In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Part 2*, pp. 111-127, Bled, Slovenia, September 2009.

Learning to Disambiguate Search Queries from Short Sessions

Lilyana Mihalkova and Raymond Mooney

The University of Texas at Austin
Department of Computer Sciences
1 University Station C0500
Austin, Texas 78712-0233
{lilyanam,mooney}@cs.utexas.edu

Abstract. Web searches tend to be short and ambiguous. It is therefore not surprising that Web query disambiguation is an actively researched topic. To provide a personalized experience for a user, most existing work relies on search engine log data in which the search activities of *that particular user*, as well as other users, are recorded over *long* periods of time. Such approaches may raise privacy concerns and may be difficult to implement for pragmatic reasons. We present an approach to Web query disambiguation that bases its predictions only on a short glimpse of user search activity, captured in a brief session of 4–6 previous searches on average. Our method exploits the relations of the current search session to previous similarly *short* sessions of other users in order to predict the user's intentions and is based on Markov logic, a statistical relational learning model that has been successfully applied to challenging language problems in the past. We present empirical results that demonstrate the effectiveness of our proposed approach on data collected from a commercial general-purpose search engine.

1 Introduction

Personalizing a user's Web search experience has become a vibrant area of research in recent years. One of the most actively researched topics in this area is Web query disambiguation, or automatically determining the intentions and goals of a user who enters an ambiguous query. This is not surprising, given the frequency of ambiguous searches and the unwillingness of users to enter long and descriptive queries. For example, Jansen and Spink [1] found that about 30% of search queries, submitted to several engines, consisted of a single word. Furthermore, Sanderson [2] reports that anywhere between roughly 7% and 23% of the queries frequently occurring in the logs of two search engines are ambiguous, with the average length of ambiguous queries being close to one.

Ambiguity exists not only in cases such as the all-too-familiar "jaguar" example (which can be a cat, car, or operating system), but also in searches that do not appear ambiguous on the surface. Queries that are commonly considered unambiguous often become ambiguous as a result of the wealth of Web sources,

—**Search Session 1**—			—**Search Session 2**—		
98.7 fm	→	`www.star987.com/main.html`	huntsville hospital	→	`www.huntsvillehospital.org`
kroq	→	`www.kroq.com/`	ebay.com	→	`ebay.com`
scrubs	→	`scrubs-tv.com`	scrubs	→	`www.scrubs.com`

Table 1. Two sessions in which the users searched for the query "scrubs."

which examine different aspects of a given topic. For example, as we observed in our data, a search for "texas"[1] may be prompted by at least two different kinds of intentions. In one session, a user who had first searched for "george w. bush" proceeded to search for "texas" and selected `www.tea.state.tx.us`, thus indicating an interest in Texas government agencies. In another session, the user intended to learn about travel to Texas because repeated searches for "georgia travel" were followed by a search for "texas" and a click to `www.tourtexas.com`. This indicates that even a query, such as "texas" that normally refers to a single entity, may become ambiguous.

Most approaches to Web query disambiguation leverage a user's previous interactions with the search engine to predict her intentions when entering an ambiguous query. Typically, the actions of each user are logged over long periods of time, e.g., [3–5]. While techniques that assume the availability of long search histories *for each user* are applicable in some situations, in many cases such approaches may raise privacy concerns and may be difficult to implement for pragmatic reasons. After the release of AOL query log data allowed journalists to identify one user based on her searches [6], many people have become especially wary of having their search histories recorded. To address such concerns, we present an approach that bases its predictions only on short glimpses of user search activity, captured in a brief search session. Our approach relates the current search session to previous *short* sessions of *other* users based on the search activity in these sessions. Crucially, our approach does *not* assume the availability of user identifiers of any sort (i.e. IP addresses, login names, etc.) and thus such information, which could allow user searches to be tracked over long periods of time, does not need to be recorded when our approach is used.

As an example, consider the query "scrubs," which could refer either to the popular television show or to a type of medical uniform. Table 1 juxtaposes the users' actions in two sessions. The sessions are short, with each containing only two searches preceding the ambiguous query; nevertheless, this short glimpse of the users' actions is sufficient to provide an accurate idea of the users' intentions because by examining historical data, one may discover that people who search for radio stations are probably "ordinary" users and would therefore be interested in the television show. On the other hand, by relating Session 2 to sessions of other users who searched for medical-related items, we may be able to predict that the second user has more specialized interests.

[1] We write these queries in lower-case because this is how they were typed by the searchers in our data set.

Our proposed approach is appealing also from a pragmatic standpoint because it does not require search engines to store, manage, and protect long user-specific histories. Identifying users across search sessions is another difficulty arising from methods based on long user-specific search histories. One possibility, to require users to log in before providing personalized search, may be cumbersome. The alternative of using as an identifier the IP address of the computer from which the search was initiated is also unsatisfactory, especially in cases when entire organizations share the same IP address or when all members of a household search from the same computer. Disambiguation techniques that explicitly do not use such identifiers and instead rely only on information from brief sessions avoid such difficulties.

