Learning User Clicks in Web Search

Ding Zhou, Levent Bolelli, Jia Li, C. Lee Giles, Hongyuan Zha
Department of Computer Science and Engineering
Department of Statistics
College of Information Sciences and Technology
The Pennsylvania State University, University Park, PA 16802

Abstract

Machine learning for predicting user clicks in Web-based search offers automated explanation of user activity. We address click prediction in the Web search scenario by introducing a method for click prediction based on observations of past queries and the clicked documents. Due to the sparsity of the problem space, commonly encountered when learning for Web search, new approaches to learn the probabilistic relationship between documents and queries are proposed. Two probabilistic models are developed, which differ in the interpretation of the query-document co-occurrences. A novel technique, namely, *conditional probability hierarchy*, flexibly adjusts the level of granularity in parsing queries, and, as a result, leverages the advantages of both models.

1 Introduction

Predicting the next click of a user has gained increasing importance. A successful prediction strategy makes it possible to perform both prefetching and recommendations. In addition, the measurement of the likelihood of clicks can infer a user's judgement of search results and improve ranking.

As an important goal of Web usage mining [Srivastava *et al.*, 2000], predicting user clicks in Web site browsing has been extensively studied. Browse click prediction typically breaks down the process into three steps: (1) clean and prepare the Web server log data; (2) extract usage patterns; and (3) create a predictive model. There have been a variety of techniques for usage pattern extraction, user session clustering [Banerjee and Ghosh, 2001; Gunduz and Ozsu, 2003], page association rule discovery [Gunduz and Ozsu, 2003], Markov modeling [Halvey *et al.*, 2005], and implicit relevance feedback [Joachims, 2002].

Contrary to the rich research in click prediction in Web site browsing, the prediction of user clicks in Web search has not been well addressed. The fundamental difference between predicting click in Web search and Web site browsing is the scale of the problem space. The modeling of site browsing typically assumes a tractable number of pages with a reasonable number of observations of the associations among these pages. This assumption, however, no longer holds in the Web search scenario. The vast amount of documents quickly result in very high dimensional space and hence sparsifies the problem space (or equivalently leads to a lack of observations), which effects model training. In order to reduce the problem space, some previous work first classifies the large document collection into a small number of categories so as to predict topic transition in Web search [Shen *et al.*, 2005]. However, since the prediction of user clicks requires the granularity of a single document, we do not consider issues in document clustering but rather work on the full space of the document collection.

Consider the problem of learning from Web search logs for click prediction. We define the task as learning the statistical relationship between queries and documents. The primary assumption is that the clicks by users indicate their feedback on the quality of query-document matching. Hypothesize that the vocabulary of query in terms of words remains stable over a period of time. Denote this by Σ. Ideally, if we were able to collect sufficient instances for every combination of words (2^{Σ}) and the clicked documents with these queries, we would be able to estimate the probability of future clicks based on past observations. The feasibility of the approach, however, relies on the assumptions that the training data exhausts all the possible queries.

However, since the number of different queries in Web search explodes with emerging concepts in user knowledge and randomness in the formation of queries, the tracking of all possible queries becomes infeasible both practically and computationally. For example, Fig. 1 illustrates the increase in the number of distinct queries submitted to CiteSeer over a period of three months. The linear growth of distinct queries over time indicates that we can hardly match even a small fraction of the new queries exactly with the old queries. As a result, prediction cannot be performed for new queries, yielding low predictability. Furthermore, many new documents are being clicked even after having accumulated documents that were clicked on over considerably long time. Learning of the relationship between the complete queries and the documents is consequently a highly challenging problem.

An alternative naive solution to the lack of training instances is to break down queries into words. An observation of a query and the corresponding clicked document (*query-document pair*) is transformed into several independent observations of word and document, i.e. *word-document pair*.

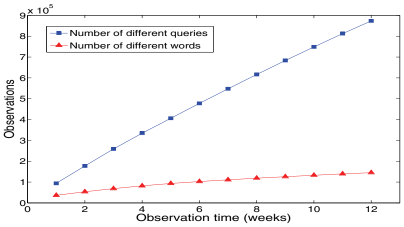


Figure 1: Number of distinct queries and words in CiteSeer over 12 weeks.

