is the worst and the Active Bibliography is the best stand-alone similarity-based recommender, while Sentence Similarity and Cited By perform better than Co-citation, however, their numbers of hits are still less than half of the maximum entropy. Correlation improves its performance for larger values of K, scoring one of the best for $K = 10$.

In Figure 4 we report the average height of predictions for all recommenders and bins. Note that we treat average height within bins as a secondary statistic only used for comparison of recommenders with similar hit ratios. Unlike hit ratios, the lower the height the better. Of the best 4 competing techniques with respect to the hit ratios, namely the maximum entropy, the multinomial mixture, the correlation and the ResearchIndex merge, the maximum entropy is the best with respect to the average height, while the other three are in fairly close tie.

Overall, we conclude that model-based approach provides an attractive alternative in terms of the quality of predictions to currently used recommenders.

TABLE III
AVERAGE TIME PER 1000 PREDICTIONS AND TOTAL MEMORY USED BY VARIOUS MODELS.

	Time, s	Memory, MBytes
Cited By	0.0062	3.8
Active Bibliography	0.0082	6.0
Sentence Similarity	0.0056	3.3
Users Who Viewed	0.0078	5.9
Text Similarity	0.0080	6.3
Co-citation	0.0049	2.4
Cites	0.0072	7.2
On Same Site	0.0068	6.0
RsearchIndex Merge	0.0144	24.0
Mix. of Mult., 60 comp.	2.5285	60.75
Mix. of Markov, 1 comp.	0.0703	9.73
Mix. of Markov, 30 comp.	0.5204	292.0
Maxent	7.6281	458.1
Correlation	$> 1hour$	80.9

In Table III we present comparison of various models with respect to the average time taken and memory required to make a prediction. The table clearly illustrates that all of the models we chose for experiments are on average faster than $\frac{1}{100}$ of a second, which is currently the maximum time allowed by the system to generate a recommendation. In line with our expectations, the memory requirements of our models are much higher than for the individual recommenders. However, as we argued above this should not constitute a practical problem for a recommender with attractive and reasonably fast performance, running on a stand alone machine. For the maximum entropy approach, there is also a possibility for further reducing the memory consumption by either limiting the maximum size of the cluster or to increasing the number of clusters. In the former case, if the maximum size of the cluster is fixed at 1000, the memory usage drops from 458 MegaBytes to 276 MegaBytes with a negligible decrease in performance. Increasing the number of clusters will probably also result in less accurate models and will require a better modeling of $P(cluster|H)$.

V. CONCLUSIONS AND FUTURE WORK

We described a model-based collaborative filtering approach to generating document recommendations in ResearchIndex. We represented our data as a collection of sequences of document requests ordered in time. Most of our models, such as mixtures of Markov and multinomial models, are easy to fit to sparse, high-dimensional data without any additional preprocessing using the EM algorithm. We demonstrated that by employing a clustering of the documents based on the user navigation patterns we can also apply the maximum entropy modeling to the ResearchIndex data. The maximum entropy approach provides us with a unique advantage of being able to model long-term interactions and dependencies in the data sequences. In our definition, the maximum entropy model combines zero and first order Markov terms as well as the triggers with high information content. We empirically demonstrated that maximum entropy delivers performance that compares well with all of the currently existing individual similarity-based recommenders in ResearchIndex. It also outperforms the combined recommender in a number of experimental conditions, as well as correlation, one of the more popular collaborative filtering techniques. Another model that performed well in experiments was the mixture of multinomials. All models generated fast enough recommendations to be used in real-time on high volume web servers.

Note that in any application domain the question "which recommender is the best" cannot be answered solely by offline experiments on the historical logs, as was conducted in this work. ResearchIndex for instance enforces a certain interface, showing only the top 3 documents according to each of the available similarity-based recommenders, and this clearly affects user interaction with the system, consequently biasing the logs. In future work, we plan to perform "live" testing of the clustering approach and various models in ResearchIndex, using the **RE**commender **F**ramework and **E**valuator for **RE**s**E**archIndex, or REFEREE [1]. We also plan to enhance our maximum entropy approach with a better model for $P(cluster|H)$, which could also be modeled using maximum entropy, as advocated by Goodman [3]. Our recent work [7] suggests that for difficult prediction problems improvement beyond the plain maximum entropy models can be sought by employing the mixtures of maximum entropy models. Our expectation is that such combining could yield better accuracy and faster recommendations; however, we still need to explore the feasibility of fitting this model to the data. We also plan to evaluate different clustering methods for documents and try to combine prediction results for different clusterings. An idea we started exploring is the use of content-based recommendation information for clustering, and clustering based on user queries. Yet another interesting area of research is combining the collaborative filtering and content recommendations.

When mining the log files of ResearchIndex, we observed different patterns of browsing by different groups of users. There were users who never followed the recommendations, and there were users who used a totally different order of

similarity-based predictors. We believe that further personalization of recommendations can boost the system's performance.

VI. Acknowledgements

We would like to thank Steve Lawrence for making available the ResearchIndex log data, Eric Glover for running his naming code on our clusters, and Kostas Tsioutsiouliklis and Darya Chudova for many useful discussions.

References

[1] D. Cosley, S. Lawrence, and D. Pennock. An open framework for practical testing of recommender systems using ResearchIndex. In *International Conference on Very Large Databases (VLDB'02)*, 2002.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B-39:1–38, 1977.

[3] J. Goodman. Classes for fast maximum entropy training. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[4] F. Jelinek. *Statistical Methods for Speech Recognition*. Cambridge. MA:MIT Press, 1998.

[5] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[6] D. Pavlov and D. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Proceedings of Neural Information Processing Systems*, 2002.

[7] D. Pavlov, A. Popescul, D. Pennock, and L. Ungar. Mixtures of conditional maximum entropy models. Technical Report NECI TR, NEC Research Institute, 2002.

[8] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 727—732, 1998.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommender algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[11] G. Shani, R. Brafman, and D. Heckerman. An MDP-based recommender system. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 453—460, 2002.