When so little is known about a searcher, the problem of query disambiguation becomes very challenging. In fact, it has previously been argued that "it is difficult to build an appropriate user profile even when the user history is rich" [5]. We develop an approach that successfully leverages the small amount of information about a user captured in a short search session to improve the ranking of the returned search results. Our approach is based on statistical relational learning (SRL) [8] and exploits the relations between the session in which the ambiguous query is issued and previous sessions.

SRL addresses the problem of learning from multi-relational data models that support probabilistic reasoning. SRL is appealing for our problem because, first, the data is inherently relational—there are several types of entities: queries, clicked URLs, and sessions, which relate to each other in a variety of ways, e.g., two sessions may be related by virtue of containing clicks to the same URLs; queries may be related by sharing words. Second, data recording human interactions with a search engine is likely to be noisy. SRL models allow for probabilistic inference, helpful when reasoning from noisy data.

We used Markov logic networks (MLNs) [9]. An MLN consists of a set of weighted formulae in first-order logic and defines a Markov network when provided with a set of constants. The probability of a possible world decreases exponentially in the weight of formulae it fails to satisfy. We chose MLNs because of their generality, their successful application to other language-related tasks, e.g., [11–13], and the availability of a well-maintained code base [14].

2 Related Work

Personalized search is an important problem that has been studied under many settings and assumptions. We review some of this research and draw distinctions between previous work and ours.

Several authors have proposed techniques addressing the case where, for each particular user, a relatively long history of that user's interactions with the search engine is available. Sugiyama *et al.* [3] present a personalization method that builds a user preference model by modeling separately the long-term and "today's" user interests. In addition to relying on long-term records of user activity, their approach also uses the content of browsed pages. In contrast, we are interested in a more light-weight approach that does not necessarily use page content. Sun *et al.* [4] use spectral methods to perform personalization by organizing the data into a three-dimensional tensor comprised of users, queries, and clicked pages. These tensor-based methods are unlikely to be effective in our case because of data sparsity. A comprehensive empirical study of Web search personalization techniques is presented by Dou *et al.* [5]. These techniques also use longer-term histories (up to 12 days) of the same user. The authors find that the best-performing methods are based on the intuition that the Web pages most relevant to a user are those clicked frequently in the past by that user or by related users, where user similarity is measured by estimating user membership in a pre-defined set of categories. Such a strategy is unlikely to work in our setting because the sessions in our data represent one-time interactions that usually do not contain repeated clicks to the same URL. Joachims [15] and Radlinski and Joachims [16] use a clever method for deriving constraints about user preferences by observing whether or not the user clicked on or skipped over particular search results. These preferences are then used to train a system for ranking search results. All work discussed in this paragraph assumes that long-term information about *each user* is available. In contrast, we study the setting where personalization is performed based on records of very short interactions with the search engine.

To the best of our knowledge, the only previous work that targets query disambiguation from short sessions is that of Almeida and Almeida [17] in which users are identified as belonging to a set of communities in order to determine their interests. The authors experimented with data from online bookstore search sites for computer science literature, and their approach is tailored for situations when user interests fall into a small set of categories, organizing users into 10 communities. While in a more restricted application of search, such as specialized book search, this small number of communities may be sufficient to model different aspects of user interests, if, as in our case, the goal is to disambiguate queries in a general-purpose search engine, a small number of communities is likely to be insufficient to effectively model the variety of user interests, and allowing for more communities may be prohibitively costly. Privacy-aware Web personalization has been addressed by Krause and Horvitz [18], whose method considers the privacy cost of a given piece of user information and explicitly models the improvement in personalization versus the cost of the used information. While the ability to trade off performance with cost is highly desirable, their method relies on more information about the user than is available to us.

Query disambiguation is also related to determining user goals and intentions, as done by the TaskPredictor [19], which learns to predict the current task of a user based on the properties of the currently open window, or of an arriving e-mail message. Because training this system requires potentially sensitive information, it is intended to be run on the user's local machine. Another project [20] that relies on sensitive user information studies ways for personalizing Web search by constructing a user profile from long-term observations on the user's activities, ranging from browsing history to e-mail.

An orthogonal issue is producing a diverse set of documents for a given query. Recent work includes that of Chen and Karger [21], whose technique ranks results so as to cover as many different aspects of interest as possible, and that of Yue and Joachims [22] whose technique is based on structural SVMs. A related area is that of clustering search results in groups of common topics. Wang and Zhai [23] use search log data to learn useful aspects of queries in order to cluster them. The ability to disambiguate user intent complements these contributions because it would allow the most relevant cluster, or the most relevant results from a diverse set, to be placed ahead of all others on the search page.

Collaborative filtering, where the goal is to suggest items that would be of interest to a user, based on that and other users' previous preferences, is also related. Early comparative studies of collaborative filtering algorithms include [24, 25]. More recently, Popescul *et al.* [26] and Melville *et al.* [27] proposed approaches that combine collaborative and content-based information in forming recommendations. These were not applied to personalizing Web search.

3 Background

This section provides some necessary background on first-order logic and MLNs.