This solution can predict unknown queries as long as the new query contains some known words. However, this solution suffers in prediction accuracy due to the discarding of word proximity information in existing queries.

We first propose two probabilistic models, namely *full model* and *independent model*, in order to capture the ideas behind the above intuition, which interpret *query-document pairs* differently. The *full model* tends to achieve high prediction accuracy if sufficient training instances are provided, but it cannot be applied when new queries are encountered. On the other hand, the *independent model* yields high predictability while suffering in accuracy. In order to tradeoff the prediction accuracy with predictability, we suggest a new *conditional probability hierarchy* technique that combines the two models. We are not aware of any previous work studying the click prediction problem for Web search on such large scale search logs. In addition, as a by-product of the new combination approach, n-grams of words are discovered incrementally.

2 Related Work

User click prediction is based on understanding the navigational pattern of users and, in turn, modeling observed past behavior to generate predictive future behavior. Usage pattern modeling has been achieved traditionally by session clustering, Markov models, association rule generation, collaborative filtering and sequential pattern generation.

Markov model based prediction methods generally suffer from high order, usually needing clustering of user clicks to reduce the limitations stemming from high state-space complexity and decreased coverage. On the other hand, lower order Markov models are not very accurate in predicting user's browsing behavior, since these models keep a small window to look back in the history, which is not sufficient to correctly discriminate observed patterns [Deshpande and Karypis, 2004]. Markov models seem to be more suitable for mobile applications [Halvey *et al.*, 2005] where the number of states is low due to the few links a user can navigate.

User session clustering [Banerjee and Ghosh, 2001; Gunduz and Ozsu, 2003] identifies users' navigational paths from web server logs and defines session similarity metrics based on similarity of the paths and time spent on each page on a path. Clustering is employed on the similarity graph which are utilized for predicting users' requests.

Beeferman and Berger [Beeferman and Berger, 2000] proposed query clustering based on click-through data. Each (query, clicked URL) pair is used to construct a bipartite graph and queries that result in clicking to the same URL are clustered. The content features in the queries and the documents are discarded. Joachims [Joachims, 2002] introduces a supervised learning method that trains a retrieval function based on click-through data that takes the relative positions of clicks in a rank as training data set. We pursue a different approach that seeks to learn the statistical relationship between queries and user actions in this paper.

3 Problem Statement

Descriptions of the problem is formalized. Let $\aleph$ be the document full set. $\{d_i\} = D \in \aleph$ denotes the set of documents that have ever been shown to users in search results and $C \subseteq D$ is the document set that has been clicked on. $\Sigma = \{w_i\}$ denotes the word vocabulary. The query set is $Q = \{q_j\}$, where $q_j = \{w_{j1}, ..., w_{jk}\} \in 2^{\Sigma}$. Our observation of Web search log is abstracted as a sequence of query-document pairs: $\Gamma \subseteq Q \times C$. The posterior probability for observing each click on a certain document d is $P(d|q, \mathbf{d_q})$, where $\mathbf{d_q}$ represents the document list returned with query q.

The problem is to predict $P(d|q, \mathbf{d_q}, \Gamma)$, which measures the probability of user clicking on document d for query q when presented with the result list $\mathbf{d_q}$. The prediction of user click for query q becomes: $\hat{d} = \arg\max_d P(d|q, \mathbf{d_q}, \Gamma)$, where $d \in \mathbf{d_q}$ and Γ is the observation of query-document click so far.

4 Probabilistic Models

The two probabilistic models we propose are for acquiring estimations of $P(d|q)$ for each d in $\mathbf{d_q}$ given query q. Depending on interpretations of a click on d for q, two types of models are introduced, namely, the *independent model* and the *full model*.

4.1 Independent model

When we observe a *query-document pair* $\langle d, q \rangle$, how do we interpret it? The independent model we propose firstly assume each word in q is independent of each other. Formally, the *independent model* we define interprets an instance $\langle d, q \rangle$ as observing d and given d, observing the words $w_1, ..., w_k$ independently.

