Priors in Web Search*

Michael Bendersky
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
bemike@cs.umass.edu

Kenneth W. Church
Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
kenneth.church@jhu.edu

ABSTRACT

Web search combines information obtained at query time with prior knowledge to form a posterior. This paper focuses on the prior, which we believe is interesting, given the poverty of the query stimulus (many of the web queries are no more than a word or two).

We propose a learning framework based on the Noisy Channel Model for combining prior evidence from multiple sources including both the authors' perspectives (e.g., PageRank - the principal eigenvector of the web graph) as well as the readers' perspectives (e.g., click logs and toolbar activity). The framework is general enough that it can be applied to both documents and queries, both of which have strong priors.

We show that even features that appear to depend on the combination of queries and documents and are often used for learning a ranking function (such as relevance judgments or retrieval scores) can be included in the prior model using multiple mechanisms of aggregation (e.g., moments or entropy). More is more. The prior model improves with both more features and more aggregates.

We conduct an empirical evaluation of the proposed framework, demonstrating its benefits over a diverse set of learning tasks including: (1) query difficulty estimation, (2) click types prediction and (3) document ranking.

1. INTRODUCTION

Web search combines information obtained at query time with prior knowledge to form a posterior. In a standard *ad hoc document retrieval* setting [26], a uniform distribution over both the documents in the collection and the possible user queries is usually assumed for convenience. However, in the context of web search, the priors are highly non-uniform in important ways.

In fact, given the poverty of the stimulus (many web search queries are just a word or two long), there are not enough bits in the query to address all the web pages, if document prior had a uniform distribution. According to Mei and Church [27], queries have about 22 bits of entropy, and can only address $2^{22} \approx 4M$ pages, if all pages are equally likely. However, some pages are much more likely than others to be the answer to the next question. Modern search engines are able to search over tens of billions of pages[1], even though the queries are too short to address such a large address space, and they depend on the prior to make up for this difference. This fact is confirmed by the wide adoption of PageRank [4] and other document priors [19, 11, 25] in web search. PageRank's success led to the investigation of other types of *document priors*, based, for instance, on document centrality in a link graph [19, 11], document popularity [25] or both [30].

A distribution of user needs in web search is also far from being uniform. Queries vary significantly by difficulty and intent. The observed user behavior in response to the query, however, is very scarce. In most cases, user performs only one or two clicks on the first page of results, if at all, making it hard to distinguish between easy and hard queries and queries with different intents.

Accordingly, to improve the retrieval performance and evaluation, and to better model the user experience in the context of web search, an accurate and robust estimate of prior knowledge about both the documents and the queries is important. Thus far, however, this prior knowledge was studied in separate contexts, e.g. applied to document ranking [32, 3, 11, 20, 28], or to query performance prediction [12, 8, 23]. Our aim is to bring together these research threads in the context of web search.

We consider multiple features (e.g., clicks, statics ranks, toolbar counts and retrieval scores) that are often available in the context of web search, as well as multiple aggregates (e.g., moments or entropy) and find that performance improves with both (over a range of different tasks).

More is more. More aggregates are better than fewer aggregates, and more features are better than fewer. Some pieces of this pattern are more obvious than others. It is not surprising that performance improves with more features. Learning systems in general and, specifically, learning-to-rank systems are quite effective in taking advantage of multiple features [24]. In this paper we demonstrate that similar mechanisms can be used to take advantage of multiple aggregates for learning better priors, an approach which is not used as much in practice, to the best of our knowledge.

This *feature aggregation* approach allows us to develop a learning framework for estimating prior knowledge about either a given document or a given query. We base our framework on a well known noisy channel model, and empirically explore its utility for web search. Our framework is general enough to accommodate prior evidence from multiple sources including both the authors' perspectives (e.g., PageRank) as well as the readers' perspectives (e.g., click logs and toolbar activity). It is often assumed that we cannot use pairwise features (features that depend on both queries

*Work done while both authors were at Microsoft Research.

[1]For instance, the homepage of the web search engine Cuil (http://www.cuil.com/) states that it indexes almost 125 billion pages.

SIGIR'09 LR4IR workshop, July 23, 2009, Boston, MA.

and documents) in a prior. The feature aggregation approach makes it clear how to take advantage of such features for estimating a better prior.

The flexibility of our framework allows us to develop several novel practical models. In addition to being novel contributions by themselves, these models confirm the validity of our general framework and our *more is more* approach for prior prediction over a wide range of different tasks.

First, in Section 3.2 we propose a model for query difficulty prediction, which generalizes and improves over the previous approaches. Second, in Section 3.3 we develop a novel technique for estimating query malleability and commercial intent which augments click data with non-click features. Finally, in Section 4.2 we show that click aggregates can be used as priors (rather than posteriors, as is usually done with clicks [17]) for learning a better ranking function.

There is some controversy over clicks among our colleagues. Some people love them, and some don't. We find merit with both positions. The combination of clicks with other (non-click) features such as PageRank or retrieval scores is better than either feature by itself, but if one had to choose a single feature, clicks may not be the single best choice for all cases.

The best combination of features/aggregates in Section 3.2 shows performance that is perhaps as good as that of the human judges. Clicks are often useful when we have them, but we don't always have them (especially for rare queries in the "long tail"). It is helpful to fall back on other features (as we show in Section 3.3), as well as to aggregate over other queries (as we show in Section 4.2).