First-order logic uses 4 types of symbols—constants, variables, predicates, and functions [28]. **Constants** describe the objects in the environment, e.g., `www.ecmlpkdd2009.net` and `1acadc00158440d9` are constants representing a url and a sessionId. **Predicates** represent relations, such as `ClickOn`, and can be thought of as functions that evaluate to `true` or `false`. A **term** is a constant, a variable, or a function applied to terms. Ground terms contain no variables. An **atom** is a predicate applied to terms. A positive (negative) **literal** is a (negated) atom. For example, `ClickOn(www.ecmlpkdd2009.net, 1acadc00158440d9)` is a ground positive literal. Its value is `true` iff `www.ecmlpkdd2009.net` is clicked in session `1acadc00158440d9`. A possible world is a truth assignment to all possible ground literals in an environment. A first-order formula uses conjunction ($\wedge$) and disjunction ($\vee$) to combine positive and negative literals into a logical statement. A **grounding** is a ground formula or literal.

A Markov logic network (MLN) [9] consists of a set of first-order formulae, each of which has an associated weight. MLNs can be viewed as relational analogs to Markov networks whose features are expressed in first-order logic. In this way MLNs combine the expressivity of first-order logic with the ability of probabilistic graphical models to reason under uncertainty.

Let $\mathbf{X}$ be the set of all possible ground literals in the environment, $\mathcal{F}$ be the set of all first-order formulae in the MLN, and w_i be the weight of formula $f_i \in \mathcal{F}$. Then, the probability of a particular truth assignment $\mathbf{x}$ to $\mathbf{X}$ is given by

$P(\mathbf{X} = \mathbf{x}) = \frac{\exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(\mathbf{x})\right)}{\sum_{\mathbf{x}'} \exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(\mathbf{x}')\right)}$ [9], where $n_i(\mathbf{x})$ is the number of groundings of f_i that are true given the truth assignment $\mathbf{x}$ to $\mathbf{X}$. Intuitively w_i determines how much less likely a world is in which a grounding of f_i is not satisfied than one in which it is satisfied. The first-order formulae are called the *structure*. By grounding the formulae of an MLN with the constants in the environment, one defines a Markov network, over which inference can be performed to determine the probability that each of a set of unknown ground literals is true, given the truth values of a set of evidence ground literals. In our case, the evidence literals, which we define in Section 4, provide information on the user activity in the current session and how it relates to previous search sessions, and the goal is to predict the probability that each grounding of the `clickOn` predicate is true. Several algorithms are available to perform inference over a ground MLN. We used MC-SAT [10], which has been demonstrated to give good performance.

4 Proposed Approach

Our general approach follows that of previous applications of MLNs to specific problems, e.g., [12]: we hand-coded the structure of the model as a set of first-order formulae and learned weights for these formulae from the data. The key idea behind our approach is to relate the current, *active*, session A in which an ambiguous query Q is issued to previous, *background*, sessions from historical data, where it is assumed that both the active session and the background sessions are short. Sessions are related by sharing various types of information. We define the following predicates to capture these relationships. Since every training/testing example refers to a single (Q,A) pair, A and Q are implicit in the example and do not need to appear as arguments of the predicates.
-`result(r)`: `r` is a search result for Q.
-`choseResult(s, r)`: Background session `s` clicked on `r` after searching for Q.
-`clickOn(r)`: User in session A clicks on result `r` in response to the search for Q.
-`sharesClick(s, d)`: Sessions `s` and A share a click to URL with hostname `d`.
-`sharesKeywordBtwnClicks(s, k)`: Background session `s` and A share a keyword `k`, found in the hostnames of clicked URLs in each of the sessions.
-`sharesKeywordBtwnClickAndSearch(s, k)`: Background session `s` and A share a keyword `k`, found in the hostname of a clicked URL in A and a search in `s`.
-`sharesKeywordBtwnSearchAndClick(s, k)`: Background session `s` and A share a keyword `k`, found in a search in A and the hostname of a clicked URL in `s`.
-`sharesKeywordBtwnSearches(s, k)`: Sessions `s` and A share a keyword `k` that appeared in searches in both sessions.
-`clicksShareKeyword(r, d, k)`: Keyword k appears in the hostname of both result r and previous click d from session A.
-`clickAndSearchShareKeyword(r, s, k)`: Keyword k appears in the hostname of result r and in previous search query s from session A.

Fig. 1 illustrates the predicates used to relate two sessions. The last two predicates capture information local to the active session. In the active session A, only the clicks and searches temporally *preceding* Q are used. For the predicates in which a keyword relates two sessions, we used only keywords that appeared at least 100 times (i.e., we removed keywords appearing less than 0.00083% of the time) and at most $10,000$ times (i.e., we removed the top 61 most popular

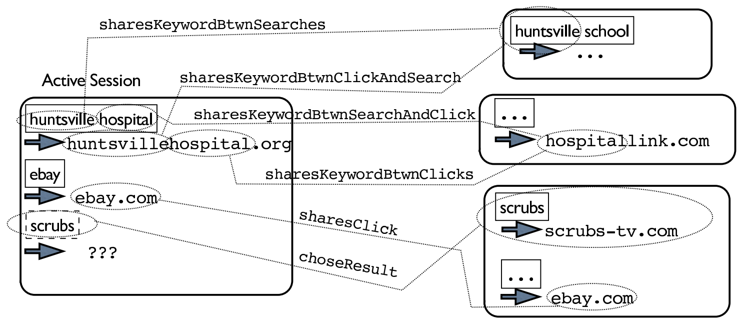


Fig. 1. An illustration of predicates that relate sessions. Tokens in boxes represent queries, whereas tokens preceded by an arrow represent the clicked result. The *active* session, on the left, is related to some of the *background* sessions, on the right, by shared clicks or keywords. Not all possible relations are drawn.