Let us consider how $P(d|q)$ is estimated under the *independent model*. In the case where the query consists of k words $q = [w_1, ..., w_k]$ and the clicked document is d, we measure the probability $P(d|w_1, ..., w_k)$ as:

$$
\begin{aligned}
P(d|w_1, ..., w_k) &= \frac{P(d, w_1, ..., w_k)}{P(w_1, ..., w_k)} \\
&= \frac{P(d)P(w_1|d)...P(w_k|d)}{\sum_{d \in D} P(d, w_1, ..., w_k)} && (1) \\
&= \frac{P(d) \prod_{i=1}^{k} P(w_i|d)}{\sum_{d' \in D} P(d') \prod_{i=1}^{k} P(w_i|d')} && (2)
\end{aligned}
$$

Eq. 1 is obtained using the Bayes formula. The transition from Eq. 1 to Eq. 2 assumes the conditional independence between w_i and w_j according to the model definition. $P(d)$ is the marginal probability of document d, which is proportional to the number of occurrences of d. The above derivations show that for the purpose of calculation, each $\langle d, q\rangle$ can be broken down to a collection of independent instances $\langle d, w_i\rangle$, where $i = 1, ..., k$.

As indicated in Eq. 2, the estimation requires calculation of $P(d)$ and $P(w_i|d)$. Since we are able to keep track of $P(d)$ easily, the computation for Eq. 2 then transforms to $P(w_i|d)$. In § 6, we discuss the Bayesian estimation of $P(w_i|d)$ to address the sparsity in training data.

4.2 Full model

While the *independent model* treats each query as a set of independent single words, the *full model* reflects the other extreme where all words within a query are treated as a group. In our definition, the *full model* treats the q in an instance $\langle d, q\rangle$ as a singleton. The combination of words in q is regarded as an entity.

The *full model* emphasizes a query as a group and hence yields high prediction accuracy, provided that a large amount of queries have been observed with large supports. However, as noted before, the number of different queries grows so quickly that the *full model* always suffers the lack of training data in practice. In the next section, we will introduce the method to combine the *full model* and the *independent model* in order to address the sparsity issue.

5 Probability Hierarchy

Ideally, if we are able to observe all future queries with the documents being clicked in the log file, especially with decent number of instances, the *full model* alone can be sufficient. However, new queries keep emerging which dramatically enlarges the problem space if we simply learn from the co-occurrence $\langle d, q\rangle$. In order to address such sparsity issues, we introduce this new *conditional probability hierarchy (CPH)* method, which recursively generates multiple intermediate combinations between the full and independent models. The shrinkage rate is tunable according to the supports of the *full model.*

5.1 Hierarchical shrinkage

The *conditional probability hierarchy (CPH)* we propose combines the *full model* and the *independent model.* It starts with treating a query as a set of independent words, i.e. obtaining $P(d|w_j)$. Hierarchically, words are merged into *n-grams* [1] (or *units*) of increasing length, getting $P(d|u_k)$. The final estimation of $P(d|q)$ is the hierarchical combination across several levels.

Fig. 2 illustrates an example estimation of $P(d'|a, b, c, d, e)$. Suppose we have a document d and a word sequence u (u can be either a single word of a multiple word query). Let $P(d|u)^\alpha$ denote the estimation of $P(d|u)$ under the *full model* and $P(d|u)^\beta$ under *independent model.* We use $P(d|u)^h$ to denote the estimation of $P(d|u)$ with the CPH. A query $q = \langle a, b, c, d, e\rangle$ consists of five individual words a, b, c, d and e. Suppose we already have the *full model* estimation of $P(d'|w)^\alpha$ for $w = a, b, c, d$ and e. We set the $P(d'|w)$'s at L_1 as equivalent to *full model* estimation:

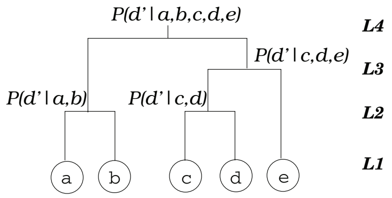


Figure 2: CPH: Hierarchical combination of conditional probabilities at different levels. To estimate $P(d'|a, b, c, d, e)$ for document d' and query $\langle a, b, c, d, e\rangle$, $P(d'|a, b)$ and $P(d'|c, d, e)$ are combined. $P(d'|a, b)$ is the combination of $P(d'|a)$, $P(d'|b)$ and $n(d', a, b)$, where $n(d', a, b)$ denotes the number clicks on d' with queries containing $\langle a, b\rangle$.

$$L_1 : \quad P(d'|w)^h = P(d'|w)^\alpha = P(d'|w)^\beta \qquad (3)$$

where $w = a, b, c, d, e$. Note that at the single word level, *full model* and *independent model* are equivalent. Thus we have $P(d'|w)^\alpha = P(d'|w)^\beta$.