The remainder of the paper is organized as follows. The prior estimation framework and the data sources are detailed in Section 2. Sections 3 and 4 discuss and evaluate the applications of query and document priors, respectively. Section 5 outlines the related work. Finally, the conclusions are discussed in Section 6.

2. PRIOR ESTIMATION FRAMEWORK

2.1 Motivation: The Noisy Channel Model

Shannon's noisy channel model has been applied to a wide range of natural language processing tasks, following success in speech recognition [16]. It decodes the most likely unobserved input $\hat{I}$ from an observed output O

$$\hat{I} = \underset{I}{\operatorname{argmax}}\, p(I|O) = \underset{I}{\operatorname{argmax}}\, p(O|I)p(I). \qquad (1)$$

The noisy channel makes an explicit distinction between the *channel model*, $p(O|I)$, and the *prior* (or language model), $p(I)$. This distinction led to a sizeable literature on language modeling [31], with the promise that advances in language modeling can be generalized across a wide range of applications. Language modeling makes a difference when $p(I)$ is far from uniform. Language modeling is particularly important for long tailed distributions, which are common in web search. Some applications would not be possible without language modeling, such as applications with a lot of noise, underspecified inputs or relatively weak channel constraints.

Web search, for instance, would not be possible if all documents were equally likely. In web search a "perfect" relevant document can be viewed as an unobserved input, and a user query as an observed noisy output. Documents in the collection can then be ranked by their probability of "generating" the user query [29].

The document prior, $p(I)$, is very informative, since the observed short keyword query (often, just one or two words) does not have enough bits to address the entire document space (tens of billions of pages). Web search works by taking advantage of the strong priors. Some documents are much more likely than other documents.

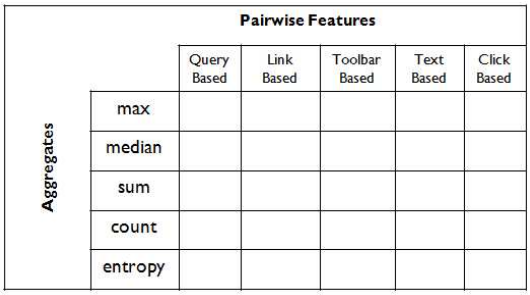
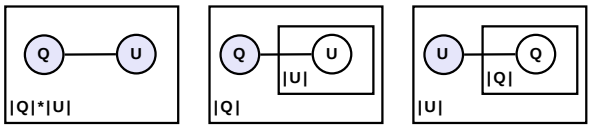


Figure 1: Feature/aggregate matrix.



(a) pairwise features (b) query priors (c) document priors

Figure 2: Graphical model of prior estimation.

Another example is a query performance prediction task, a well known problem in IR [12]. Here, the search engine attempts to automatically predict the quality of the ranked list of documents retrieved in response to a query. The input I is the actual retrieval performance of a query (as measured by some precision/recall metric). I is not directly observed, since the search engine does not have an a priori knowledge of the relevant documents for each possible query. Instead, the user clicks on the documents in the ranked list are the observed noisy output O. Since the observed click data is often sparse, a strong query difficulty prior, $p(I)$, is vital in order to reliably evaluate query performance.

Given the importance of priors in web search, how can they be estimated? In the next section we present a novel feature aggregation method to estimate priors by bootstrapping an existing large scale data repositories often used for learning ranking functions.

2.2 Feature Aggregation

Assume we are given a joint distribution $p(q, u)$ over queries and documents. Both *query and document* priors can be estimated from this joint distribution. Given a query q and a collection of documents $\mathcal{U}$, we can express the query prior as $p(q) = \sum_{u \in \mathcal{U}} p(q, u)$. Conversely, given a document u and a collection of queries $\mathcal{Q}$, we can express the document prior as $p(u) = \sum_{q \in \mathcal{Q}} p(q, u)$.

In this work we consider two data sources for estimating $p(q, u)$. The first source stems from a large body of recent work on "Learning To Rank" [24] techniques. This work led to the development of *pairwise features* that take into account both the document and the query. The second source is a search log that holds records of click pairs (q,u), where q is the issued query and u is some document clicked in response to q. Instead of using these pairwise features directly, as is usually done (e.g., to learn a ranking function), we use their *aggregates* to learn either a query or a document prior.

For instance, the marginalization process presented above can be viewed as applying a summation aggregate to the pairwise features. However, resorting to a single aggregate function might limit the quality of the resulting features. A well known *tfidf* weighting scheme in IR is a good example of how a combination of two aggregates performs better than either of them on its own.

Source	Features	Label
R_{qu}		
- Query-Based	#terms(q)	
- Link-Based	PageRank(u), PageRank(D)	
- Toolbar-Based	#visits (u), #visits (D), #uniq_users (u), #uniq_users (D)	l_{qu}
- Text-Based	BM25F(q,u), tf_title(q,u) tf_URL (q,u), tf_URL (q,D)	
L_{qu}	#clicks(q,u), click_pos(q,u)	t_{qu}

Table 1: Summary of data in feature repository R_{qu} and click repository L_{qu}. Features are categorized by type, and are based either on the query-document pair (q,u), query (q), document (u), or document web-domain (D).

Church and Gale [9] show that introducing an additional aggregate, burstiness, helps to even better distinguish content-bearing words in the collection. Following this line of thought, we choose to work in a discriminative machine learning framework, and cast different *feature aggregates* as different *features*, letting a supervised model to decide on the importance of each feature.