1: $\texttt{result(r)} \wedge \texttt{sharesClick(s, d)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
2: $\texttt{result(r)} \wedge \texttt{sharesKeywordBtwnClicks(s, k)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
3: $\texttt{result(r)} \wedge \texttt{sharesKeywordBtwnClickAndSearch(s, k)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
4: $\texttt{result(r)} \wedge \texttt{sharesKeywordBtwnSearchAndClick(s, k)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
5: $\texttt{result(r)} \wedge \texttt{sharesKeywordBtwnSearches(s, k)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
6: $\texttt{result(r)} \wedge \texttt{choseResult(s, r)} \wedge \texttt{clickOn(r)}$
7: $\texttt{result(r)} \wedge \texttt{clicksShareKeyword(r, d, k)} \wedge \texttt{clickOn(r)}$
8: $\texttt{result(r)} \wedge \texttt{clickAndSearchShareKeyword(r, s, k)} \wedge \texttt{clickOn(r)}$
9: $\texttt{result(r1)} \wedge \texttt{result(r2)} \wedge \texttt{r1} \neq \texttt{r2} \wedge \texttt{clickOn(r1)} \Rightarrow \neg\texttt{clickOn(r2)}$

Table 2. Formulae included in the model.

keywords) over our training data. This was done to avoid rare or misspelled keywords and to make the size of the data more manageable. The goal is to predict the `clickOn(r)` predicate, given as evidence the values of the remaining ones. The search results available for a given query are then ranked by the predicted probability that the user will click on each of them.

4.1 Model Structure

This section describes the formulae used in our MLN models.

Collaborative Formulae: The collaborative formulae, shown in lines 1-5 of Table 2, draw inferences about the interests of the active user based on the choices made by related users from background sessions. For example, formula 1 establishes a relationship between the event that the active user chooses result r and the event that the user in a previous session s, related to the active session by sharing a click to a URL with hostname d, chose result r after searching for the current ambiguous query. This formula exploits one type of relation between the active session and background sessions to provide evidence of the active user's intentions. This formula is always false when one of the first three evidence predicates is false, and in such cases it does not influence the probability that a particular search result is chosen; i.e., this formula plays a role only for background sessions that share clicks with the active session and chose a particular result r. The larger the number of such sessions, the stronger the belief that the active user will also pick r. Formulae 2-5 encode analogous dependencies using each of the remaining session-relating predicates.

Popularity Formula: Formula 6 in Table 2 encodes the intuition that the user will click the result that was the most popular among background users that searched for this ambiguous query. As before, the result for which there are the largest number of clicks in background data, and thus the largest number of groundings of this formula that are not falsified by the evidence, will have the largest probability of being clicked.

Local Formulae: Formulae 7-8 in Table 2 use information local to the active session to predict the user's preferences. Formula 7 (8) states that the user will click a result that shares keywords with a previous result (search) from the active session. We clarify that keywords were *not* extracted from the pages to which a URL points, but only from the URL itself because we are interested in developing a light-weight re-ranker. Because in our setting sessions are very short, we do not expect the local formulae to contribute much to the overall model performance. We include them in order to verify this.

Balance Formula: Formula 9 in Table 2 sets up a competition among the possible results by stating that if the user clicks one of the results, the user will not click another one. This formula prevents all possible results from obtaining a very high probability of being clicked. This makes the model more discriminating and allows the same set of weights to perform well even as the number of groundings of the other formulae varies widely across active sessions.

These formulae encode "rules of thumb" and useful features, which we expect will hold in general, but may sometimes be violated, e.g., the balance formula is violated when a user clicks more than one result for a query. The ability of MLNs to combine such varied sources of information effectively and in a principled way is one of the main considerations that motivated our choice of model. Using these formulae, we defined three MLNs:
MLN 1 – Purely Collaborative: Contains only the collaborative formulae (1-5) and the balance formula (9).
MLN 2 – Collaborative and Popularity: Contains formulae 1-6 and the balance formula (9).
MLN 3 – Collaborative, Popularity, and Local: Contains all formulae. It can be viewed as a mixed collaborative-content-based model, e.g., [26, 27].

4.2 Weight learning

To learn weights for the structures defined above, we used the contrastive divergence algorithm (CD) described by Lowd and Domingos [29]. CD can be viewed as a voted-perceptron-like gradient descent algorithm in which the gradient for updating the weight of formula C_i is computed as the difference between the number of true groundings of C_i in the data and the expected number of true groundings of C_i, where the expectation is computed by carrying out a small number of MCMC steps over the model using the currently learned weights. Like Lowd and Domingos [29], we computed the expectations with MC-SAT [10]. We used the implementations of these algorithms in the Alchemy package [14], except that we adapted the existing implementation of CD so that learning can proceed in an online fashion, considering examples of sessions containing ambiguous queries one by one. This was done because otherwise our data was too large to fit in memory. We set the learning rate to 0.001 and the initial weight of formulae to 0.1 and kept all other parameters at their default values. Parameter values were selected on a validation set, strictly disjoint from our test set.