Then we arrive at the combination of probabilities in level L_1 to L_2:

$$L_2 : \quad P(d'|a, b)^h = (1 - \lambda)P(d'|a, b)^\beta + \lambda P(d'|a, b)^\alpha \quad (4)$$

$$L_2 : \quad P(d'|c, d)^h = (1 - \lambda)P(d'|c, d)^\beta + \lambda P(d'|c, d)^\alpha \quad (5)$$

where λ is the shrinkage rate that we set according to our confidence with the *full model* in this case. Note that λ can vary for every case and is tunable according to observation supports of long units.

Similarly, the estimation $P(d'|c, d, e)^h$ at L_3 is expressed as:

$$L_3 : P(d'|c, d, e)^h = (1 - \lambda)P(d'|c, d, e)^\beta + \lambda P(d'|c, d, e)^\alpha \quad (6)$$

where the *independent model* estimation of $P(d'|c, d, e)^\beta$ is the combination of $P(d'|c, d)^h$ and $P(d'|e)^h$ by setting:

$$P(d'|c, d, e)^\beta = \frac{P(d')P(c, d|d')^h P(e|d')^h}{\sum_{d'} P(d')P(c, d|d')^h P(e|d')^h}. \qquad (7)$$

Finally, the estimation $P(d'|a, b, c, d, e)^h$ becomes:

$$L_4 : \qquad P(d'|a, b, c, d, e)^h = (1 - \lambda)P(d'|a, b, c, d, e)^\beta + \lambda P(d'|a, b, c, d, e)^\alpha \quad (8)$$

where, again, $P(d'|a, b, c, d, e)^\beta$ is estimated using *independent model* on $P(d'|a, b)^h$ and $P(d'|c, d, e)^h$.

In general, for a query q, and document d', the CPH $P(d'|q)^h$ is:

$$P(d'|q)^h = (1 - \lambda)P(d'|q)^\beta + \lambda P(d'|q)^\alpha. \qquad (9)$$

[1] The *n-gram* (or *unit*) in our case is referred to as a word sequence of length n.

Then $P(d'|q)^\beta$ is estimated using:

$$P(d'|q)^\beta = \frac{P(d')P(q_l|d')^h P(q_r|d')^h}{\sum_{d'} P(d')P(q_l|d')^h P(q_r|d')^h}. \tag{10}$$

where $q_l \oplus q_r = q$, i.e. q_l and q_r are a split of q, $P(q_l|d')^h$ is obtained using $P(d', q_l)^h / P(d')$, $P(d'|q)^\alpha$ is the *full model* estimation. The structure of the tree reflects a particular recursive parsing of the queries into smaller word combinations and ultimately single words. With the structure of the tree fixed, the probability $P(d|q)^h$ can be computed recursively by a bottom-up procedure. Eq. 9 and Eq. 10 illustrate the recursion. We refer to the tree structure, exemplified in Fig. 2, as the *conditional probability hierarchy (CPH)*.

The construction of the tree has much to do with the overall performance. Note that we only use 2-way tree here. A greedy algorithm is used to produce the hierarchy. In particular, we iteratively search for the binary adjacent units (can be a word) with largest support in each level and feed the merged units to the higher level.

Now we need to determine the parameter λ in Eq. 9. Intuitively, λ should depend on the number of instances that $w_1, ..., w_{k'}$ appear together. It is natural to give higher weight to the *full model* when there are many such observations since the *full model* tends to be more accurate. Accordingly, we weight λ as: $\lambda = \frac{n(w_1,...,w_{k'})}{\alpha + n(w_1,...,w_{k'})}$, where $\alpha > 0$. A large α indicates a higher trust in previous estimations and a slower update rate.