Fig. 1 illustrates the feature aggregation process. Each cell in the matrix at Fig. 1 is a single feature in the feature aggregate vector. Note that in this fashion, we can potentially improve the performance of our model by either adding new features or new feature aggregates. As the experimental results in Sections 3 and 4 demonstrate, *more is more*: both new features and new aggregates consistently help to improve the prior estimation.

Fig. 2 depicts a graphical model of our priors. Given a set of $|\mathcal{Q}| * |\mathcal{U}|$ observed pairwise features (a), we can either derive a set of $|\mathcal{Q}|$ query priors (b), or a set of $|\mathcal{U}|$ document priors (c). Feature aggregate vector (Fig. 1) can then serve as an input to a regression model for learning the priors, an approach we adopt in this paper.

Note that although the graphical models in Fig. 2(b),(c) are fairly straightforward, they can be extended to more complex models, e.g., by considering inter-query or inter-document relations. In this case, instead of linear/multinomial regression-based models employed in this work, relational learning methods such as CRF [21] can be used for estimating priors. We leave the exploration of these methods to future work.

Theoretically, all the possible query-document pairs need to be considered for the correct computation of the prior. In practice, for most such pairs the joint probability $p(q,u)$ is negligible, allowing us to consider only a subset of related query-document pairs. We assume that pairs can be related either through clicks, retrieval process or explicit human judgments. In the following section we provide more details on the construction of our principal data sources for determining these pairs.

2.3 Data Sources

Two main data sources are used throughout this paper: a *feature repository* obtained from a large web corpus and a *click repository* obtained from a query log of a commercial search engine. Table 1 presents a summary of these two sources. In what follows, we give details on the process of obtaining and analyzing the data.

2.3.1 Feature Repository

We use a large-scale feature repository, containing pairwise query-document features, of the form typically used for training "Learning to Rank" [24] algorithms. Formally, such a repository, denoted R_{qu} is a set of 4-tuples $\{(q,u,\mathbf{f}^{\mathsf{R}}_{qu},l_{qu})\}$. Each tuple is uniquely identified by a query string q and a URL[2] u pair. R_{qu} contains approximately 4 million such tuples for 31,962 unique queries. $\mathbf{f}^{\mathsf{R}}_{qu}$ is a feature vector for (q,u) pair, and l_{qu} is a human rating for the relevance of document u w.r.t. q. In our experiments, $\mathbf{f}^{\mathsf{R}}_{qu}$ contains 11 features describing textual query-document match, document quality and popularity. $l_{qu} \in [1,5]$ is an ordinal value, where the range corresponds to the relevance ratings [*1-Perfect, 2-Excellent, 3-Good, 4-Fair, 5-Bad*].

[2]We use the terms "document" and "URL" interchangeably in this paper, assuming that a document is a web page represented by a single unique URL.

	$G_{u\in\mathbf{u}}(f_{qu})$
LOG_SUM	$\log(\sum_{u\in\mathbf{u}} f_{qu}+1)$
LOG_COUNT	$\log(\lvert\mathbf{u}\rvert+1)$
MEDIAN	$\text{median}_{u\in\mathbf{u}} f_{qu}$
MAX	$\max_{u\in\mathbf{u}} f_{qu}$
ENTROPY	$-\sum_{u\in\mathbf{u}} \frac{f_{qu}}{\sum_{u\in\mathbf{u}} f_{qu}} \log \frac{f_{qu}}{\sum_{u\in\mathbf{u}} f_{qu}}$

Table 2: List of aggregate functions for query priors.

2.3.2 Click Repository

In addition to the feature repository, we use a click repository, denoted L_{qu}. L_{qu} is a summary of the portion of the query log from a commercial web search engine, containing click information about queries and documents represented in R_{qu} for the period of February-May, 2008. L_{qu} is based on approximately 11 million clicks.

Similarly to the feature repository R_{qu}, click repository L_{qu} consists of the four-tuples of the form $\{(q,u,\mathbf{f}^{\mathsf{L}}_{qu},t_{qu})\}$, which are uniquely identified by a (q,u) pair. $\mathbf{f}^{\mathsf{L}}_{qu}$ holds information about the number of clicks and the click position for (q,u). $t_{qu} \in \{alg, ad, sug\}$ is a click type for the clicks on the pair (q,u). alg is a click on an algorithmic search result, ad is a click on a contextual ad and sug is a click on query suggestion.

Since both the feature repository and the click repository contain data for query-document pairs, we use the concatenation of the two feature vectors $\mathbf{f}^{\mathsf{R}}_{qu}$ and $\mathbf{f}^{\mathsf{L}}_{qu}$ as the complete feature vector $\mathbf{f}_{qu}$, containing information from both sources.

3. QUERY PRIORS

3.1 Query Prior Definition

The pairwise features described in Section 2.3 are convenient for certain purposes, e.g., learning the correct rankings of documents in response to a query [24]. However, they are less suitable for estimating a query prior. Instead, we are interested in a query-dependent feature repository R_q consisting of set of tuples $\{(q,\mathbf{f}_q)\}$.