5 Data and Methodology

We used data provided by Microsoft Research containing anonymized query-log records collected from MSN Search in May 2006. The data consists of timestamped records for individual short sessions, the queries issued in them, the URLs clicked for each query, the number of results available for each query and the position of each result in the ranked results. We removed queries for which nothing was clicked. The average number of clicked results per session, over all sessions in the data, is 3.28. The data does not specify what criteria were used to organize a set of user interactions into a session; e.g., we do not know how multiple open tabs in a browser were treated. Although some of the sessions may belong to the same users, the data excludes this information through the lack of user-specific identifiers. This dataset therefore perfectly mirrors the scenario of disambiguating user intent from short interactions that we address in this research. Because there is a one-to-one correspondence between users and sessions, we will use these two terms interchangeably.

The data has two main limitations. First, it does not state which search queries are ambiguous. Automatically detecting ambiguity from user behavior is an interesting research question but is not the focus of this work. We therefore employed a simple heuristic to obtain a (possibly noisy) set of ambiguous queries, using DMOZ (`www.dmoz.org`): a query string is considered ambiguous if, over all URLs clicked after searching for this string, at least two fall in different top-level DMOZ categories. This heuristic does not require human effort beyond that already invested in constructing DMOZ. We did not include DMOZ category information into our models because many Web pages are not classified in the hierarchy. We limited ourselves to strings containing up to two words, thus obtaining $6,360$ distinct ambiguous query strings. Limiting the length of potentially ambiguous queries to two was motivated by the fact that most ambiguity occurs in short queries. For example Sanderson [2] found that the average length of ambiguous queries in two search log datasets ranges from 1.02 to 1.26 words. Queries of length at most two constituted 43.7% of all queries in our data. Of these queries, using the above method, we identified 2.4% as ambiguous, which

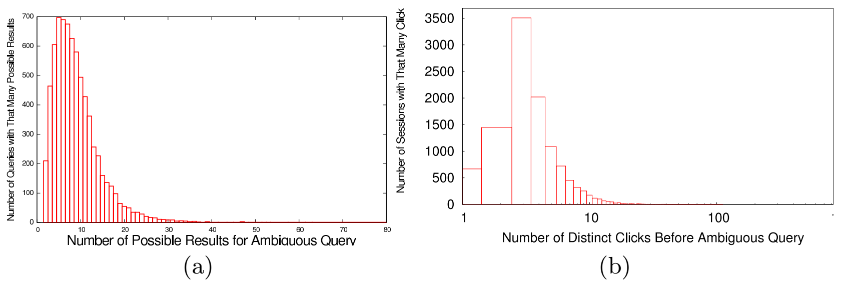


Fig. 2. Histograms showing (a) the distribution over the number of possible results available for an ambiguous query and (b) the distribution over the number of clicks preceding an ambiguous query in the test data. The X axis in (b) is drawn in log-scale.

agrees with the statistics reported by Sanderson, who found that between 0.8% and 3.9% of all queries are ambiguous [2]. [2]

Another limitation is that our data does not list all URLs presented to the user after a search but just the clicked ones. To overcome this, we assumed that the set of all URLs clicked after searching for a particular ambiguous query string, over the entire dataset, was the set of results presented to the user. Our approach contrasts with that used in previous work, e.g., that of Dou *et al.* [5], in which missing possible results lists are generated by separately querying the MSN search engine (on which data was collected) for each query. Although the queries were performed less than a month after the data was collected, the authors found that 676 queries from 4,639 "lost the clicked web pages in downloaded search results." Because in our case almost 3 years have passed since the MSN06 data was collected, we preferred the simpler approach based on the available data. With this method, the average number of possible results for an ambiguous query string was 9.10. Figure 2 (a) shows the distribution over the number of ambiguous queries for which we have a particular number of possible results. Although this heuristic is imperfect, it is likely to bias the results *against* our proposed solution—since every possible result was found to be relevant by at least one user, our systems cannot get high scores by simply separating the useful results from the totally irrelevant ones.

Figure 2 (b) shows the distribution over the number of clicks preceding an ambiguous query in our test data. As can be seen, our test sessions, are indeed very short. Several of the predicates we define use keywords. To generate a list of keywords, we performed a pass over all training sessions. Any token separated by spaces was considered a keyword. As mentioned in Section 4, we then kept keywords that appeared at least 100 times and at most 10,000 times. To determine which keywords occur in a given hostname, we first use the non-alphanumeric characters in the hostname to break it down into pieces and then match each

[2] In the Introduction, we cited Sanderson's findings for *frequently occurring* queries, whereas here we refer to his findings over *all* queries.

piece with keywords such that as much of the piece is covered as possible, using the smallest number of keywords.