6 Bayesian estimation

So far, we have seen that both models depend on obtaining estimation for the probability $P(d|q)$ from the observations of $\langle d, q \rangle$ or $\langle d, w \rangle$. In the estimation for the *independent model*, the estimation of $P(w_i|d)$ is required but boils down to the estimation of $P(d|w_i)$ using Bayesian formula. For brevity, we only focus on deriving the estimation for $P(d|w)$. We apply Bayesian estimation due to the lack of sufficient observations. Let $P(d|w) = \theta \sim \mathcal{B}(\theta)$. We need to estimate θ. Let n be the number of times that w has been clicked on with any document, x be the number of clicks on d.

We assume users carry out queries and clicks independently. Then $P(x|\theta)$ is a binomial distribution. If we use the beta distribution $Beta(\alpha, \beta)$ as the conjugate prior for θ, we will easily see that $P(\theta|x)$ also follows the beta distribution and the beta distribution is parametrized as:

$$P(\theta|x) \sim Beta(\alpha + x, n + \beta - x) \tag{11}$$

Now $P(\theta|x) \sim Beta(\alpha + x, n + \beta - x)$, we obtain the estimation of θ, conditioned on x, as the expectation of $P(\theta|x)$:

$$\hat{\theta} \ = \int_0^1 P(\theta|x)\theta d\theta = \ \frac{\alpha + x}{\alpha + \beta + n}. \tag{12}$$

The estimation of $\hat{\theta}$ in Eq. 12 will serve as our estimation for $P(d|w)$ in the problem. Eq. 12 gets around the sparsity issue in training data and is capable to provide non-zero value for $P(d|w)$ even with no previous observation of $\langle d, w \rangle$.

The only question left for the Bayesian estimation of $P(\theta|x)$ is the parametrization of the conjugate prior $P(\theta)$, i.e. the determination of α and β for $P(\theta)$. Assume each document is equally likely to be clicked on[2]. We want to set expectation $E(P(\theta))$ as $1/m$, where m is the number of distinct documents in the whole collection. Since we have $E(Beta(\alpha, \beta)) = \frac{\alpha}{(\alpha+\beta)}$, we set $\frac{\alpha}{(\alpha+\beta)} = \frac{1}{m}$, obtaining $\alpha = \frac{\beta}{m-1}$. The Bayesian estimation for $P(d|w)$ becomes:

$$\hat{\theta} = \widehat{P(d|w)} = \frac{\frac{\beta}{m-1} + x}{\frac{\beta}{m-1} + \beta + n} \tag{13}$$

where, again, x is the number of clicks on d and m represents the number of candidate documents. n is the number of times that w has been clicked on with any document. We need $\alpha, \beta > 0$. In experiments, we tune β.

7 Experiments

For evaluation, we study the property of our approach from three perspectives: (a) prediction accuracy; (b) query segmentation quality and (c) prediction power, i.e. predictability. The accuracy in prediction is evaluated using both MLE and Bayesian estimation. The query segmentation quality examines the semantic structure in queries.

7.1 Data preparation

We apply our click model to the search environment in CiteSeer (citeseer.ist.psu.edu), a popular online search engine and digital library. We collect the Apache access logs at one CiteSeer server over 90 days period. There are in total 56,452,022 requests in this period.

We remove the robots by their *agent* field in Apache logs and time constratints. We further identify the queries performed at CiteSeer obtaining a total of 886,957 distinct queries and 1,826,817 query-click pairs. There are in all 510,409 distinct documents ever shown in search results, 301,052 of which have been clicked on. For each query and the document being clicked, we collect the first 20 documents from which this document was picked.

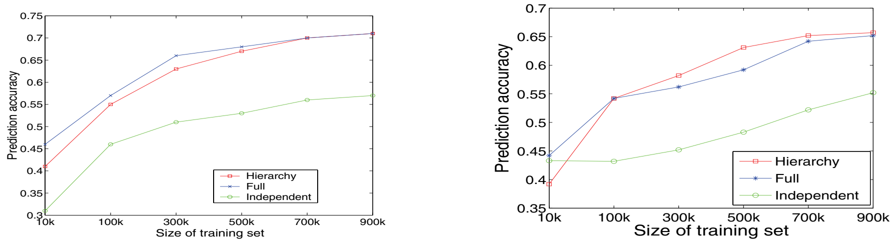
7.2 Evaluation metrics

Two important quantitative metrics for evaluation are (a) prediction accuracy and (b) predictability.