For this purpose, we use the marginalization property discussed in Section 2.2 to leverage, by aggregation, the available pairwise features. Accordingly, each query q is represented in R_q by a feature vector $\mathbf{f}_q$ of size $|a| * |\mathbf{f}_{qu}|^q$, where $|a|$ is the number of aggregates used in the model, and $|\mathbf{f}_{qu}|^q$ is the number of available pairwise features for q. Each feature $f_q \in \mathbf{f}_q$ is of the form

$$f_q = G_{u\in\mathbf{u}}(f_{qu}), \tag{2}$$

where $\mathbf{u}$ represents the set of documents for which pairwise features with query q are available, and $G_{u\in\mathbf{u}}(\cdot)$ is an aggregate function over $\mathbf{u}$.

Table 2 presents 5 aggregate functions that were used throughout our experiments[3]. Note that aggregate functions can be com-

[3]We experimented with additional aggregate functions such as standard deviation, however including these aggregates did not improve the perfor-

Rating	l_{qu}	l_q
1-Perfect	20,579	7,161
2-Excellent	57,204	9,486
3-Good	423,540	9,346
4-Fair	1,157,281	3,695
5-Bad	2,198,598	2,274

Table 3: Distribution of pairwise query-document (l_{qu}) ratings and query difficulty (l_q) ratings.

bined. For instance, log of the mean aggregate can be expressed as LOG_MEAN = LOG_SUM − LOG_COUNT.

The remainder of this section presents two applications of aggregate features for estimating query priors. Section 3.2 describes their use for query difficulty prediction. Section 3.3 describes their use for predicting click types.

3.2 Query Difficulty Prediction

Query performance prediction is an active research topic in traditional IR [8, 12, 13, 15]. Accurate prediction of query performance is, clearly, an important task for web search engines as well. For example, if the search engine estimates with high probability that a query has a single perfect answer, the user experience can be improved by directly serving the relevant page, rather than displaying the entire result list[4] [1]. On the other extreme of the query performance spectrum, if the search engine estimates with high probability that a query will not retrieve relevant pages, query-suggestion mechanisms can be activated to propose alternative queries, for which the search engine has better answers [34].

In this paper, we cast the task of query performance prediction as an estimation of a *query difficulty prior*. Since we view the query difficulty as an inherent property of the query itself, and do not base it on the ranking performance of a particular search engine, as is usually done in previous work (e.g., [12]), we define it as an upper bound on the retrieval performance given the query.

Formally, letting l_{qu} be a human rating assigned to a particular query-document pair, we define the estimated query difficulty by

$$l_q = \min_{u \in \mathbf{u}} l_{qu},$$

where $\mathbf{u}$ represents the set of documents for which human ratings w.r.t. query q are available. Higher l_q indicates more difficult query. In other words, queries that have at least one "Perfect" rating w.r.t. some document are the easiest, while those that have only "Bad" ratings are the hardest.

Table 3 shows the distribution of pairwise ratings (l_{qu}) and query difficulty ratings (l_q). Learning the pairwise ratings l_{qu} is the focus of the ongoing work on learning to rank, and, although there is much recent progress in the field, it still remains a hard problem. One of the confounding factors in the process of learning the correct pairwise ratings l_{qu}, is the fact that l_{qu} is dominated by low ratings (most of the documents in the collection are not a good answer for any given query).

The distribution of query difficulty rating l_q is, on the other hand, much more balanced, and (as we show in this section) can be easier to estimate. 52% of the queries have at least one document with *Perfect* or *Excellent* rating, and so more than half of the queries have at least one good answer.

However, while being a simpler learning problem than ranking all the documents w.r.t. the query, predicting an upper bound on query performance is still an important task. Reliable discovery of an upper bound on query performance can allow web search engines to automatically detect navigational [5] queries (queries that have a single "Perfect" answer), which constitute a large portion of web search traffic. On the other hand, if the model predicts with high confidence that the query is very hard (is unlikely to yield any good answers), query suggestion and reformulation mechanisms can be activated to help user satisfy her underlying information need.

All the features defined in Table 1 can be used as cues for determining the query difficulty. Previous work [12, 15] shows that queries with high textual overlap with the retrieved documents are easier. Same applies to queries that retrieve documents with high static ranks [23]. We also expect popular (many clicks and visits) queries to be easier. Teevan et al. [33] show that click entropy is useful in predicting the ambiguity of a query. Lee et al. [22] demonstrate that summaries of click and anchor-text distributions are useful for the identification of user goals in web search. Agichtein and Zheng [1] show that click statistics can be leveraged to identify "best bet" queries (queries that have a "Perfect" answer).

Thus, the query difficulty prior can be viewed as a generalization of previous approaches to query performance prediction. Using our *feature aggregation* method all of the above features can be combined into a single feature vector, and their aggregates can be computed as shown in Table 2. Indeed, as our experiments show, a combination of features and aggregates using a statistical model will attain a better performance than any of them by themselves.

3.2.1 Model Construction

To estimate query difficulty, we build a model of the form $\hat{l}_q \sim \mathbf{f}_q$, where $\mathbf{f}_q$ is derived from pairwise feature vector $\mathbf{f}_{qu}$, using Eq. 2. Multinomial logistic regression[5] is used for estimating $\hat{l}_q$. In a multinomial regression, a probability of a label l given observation $\mathbf{f}_q$ is determined by

$$p(l|\mathbf{f}_q) = \frac{\exp(\mathbf{w}_l^{\mathbf{T}} \mathbf{f}_q)}{\sum_{l' \in [1,\ldots,5]} \exp(\mathbf{w}_{l'}^{\mathbf{T}} \mathbf{f}_q)},$$

where $\mathbf{w}_{l'}$ is a vector of weights which maximizes the likelihood of the training set. For each query q in the test set, the estimated query difficulty is obtained by

$$\hat{l}_q = \operatorname*{argmax}_{l \in [1,\ldots,5]} p(l|\mathbf{f}_q).$$

3.2.2 Model Evaluation

For evaluating the query difficulty prior, we measure the agreement between the rating l_q assigned by the human assessors and the rating $\hat{l}_q$ assigned by our model. We achieve this by translating confusion matrix A (where A_{ij} is the number of times human raters assigned rating i, and the model assigned rating j) into a single summary statistic.