To ensure a fair evaluation, the data was split into training and testing periods. The training period was used for training, validation, keyword generation, and *idf* [30] calculations (*idf*s were used by one of the baselines) and consisted of the first 25 days of data. The remaining 6 days were reserved for testing. Sessions that started in the training period and ended in the test period were discarded to avoid contaminating the test data. As validation/testing examples we used sessions that contained an ambiguous query from the training/testing periods respectively. To decrease the amount of random noise in the results, we removed from the test set sessions that contained no relational evidence, i.e., we removed the sessions that contain no true groundings of the `sharesKeyword/Click` predicates introduced in Section 4. In this way we obtained $11,234$ test sessions, which constitutes 72% of the available test sessions. The distribution over the number of previous clicks in these sessions is shown in Figure 2 (b). As can be seen, the peak is at 3 distinct clicks before the ambiguous query.

During testing, only the information *preceding* the ambiguous query in the active test session is provided. The set of possible results for this ambiguous query string is given, and the goal is to rank these results based on how likely it is that they represent the intent of the user. The user may click more than one result after searching for a string. This behavior might be indicative of at least two possible scenarios: either the user is performing an exploratory search and all clicked results were relevant, or the user was dissatisfied with the results and kept clicking until finding a useful one. Since the data does not indicate which of these scenarios was the case, we treated all results clicked by the user after searching for the ambiguous query as relevant to his or her intentions. This presents yet another source of noise, and in the future we plan to explore approaches similar to the implicit feedback techniques described by Radlinski and Joachims [16] to disentangle these possibilities, although the exact method introduced by these authors would not be applicable to our data because it requires the availability of an *ordered* list of the results returned to the user by the search engine.

Learning was performed as described in Section 4.2. To evaluate the learned models, we used Alchemy's implementation [14] of the MC-SAT algorithm [10] for inference. During inference, we ran for 1,000 burn-in steps and 10,000 sampling steps. All other inference parameters were kept at their Alchemy defaults.

Evaluation Metrics: For evaluation purposes, query disambiguation can be viewed as an information retrieval problem: rank the set of possible results so that the URLs reflecting the user's intentions appear as close to the top as possible. Thus, we used standard information retrieval metrics to evaluate the performance of our system [30] (Chapter 8):

(MAP) Area under the precision-recall curve, which is identical to the $\underline{\text{M}}$ean $\underline{\text{A}}$verage $\underline{\text{P}}$recision metric, commonly used in IR. The MAP score is computed over a set of test instances T as follows: $\text{MAP}(T) = \frac{1}{|T|} \sum_{t \in \text{T}} \frac{1}{|R_t|} \sum_{r \in R_t} \text{P@}r$, where R_t is the set of possible results for the t-th test instance and P@r is the precision of the top r results: $\text{P@}r = \frac{\text{Num relevant docs among the top } r}{r}$.

(AUC-ROC) Area under the ROC Curve, which can be viewed as representing the mean average true negative rate. Using the notation from above, this metric is computed as follows: AUC-ROC$(T) = \frac{1}{|T|} \sum_{t \in \mathrm{T}} \frac{1}{|R_t|} \sum_{r \in R_t} \mathrm{TN}@r$, where TN@$r$ is the true negative rate of the top r results, defined as TN@$r = \frac{\text{Num irrelevant docs in positions } > r}{\text{Total num irrelevant docs}}$.

Intuitively, the MAP measures how close the relevant URLs are to the top. One disadvantage of this metric in our case is that it is insensitive to the number of results to be ranked. For example, ranking a relevant result in the second position obtains the same score both when the number of possibilities is 2 and when it is 100, even though in the second case the task is clearly more difficult. Assuming that the user starts scanning the page of returned results from top to bottom and does not consider any results appearing after the relevant ones, the AUC-ROC intuitively represents the percentage of irrelevant results that were *not* seen by the user. Thus, a random ranker would obtain an AUC-ROC of 0.5. Another useful characteristic of this measure is that unlike the MAP, it is sensitive to the number of possible results that are to be ranked.

A final issue is how to break ties when a relevant result has the same score as some irrelevant results. We report the **average case** in which the relevant result is placed in the middle position within the group of results with equal scores. For the most interesting systems, we also report the **worst case** in which the relevant result is placed last within the group of results that share scores. This is motivated by the goal of performing effective personalization *consistently*. The best case is not interesting because for it perfect performance can be obtained by giving all results the same score.

Systems Compared: We compared the MLNs from Section 4 to:

Random: Ranks the possible results randomly.

Collaborative-Pearson: Implements a standard collaborative filtering algorithm [25] that weights each previous user based on the Pearson correlation between the preferences (i.e. clicks) of that user and the active user. We considered a clicked result to have rating 1, and an unclicked result that was clicked by another user for the same query to have rating 0, and all other results to be unrated. The n closest neighbors are chosen (we used $n = 30$ following [25]), and the prediction that a given result is selected is formed as a weighted average of the deviations from the mean of each neighbor.

Collaborative-Cosine: Identical to **Collaborative-Pearson** except that it computes the similarity between the active user and a previous user as the cosine similarity between the *idf*-weighted vectors of their clicked results.

Popularity: Ranks each result according to the number of previous sessions that searched for the ambiguous query and chose it.