We define a prediction accuracy in evaluation as proportional to number of "correct" predictions of clicked documents from the candidate list. Formally, we have prediction accuracy defined as: $\rho_1 = \frac{n_c}{n_s}$, where n_c is the number of correct prediction and n_s is the size of tested sample queries. For each query, the original returned list of documents are provided as candidates.

We define the predictability metric as the measurement of the models' robustness to new queries. Consider when the model estimates the $P(d|q)$ as 0 for all candidate d's, the failure of prediction happens. We denote this percentage as P_f. Quantitatively, the predictability of a model equals to $1 - P_f$.

[2] We may change the assumption to take into consideration the ranking of documents in the result list.



(a) Maximum likelihood estimation.

(b) Bayesian estimation.

Figure 3: Prediction accuracy w.r.t. training size.

7.3 Prediction accuracy

We train the independent model, full model, and the CPH model over different sized subsets of the collection of *query-click pairs*. For each round of testing, we randomly choose 1,000 query-clicks complementary set of training. In each test, we evaluate the accuracy using the two metrics defined above. Since we will study the predictability later, the accuracy we show here is for predictable *query-click pairs*.

Fig. 3(a) and Fig. 3(b) give the experimental evaluation on the accuracy for our three models w.r.t. training size. Subsets of the whole collection with sizes from $10K$ to $900K$ *query-document pair* instances are experimented. Fig. 3(a) presents the accuracy of prediction using MLE for $P(d|w)$. Comparatively, in Fig. 3(b), we present the accuracy comparison using Bayesian estimation measurement. In both figures, the shrinkage rate λ for *CPH model* is set to 0.6 so that we give *full model* higher weight in combination. For Bayesian estimation prediction, the β is set to 5.

We are able to see that the *full model* usually outperforms *independent model* in terms of prediction accuracy, usually by 15%. MLE works better in prediction than Bayesian estimation but the MLE leads to lower predictability (we will discuss the predictability in Sec. 7.5.). The performance of the new CPH technique is slightly lower than *full model* in MLE but better than *full model* in Bayesian estimation. The CPH technique gains significantly higher accuracy in prediction than the *independent model*.

In Fig. 4, we show the impacts of the setting of shrinkage rate λ on the accuracy and predictability. We use the training set sized $500K$. As expected from Eq. 9, the accuracy tilts up as λ increases and the predictability goes down.

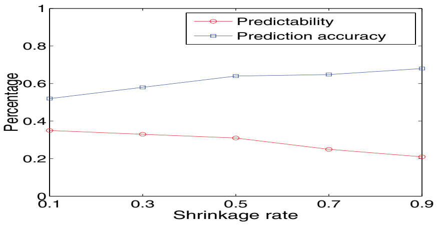


Figure 4: Prediction accuracy and power w.r.t. shrinkage rate.

The sensitivity of β for Bayesian estimation of three models are also tested. In Table. 1, we present the accuracy of CPH model w.r.t. setting of β for priors in hierarchy. As can be seen, the accuracy remains relatively stable but the smaller β gives higher accuracy.

Table 1: Prediction accuracy w.r.t. β setting.

β	Full model	Independent model	Hierarchy
1	0.56	0.47	0.61
5	0.57	0.47	0.62
10	0.58	0.46	0.61
100	0.57	0.44	0.59

Table 2: Hierarchies discovered in sample queries.

1	[tutorial, [target, tracking]]
2	[[machine, learning], [search, engine]]
3	[partial, [[least, square], regression]]
4	[[k, means], [cluster, analysis]]
5	[[markov, chain], [monte, carlo]]
6	[[spectral, clustering], jordan]
7	[[energy, efficient], [matrix, multiplication], fpgas]
8	[distributed, [cache, architecture]]
9	[[[code, red], worm], paper]
10	[[dynamic, [channel, allocation]], [cellular, telephone]]

7.4 Query segmentation

In this section, we present the quality of query segmentation formed in discovered query hierarchies. A nice segmentation of queries detects the *n-grams* in queries that provides the basis for the *full model*. Due to the limit of space, we present 10 randomly picked queries issued to CiteSeer and their segmentations performed in the CPH.