To this end, we use the weighted variant of Cohen's κ [10]. An important advantage of Cohen's κ over the more common *accuracy* statistic is the fact that it adjusts for chance agreement. κ value is negative when there is less observed agreement than is expected by chance, zero when observed agreement can be accounted for by chance, and approaches one when raters get closer to the complete agreement.

We use Eq. 2 to construct the features. To evaluate the query difficulty prediction performance, we perform a 3-fold cross-validation,

mance in our initial experiments.

[4]Case in point, Google's "I'm Feeling Lucky" feature.

[5]Implemented using `multinom` function in R package `nnet`. Ordinal regression and neural network were evaluated as well, but have not shown a significant improvement over the multinomial regression for the task of query difficulty prediction.

(a) Features	κ	(b) Aggregates	κ
R_{qu}	$0.549^{\dagger}$	LOG_MEAN	$0.559^{\dagger}$
L_{qu}	$0.374^{\dagger}$	+ ENTROPY	$0.575^{\dagger}$
R_{qu} + L_{qu}	**0.591**	+ MAX	0.588
		ALL	**0.591**

(a) All the aggregates are used, data sources vary.
(b) All the features are used, aggregates vary.

Table 4: Cohen's κ — Query difficulty prior estimation results. Higher values indicate higher agreement with the human raters. + denotes that the current row utilizes the features from the previous row as well. Statistical differences (paired t-test, $\alpha < 0.05$) with the method that uses *all* the features and *all* the feature aggregates (boldface) are marked with †.

	Bad	Fair	Good	Excellent	Perfect
Bad	148	38	8	5	1
Fair	48	154	48	13	5
Good	14	129	494	211	68
Excellent	3	12	227	525	189
Perfect	1	7	53	174	424

Table 5: Ratings confusion matrix. Diagonal (shaded) represents agreement between the human raters and the model.

each time using one of the folds as the test set and the rest as the training set. Table 4 reports the mean κ over the three runs.

The results in Table 4 (a) indicate that click-data alone (L_{qu}) is not sufficient to reliably predict query difficulty. However, combining click data with features in R_{qu}, which are traditionally used for query performance prediction, such as textual matches [12] and static ranks [23], results in a 7% improvement in κ over the baseline of using R_{qu} alone. Adding all the aggregates (Table 4 (b)) leads to a 5.5% improvement over using a single LOG_MEAN aggregate. These improvements are statistically significant, and consistent over all folds in the test set.

Table 5 presents the breakdown of the confusion matrix A for our best performing model that includes all the features and all the aggregates on a development set of $3,000$ queries. Note that for more than half of the queries there is a perfect agreement between the raters and the model. For 94% of the queries the ratings deviate by at most one relevance level. Only for two (out of $3,000$) queries there is a complete disagreement between the model and the raters.

We also compare the agreement of our model with a given human rater to the agreement between human raters. As we do not have multiple ratings for all the queries, we extract a subset of $2,057$ queries for which ratings from at least 7 human judges are available. Average weighted κ between all the judges pairs for these queries is 0.47, which is lower than the agreement achieved by our model. This result is in line with previous research on human evaluation in IR (see for instance Bailey et al.[2]).

We conclude that given a training set of reliable human ratings, our model learns a reasonable estimate of a query difficulty prior on a held-out set, perhaps better than one that could be produced by a single human rater. This result affirms our intuition about the utility of different features and feature aggregates and motivates further investigation of an additional type of query prior.

3.3 Query Click Types Prediction

In addition to human relevance ratings, the search log itself provides an evidence to the nature of the query: click types. Recall from Section 2.3 that query log L_{qu} contains, along with the number of clicks and click positions, an information about the click types. In this paper we focus on two click types common in web search: ad— click on a sponsored search result and sug— click on a query suggestion.

(a) Commercial intent			
Feat.	$RMSE_{ad}$	**Agg.**	$RMSE_{ad}$
R_{qu}	$1.694^{\dagger}$	LOG_MEAN	$0.753^{\dagger}$
L_{qu}	$0.757^{\dagger}$	+ ENTROPY	0.729
R_{qu} + L_{qu}	**0.724**	+ MAX	0.744
		ALL	**0.724**

(b) Malleability			
Feat.	$RMSE_{sug}$	**Agg.**	$RMSE_{sug}$
R_{qu}	$1.535^{\dagger}$	LOG_MEAN	$0.887^{\dagger}$
L_{qu}	$0.949^{\dagger}$	+ ENTROPY	**0.860**
R_{qu} + L_{qu}	$0.885^{\dagger}$	+ MAX	$0.867^{\dagger}$
		ALL	$0.885^{\dagger}$

Table 6: RMSE — prediction of number of clicks of types ad (comm. intent) and sug (malleability) when varying the features and aggregates used in the model. Lower values of RMSE indicate more accurate predictions. Statistical differences (paired t-test, $\alpha < 0.05$) with the best performing model (boldface) are marked with †.