6 Results

Table 3 (a) presents the performance when ties among results with the same score are broken as in the average case. The **Collaborative-Pearson** baseline performs no better than **Random** on AUC-ROC and only slightly better than

System	**MAP**	**AUC-ROC**
Random	0.317	0.502
Collaborative-Pearson	0.333	0.502
Collaborative-Cosine	**0.360**	**0.521**
Popularity	**0.389**	**0.575**
MLN 1	0.375	0.563
MLN 2	0.386	**0.587**
MLN 3	0.366	0.583

(a)

System	**MAP**	**AUC-ROC**
Popularity	0.380	0.525
MLN 1	0.373	**0.563**
MLN 2	**0.385**	**0.586**
MLN 3	0.355	0.572

(b)

Table 3. (a) Results over all test sessions that contain an ambiguous query when ties in ranking are broken as in the **(a) average case** and **(b) worst case.** Numbers in bold present significant improvements over all preceding systems at the 99.996% level with a paired t-test. Additional significant differences are: in (a) **MLN 1** is a significant improvement over all baselines except **Popularity**, and **MLN 2** improves significantly over all preceding systems except for **Popularity** also in terms of MAP; in (a), there is no significant difference between the MAP scores of **Popularity** and **MLN 2**; in (a) and (b) the MAP score of **Popularity** is significantly higher than that of **MLN 1.**

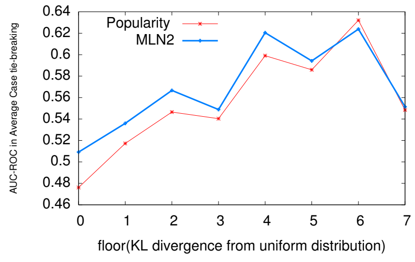
Random on MAP. Switching to cosine similarity in **Collaborative-Cosine** gives modest (but significant at the 99.996% level according to a paired t-test) improvements. The **Popularity** baseline is very strong and outperforms the other baselines, as well as **MLN 1**. However, combining popularity with relational information in **MLN 2** leads to significant gains in performance, and **MLN 2** achieves a significantly higher AUC-ROC score. **MLN 2**, our strongest model, highlights the main advantage of using MLNs: we were able to significantly improve **MLN 1** by incorporating a reliable source of information simply by adding the popularity formula to the model. Finally, as expected, we observe that adding local formulae in **MLN 3** does not improve performance. This demonstrates that the interactions of the active user prior to the ambiguous query are not directly helpful for determining intent and occurs as a result of the brevity of sessions in our data (cf. Figure 2 (b)). The inefficacy of local formulae may also be due to the fact that a session may continue when the user is dissatisfied with the results obtained so far. It is interesting to contrast this result with the findings of Dou *et al.* [5] who experimented with much longer sessions (up to 12 days) and reported that the previous interactions of the active user presented a very strong signal for personalization purposes. This emphasizes a fundamental difference in our assumptions about the data compared to previous research: because in our case user-specific session information is so limited, we cannot rely on only using the past preferences of the active user and must instead exploit relations to other, historical, users.

Next, we analyze in more detail the performance of the MLN systems to that of **Popularity**, which is the strongest baseline. Table 3 (b) presents the performance over all test sessions when ties in ranking are broken as in the worst case. As can be seen, **Popularity**'s AUC-ROC score decreases sharply, whereas the MLN models maintain their performance to almost the same level as in the average case. This behavior is observed partly because **Popularity** introduces many more ties among the scores of possible results than do the MLN models. In particular, averaged over all test sessions, the ratio between the number of possible results and the number of distinct scores for **Popularity** was 1.8, whereas for **MLN2** it was just 1.02. These results indicate that **Popularity**'s behavior is erratic and can, for the same user and the same query, lead to rankings that vary highly in quality. This kind of behavior can give the perception of poor quality to a frequent user. On the other hand, the MLN models are consistent, maintaining the quality of their rankings in the worst case.

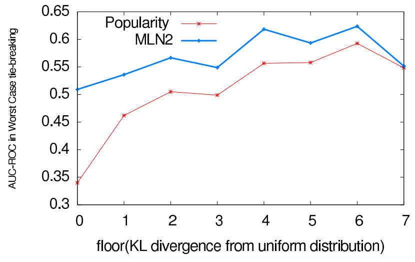
Finally, we compare the performance of **Popularity** to that of **MLN 2** while varying the degree to which some of the possible results for an ambiguous query dominate in popularity over the rest. We formalized this as follows. Let q_Q be the empirical distribution over the results clicked for an ambiguous query Q. This distribution was measured empirically on the training data, i.e., for every ambiguous query, we determined from the training sessions the proportion of time each potential search result was clicked. We then separated the test examples into bins, such that bin i contains all test sessions s for which $\lfloor KL_{q_Q||\text{uniform}} \rfloor = i$, where Q is the ambiguous query in session s and $KL_{q_Q||\text{uniform}}$ is the KL divergence of q_Q to the uniform distribution. In other words, bin 0 contains the sessions in which the possible results for the ambiguous query were all chosen with roughly the same frequency. Higher-numbered bins contain sessions in which one of the search results strongly dominates in popularity over the other possibilities. When this is the case, predicting just based on the popularity of a result gives good performance. The more challenging scenario occurs in the lower-numbered bins where the preferences over possible results are more uniformly distributed. Figure 3 compares **Popularity** to **MLN 2** when ties in ranking are broken for the average and worst cases. **MLN 2** maintains a lead over **Popularity** until the last two bins in which the distribution over possible results is furthest from uniform. As we expect, the difference between the performance of the two systems shrinks as we move to higher-numbered bins, and **MLN 2** has a greater advantage over **Popularity** in the lower-numbered bins in which the need to disambiguate is more pressing. The sharp drop in accuracy observed in bin 7 is due to the fact that one of the ambiguous queries occurring in sessions in this bin was overwhelmingly followed by clicks to what seems to be a newly appearing Web page during the test period. That page was selected only 3 times in the training period while the most popular page in the training period was selected more than 2000 times. As a final but important note, inference over the learned models was very efficient and completed in the order of a second.