The discovered hierarchies in queries are presented by nesting square brackets in Table 2. We are able to see, from the limited sample, that the hidden structure of words in plain text queries are well discovered using the *n-gram* frequency. With proper segmentations, discovered *n-grams* in queries are feed to *full models*, and the hierarchical structures are followed while evaluating Eq. 9 for *probabilistic hierarchy*. As we have seen in Sec. 7.3, the use of *n-grams* for *full model* boosts prediction accuracy of *independent model*. We will see in Sec. 7.5 that the *probabilistic hierarchy* improves in predictability from *full model* as well.

Table 3: Predictability of CPH model w.r.t. shrinkage rate.

$S \setminus \lambda$	0.1	0.3	0.5	0.7	0.9
100k	0.20	0.19	0.18	0.18	0.16
300k	0.28	0.27	0.27	0.26	0.24
500k	0.31	0.30	0.29	0.29	0.26

7.5 Predictability

The definition of predictability is given in §. 7.2. The predictability of three models are compared. Fig. 5 compares the predictability of three models w.r.t. the size of training set. Generally the predictability increases as the training size grows. In particular, the *full model* has the worst predictability and the *independent model* has the best. The CPH model is between the two.

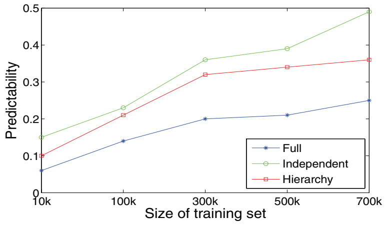


Figure 5: Predictability w.r.t. training size.

We also present predictability of CPH model w.r.t. the shrinkage rate in Table 3. S is the training size. Note the predictability drops slightly as λ grows.

One might expect an overall evaluation of the three models combining prediction accuracy and predictability. We sum up the comparisons among three models in Fig. 6. In Fig. 6, we plot the product of accuracy and predictability w.r.t. the training size. We are able to see that the CPH outperforms both models overall.

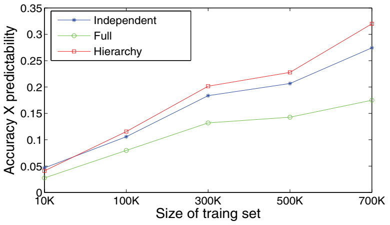


Figure 6: Combined evaluation of model performances w.r.t. training size.

7.6 Computation complexity

Finally, training time for the *independent model* and *full model* are compared. Analytically, complexity of training for *full model* should be the training time of *independent model* times the complexity of breaking queries. Let the training size be N and the average length of queries be L. The complexity for *independent model* training is $O(LN)$. Consider the maximum length of units for *full model* is set as K. The complexity for *full model* training is therefore bounded by $O(L^K N)$. Note that since normally, K and L are small integers, training for both models only requires linear time. Our experiment shows that the CPU time of training for *full model* is 3-4 times of that for *independent model*.

8 Conclusions and Future work

In this paper, we address the click prediction in the Web search scenario and explore the sparsity of problem space that often exists for machine learning methods for information retrieval research. Our experiments on the CiteSeer data show that large scale evaluation gives promising results for our three models in terms of prediction accuracy and predictability.

For future work, we will propose more sophisticated formulation methods for the conditional probability hierarchy (CPH). It would also be useful to improve our methods by considering and modeling the intent of users.

References

[Banerjee and Ghosh, 2001] A. Banerjee and J. Ghosh. Clickstream clustering using weighted longest common subsequences. In *In Proceedings of the Web Mining Workshop at the 1 st SIAM Conference on Data Mining*, 2001.

[Beeferman and Berger, 2000] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.

[Deshpande and Karypis, 2004] Mukund Deshpande and George Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Inter. Tech.*, 4(2):163–184, 2004.

[Gunduz and Ozsu, 2003] Sule Gunduz and M. Tamer Ozsu. A web page prediction model based on click-stream tree representation of user behavior. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[Halvey *et al.*, 2005] Martin Halvey, Mark T. Keane, and Barry Smyth. Predicting navigation patterns on the mobile-internet using time of the week. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 958–959, 2005.

[Joachims, 2002] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[Shen *et al.*, 2005] Xuehua Shen, Susan Dumais, and Eric Horvitz. Analysis of topic dynamics in web search. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1102–1103, 2005.

[Srivastava *et al.*, 2000] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, 2000.