Number of ad clicks measures the *commercial intent* of the query. If the proportion of ad clicks is expected to be high for a particular query, more page real estate may be allocated for displaying the ads. Alternatively, an ad display may be reduced/removed if little/no ad clicks are expected.

Number of sug clicks measures the *query malleability*, or how likely is the user to utilize a query suggestion proposed by the search engine to reformulate her query[6]. A high expected proportion of sug clicks for a particular query may lead to allocating more page real estate to query suggestions.

Query commercial intent and query malleability can both be cast as query priors. Accordingly, we examine how the technique described in Section 3.1 can be used to determine the expected number of these click types. As previously, we are not committed to a specific feature type or aggregate.

3.3.1 Model Construction

Most generally, we are estimating a model of the form $\hat{n}_t^q \sim \mathbf{f}_q$, where $\hat{n}_t^q$ is the estimated number of clicks of type t for query q, and $\mathbf{f}_q$ is given by Eq. 2. In our experiments, we use a neural network method as an estimation method. A single-hidden-layer neural network with 3 units in the hidden layer and a linear output is trained[7]. The advantage of the neural network over linear regression is its ability to learn a non-linear model of the features. This advantage was verified in our preliminary experiments, where neural network outperformed several methods based on a linear regression.

3.3.2 Model Evaluation

For evaluating the click type prior, we use the root mean square error (RMSE), a standard measure of the differences between the observed values (number of actual clicks, n_t^q), and the values predicted by the model (predicted number of clicks, $\hat{n}_t^q$). We calculate

[6]An alternative way to measure query malleability is through tracking queries that are often reformulated by the users themselves, however obtaining this information is not as straightforward as counting sug clicks [18].

[7]Implemented using R package `nnet`.

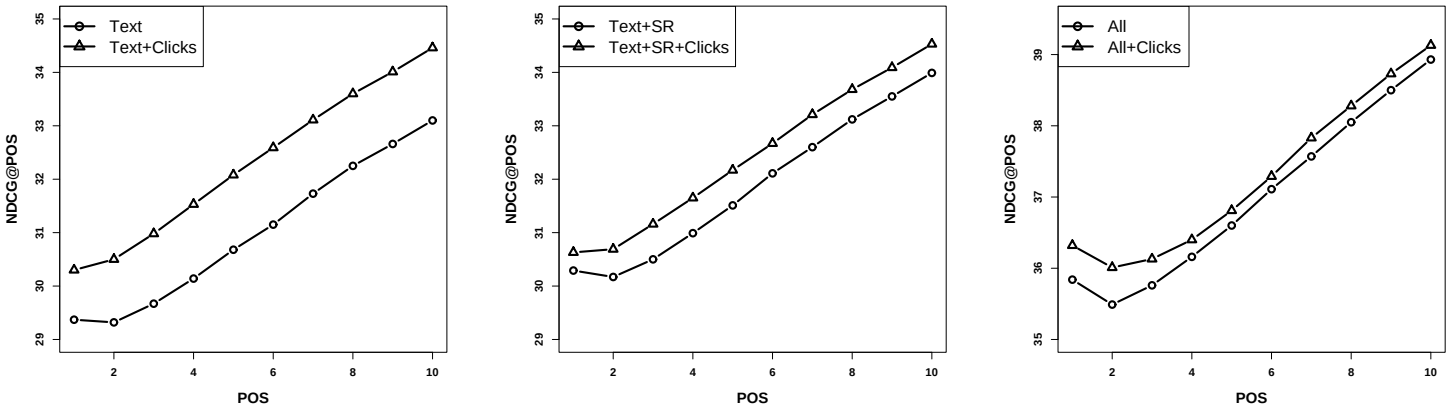


Figure 3: $NDCG@\{1-10\}$ — document ranking results. Plots compare the performance of each of the three baseline rankers to the performance of the baseline ranker combined with the click-based document prior.

RMSE for a certain type of click t as

$$RMSE_t = \sqrt{\frac{\sum_{q \in \mathcal{Q}} (\hat{n}_t^q - n_t^q)^2}{|\mathcal{Q}|}},$$

where $\mathcal{Q}$ represents the queries in the test set. Lower values of RMSE indicate better prediction performance.

We use Eq. 2 for constructing the features. To evaluate the model performance, we perform a 3-fold cross-validation, each time using one of the folds as the test set and the rest as the training set. Table 6 reports the mean RMSE over the three runs for predicting commercial intent and query malleability. Table 6 compares the performance of our model when using features from different repositories and different aggregates of these features.

Using several aggregates is always better than using LOG_MEAN alone. An addition of aggregates results in statistically significant reductions of up to 3.9% in $RMSE_{ad}$ for commercial intent, and up to 3.0% in $RMSE_{sug}$ for query malleability. Note that just as an optimal feature selection can often lead to an improved prediction in statistical learning models, same holds for aggregate selection. For instance, for query malleability, using LOG_MEAN + ENTROPY aggregates leads to an improvement of 2.8% over the baseline of using all the aggregates. Investigating techniques of optimal aggregate and feature selection for prior estimation is an interesting direction for future work.