7 Conclusions and Future Work

We addressed Web query disambiguation in the challenging setting when the only information available about any particular user is that captured in a short search session of 4–6 previous searches on average. Using the language of MLNs, we



(a)



(b)

Fig. 3. AUC-ROC when ranking ties are broken so as to simulate the **(a) average case** and **(b) worst case** for different bins of KL divergence of the distribution over possible results to uniform.

developed an approach that draws heavily on different types of relations between search sessions and demonstrated that our approach significantly outperforms several natural baselines by successfully combining the inferences of collaborative and popularity formulae. In this way, we provided evidence that despite the sparseness and noise inherently present in a short search session, it is possible to output meaningful predictions about a searcher's underlying interests.

Here our goal was a light-weight approach to Web query disambiguation. In the future, we would like to experiment with richer sources of information, such as the actual content of clicked pages. A second avenue for future work involves improving supervision by discovering ways to decrease the amount of noise in the data and developing learning algorithms that are more tolerant to noise.

Acknowledgment

We thank Tuyen Huynh, Joe Reisinger, and the anonymous reviewers for their helpful comments. This research is supported by a gift from Microsoft Research and by ARO grant W911NF-08-1-0242. Experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

1. Jansen, B.J., Spink, A.: How are we searching the World Wide Web? A comparison of nine search engine transaction logs. Information Processing and Management **42** (2006) 248–263.
2. Sanderson, M.: Ambiguous queries: Test collections need more sense. SIGIR-08.
3. Sugiyama, K., Hatano, K., Yoshikawa, M.: Adaptive web search based on user profile constructed without any effort from users. WWW-04.
4. Sun, J., Zeng, H., Liu, H., Lu, Y., Chen, Z.: CubeSVD: A novel approach to personalized web search. WWW-05.
5. Dou, Z., Song, R., Wen, J.: A large-scale evaluation and analysis of personalized search strategies. WWW-07.

6. Barbaro, M., Zeller, T.: A face is exposed for AOL searcher no. 4417749. New York Times (August 2006) `http://www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000`. Accessed on 16 Oct. 2008.
7. Conti, G.: Googling considered harmful. In: New Security Paradigms Workshop, Dagstuhl, Germany (September 2006).
8. Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. MIT Press, Cambridge, MA (2007).
9. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62** (2006).
10. Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. AAAI-06.
11. Singla, P., Domingos, P.: Entity resolution with Markov logic. ICDM-06.
12. Poon, H., Domingos, P.: Joint inference in information extraction. AAAI-07.
13. Wu, F., Weld, D.: Automatically refining the Wikipedia infobox ontology. WWW-08.
14. Kok, S., Singla, P., Richardson, M., Domingos, P.: The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington (2005) `http://www.cs.washington.edu/ai/alchemy`.
15. Joachims, T.: Optimizing search engines using clickthrough data. KDD-02.
16. Radlinski, F., Joachims, T.: Query chains: Learning to rank from implicit feedback. KDD-05.
17. Almeida, R.B., Almeida, V.A.F.: A community-aware search engine. WWW-04.
18. Krause, A., Horvitz, E.: A utility-theoretic approach to privacy and personalization. AAAI-2008.
19. Shen, J., Li, L., Dietterich, T.G., Herlocker, J.L.: A hybrid learning system for recognizing user tasks from desktop activities and email messages. IUI-06.
20. Teevan, J., Dumais, S.T., Horvitz, E.: Personalizing search via automated analysis of interests and activities. SIGIR-05.
21. Chen, H., Karger, D.R.: Less is more: Probabilistic models for retrieving fewer relevant documents. SIGIR-06.
22. Yue, Y., Joachims, T.: Predicting diverse subsets using structural SVMs. ICML-08.
23. Wang, X., Zhai, C.: Learn from web search logs to organize search results. SIGIR-07.
24. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research (1998)
25. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. SIGIR-99.
26. Popescul, A., Ungar, L.H., Pennock, D.M., Lawrence, S.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. UAI-2001.
27. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. AAAI-02.
28. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. 2 edn. Prentice Hall, Upper Saddle River, NJ (2003).
29. Lowd, D., Domingos, P.: Efficient weight learning for Markov logic networks. PKDD-07.
30. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to Information Retrieval. Cambridge University Press (2008).