Another interesting result in Table 6 is the contribution of non-click features (features in R_{qu}) to the performance of the model. Incorporation of features from R_{qu} in both cases leads to statistically significant reductions in $RMSE_t$ over using click data (features in L_{qu}) alone. $RMSE_{ad}$ reduces by 4.4%, while $RMSE_{sug}$ reduces by 6.7%.

These improvements are especially valuable for queries in the long tail, for which little or no click data is available, as they show potential for predicting user click behavior based on non-click features such as static ranks of retrieved documents or their retrieval scores. In fact, we found that considering only the features in R_{qu} halves the $RMSE_t$ in comparison to the baseline of using an average number of clicks of type t as an estimate for $\hat{n}_t^q$. To the best of our knowledge, improving click types prediction using non-click features is a novel result, not reported in any previous work.

4. DOCUMENT PRIORS

4.1 Document Prior Definition

As we have seen in Section 2.2, document priors are symmetrical to query priors: both are obtained by marginalization over the pairwise query-document features. Hence, for a task of document prior estimation, we need to obtain a document-dependent repository R_u, consisting of set of tuples $\{(u, \mathbf{f}_u)\}$ from pairwise features $\mathbf{f}_{qu}$. We can use a similar aggregation method to that presented in Section 3.1, this time aggregating over queries, rather than documents.

Each document u is represented in R_u by a feature vector $\mathbf{f}_u$ of size $|a| * |\mathbf{f}_{qu}|^u$, where $|a|$ is the number of aggregates used in the model, and $|\mathbf{f}_{qu}|^u$ is the number of available pairwise features for u. Each feature $f_u \in \mathbf{f}_u$ is of the form

$$f_u = G_{q \in \mathbf{q}}(f_{qu}), \tag{3}$$

where $\mathbf{q}$ represents the set of queries for which pairwise features with document u are available, and $G_{q \in \mathbf{q}}(\cdot)$ is one of the aggregate functions defined in Table 2 (being defined over queries, rather than documents).

4.2 Ranking with Document Priors

The utility of document priors for document ranking is not new. Previous research indicates that using both textual [3, 32], link-based [11, 28] and toolbar-based [25] document priors leads to significant improvements in the retrieval performance. As our document prior model allows for incorporation of pairwise features (that depend on both the query and the document) through aggregation, it allows us to evaluate a novel *click-based* document prior, not considered in the previous work. This prior consists of the click aggregates described in Section 4.1.

Note that our click-based document prior differs from previous applications of click data for ranking optimization (e.g., [17]). While previous work only exploits clicks for a given query and a given document, our click-based document prior incorporates information about clicks from *other queries* as well. Recall that for each document u our prior considers aggregates of number of clicks and their positions for all the queries for which clicks on u were observed. This allows bootstrapping information from the entire

query log, even when ranking documents in response to queries that have very sparse click data, or no click data at all.

We use RankNet [7] for document ranking, as this method provides a flexible framework for incorporating multiple features into the ranking function. To demonstrate the contribution of aggregated clicks, we start with three baseline rankers that do not include the information on the click data.

First baseline ranker, `Text`, uses only the four text-based features defined in Table 1. Second baseline ranker, `Text+SR`, uses the text-based features and the link-based features defined in Table 1. Third baseline ranker, `All`, uses the entire available feature set used for training the RankNet [7] (set which includes all the features in R_{qu}).

For each of these baseline rankers we add a click-based prior, which is comprised of the five aggregates of the two features in L_{qu}, namely the number of clicks and their positions (see Table 1). Fig. 3 compares the performance of each baseline ranker to that of the baseline ranker combined with the click-based prior. We measure performance in terms of average NDCG@k (normalized discounted cumulative gain at rank k). NDCG@k is often used in web search [7], where accuracy at the top of the ranked list is important, and there are multiple levels of relevance. NDCG@k for query q is computed as

$$NDCG@k_q = Z_q \sum_{i=1}^{k} \frac{2^{5-l^i_{qu}} - 1}{\log(1+i)},$$

where Z_q is a normalizing constant and l^i_{qu} is the relevance rating of a document at i'th rank.

As we can see from Fig. 3, the gain for ranker `Text` is the largest, reaching the maximum of 4.6% for $NDCG@\{4-6\}$. The average gain for ranker `Text` is 4.2%. The gains for rankers `Text+SR` and `All` are smaller, as these rankers include some document prior information through link-based and toolbar-based features. The average gains for `Text+SR` and `All`, when click-based prior information is added, are 1.8% and 0.8%, respectively. These gains are consistent over all the positions, as Fig. 3 demonstrates. Although the gains are quantitatively small, all of them are statistically significant, due to the large size of the query set.

5. RELATED WORK

Our model of prior knowledge in web search is inspired by several active areas of research. These areas were mostly explored independently, however, as we show, they can all be expressed within our framework as prior estimations.

Section 3.2 draws upon a long line of research on query performance prediction [8, 12, 13, 15, 23]. Most research in this area concentrated on textual features of a single retrieved set of documents. We extend this approach, and show that combining textual features with visitation counts, link graph structure and click data aggregates further improves prediction of query difficulty.

Sponsored search can be framed as an ad hoc document retrieval, where ads are retrieved in response to a query. Recent work has investigated the impact of the posterior (the quality of the retrieved set of ads) for learning when (not) to advertise [6]. In Section 3.3 we take a complementary approach, which estimates the prior of observing ad clicks given a query, regardless of the quality of the retrieved ads. Combining the two approaches is an interesting direction for future work.

Another area of research relevant to this paper is search personalization. Some recent work in this area proposed that the performance gains that can be attained by personalization ("potential for personalization") vary by query [14, 33]. We demonstrate that potential for personalization can be viewed as prior knowledge about the query at hand, and tie it to other prior knowledge aspects such as query difficulty.

There is a rich past research on document priors. Previous work was usually restricted to document-dependent features such as document length [32, 3, 20], anchor text [20, 28], document centrality in a link graph [4, 19, 11], document popularity (based on toolbar activity) [25] and a combination of all of the above [30]. In Section 4.2, we show that using aggregates of pairwise query-document features such as clicks in addition to previously explored document-based features can lead to better document prior estimate and improved ranking.

Most important contribution of our work is in showing that all of the above research problems can be formulated within a single framework for prior knowledge estimation. Unlike previous work, our framework is not committed to a specific type of knowledge, can be applied to both documents and queries and allows incorporating any query-dependent features, document-dependent features and pairwise query-document features through multiple aggregate functions.

6. CONCLUSIONS

In this paper we proposed a unified framework for estimation of prior knowledge in web search. Within our framework, a single set of document-dependent, query-dependent and query-document dependent features is used for estimating prior knowledge, and document and query priors are obtained by a feature aggregation process, which marginalizes out the dependencies on the unwanted dimension.

We empirically demonstrate that our framework for estimating prior knowledge about both the documents and the queries can be successfully applied to a diverse set of tasks using the same set of features. Some of these tasks (e.g., query difficulty prediction or document ranking) were explored in previous work in separate contexts using distinct data sources; some of these tasks are new (e.g., detecting query commercial intent and query malleability).

For all of the examined tasks we show that *more is more*: within our framework, a combination of multiple data sources (e.g., web link graph, toolbar counts, retrieval scores or click data) and multiple aggregates (e.g., mean or entropy) into a unified set of features always posits statistically significant performance improvements over the baselines that use either a single data source or a single aggregate.

7. REFERENCES

[1] E. Agichtein and Z. Zheng. Identifying "best bet" web search results by mining past user behavior. In *Proc. of KDD*, pages 902–908, 2006.

[2] P. Bailey, N. Craswell, I. Soboroff, P. Thomas, A. P. de Vries, and E. Yilmaz. Relevance assessment: are judges exchangeable and does it matter. In *Proc. of SIGIR*, pages 667–674, 2008.

[3] R. Blanco and A. Barreiro. Probabilistic document length priors for language models. In *Proc. of ECIR*, pages 394–405, 2008.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, April 1998.

[5] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[6] A. Broder, M. Ciaramita, M. Fontoura, E. Gabrilovich, V. Josifovski, D. Metzler, V. Murdock, and V. Plachouras. To swing or not to swing: Learning when (not) to advertise. In *Proc. of CIKM*, pages 1003–1012, 2008.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of ICML*, pages 89–96, 2005.

[8] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proc. of SIGIR*, pages 390–397, 2006.

[9] K. W. Church and W. A. Gale. Poisson mixtures. *Natural Language Engineering*, 1:163–190, 1995.

[10] J. Cohen. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220, 1968.

[11] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *Proc. of SIGIR*, pages 416–423, 2005.

[12] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR*, pages 299–306, 2002.

[13] F. Diaz and R. Jones. Using temporal profiles of queries for precision prediction. In *Proc. of SIGIR*, pages 18–24, 2004.

[14] Z. Dou, R. Song, and J. Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proc. of WWW*, pages 581–590, 2007.

[15] C. Hauff, V. Murdock, and R. Baeza-Yates. Improved query difficulty prediction for the web. In *Proc. of CIKM*, pages 439–448, 2008.

[16] F. Jelinek. Self-organized language modeling for speech recognition. In *Readings in speech recognition*, pages 450–506. Morgan Kaufmann Publishers Inc., 1990.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *Proc, of SIGKDD*, pages 133–142, 2002.

[18] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. of WWW*, pages 387–396, 2006.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[20] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proc. of SIGIR*, pages 27–34, 2002.

[21] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289, 2001.

[22] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proc. of WWW*, pages 391–400, 2005.

[23] J. Leskovec, S. Dumais, and E. Horvitz. Web projections: learning from contextual subgraphs of the web. In *Proc. of WWW*, pages 471–480, 2007.

[24] T. Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: benchmark dataset for research on learning to rank for information retrieval. In *Proc. of the Learning to Rank Workshop, SIGIR*, 2007.

[25] Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. He, and H. Li. BrowseRank: letting web users vote for page importance. In *Proc. of SIGIR*, pages 451–458, 2008.

[26] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[27] Q. Mei and K. Church. Entropy of search logs: how hard is search? with personalization? with backoff? In *Proc. of WSDM*, pages 45–54, 2008.

[28] J. Peng and I. Ounis. Combination of document priors in web information retrieval. In *Proc. of ECIR*, pages 732–736, 2007.

[29] J. M. Ponte and B. W. Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR*, pages 275–281, 1998.

[30] M. Richardson, A. Prakash, and E. Brill. Beyond PageRank: machine learning for static ranking. In *Proc. of WWW*, pages 707–715, 2006.

[31] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proc. of IEEE*, 88(8):1270–1278, 2000.

[32] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proc. of SIGIR*, pages 21–29, 1996.

[33] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *Proc. of SIGIR*, pages 163–170, 2008.

[34] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proc. of CIKM*, pages 479–488, 2008.