Matching Task Profiles and User Needs in Personalized Web Search

Julia Luxenburger
Max-Planck Institute of Informatics
Saarbrücken, Germany
julialux@mpi-inf.mpg.de

Shady Elbassuoni
Max-Planck Institute of Informatics
Saarbrücken, Germany
elbass@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute of Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Personalization has been deemed one of the major challenges in information retrieval with a significant potential for providing better search experience to individual users. Especially, the need for enhanced user models better capturing elements such as users' goals, tasks, and contexts has been identified. In this paper, we introduce a statistical language model for user tasks representing different granularity levels of a user profile, ranging from very specific search goals to broad topics. We propose a personalization framework that selectively matches the actual user information need with relevant past user tasks, and allows to dynamically switch the course of personalization from re-finding very precise information to biasing results to general user interests. In the extreme, our model is able to detect when the user's search and browse history is not appropriate for aiding the user in satisfying her current information quest. Instead of blindly applying personalization to all user queries, our approach refrains from undue actions in these cases, accounting for the user's desire of discovering new topics, and changing interests over time. The effectiveness of our method is demonstrated by an empirical user study.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*relevance feedback, retrieval models*

General Terms

Experimentation, Human Factors, Algorithms

1. INTRODUCTION

Search personalization has been pursued in many ways, in order to provide better result rankings and better overall search experience to individual users. However, blindly applying personalization to all user queries, for example, by a background model derived from the user's long-term query-and-click history, is not always appropriate for aiding the user in accomplishing her actual task. User interests change over time, a user sometimes works on very different categories of tasks within a short timespan, and history-based personalization may impede a user's desire of discovering new topics. In this paper we propose a personalization framework that is selective in a twofold sense. First, it selectively employs personalization techniques for queries that are expected to benefit from prior history information, while refraining from undue actions otherwise. Second, we introduce the notion of tasks representing different granularity levels of a user profile, and base our reasoning selectively on query-relevant user tasks. Modeling the whole spectrum of past user tasks, ranging from very specific tasks such as finding a hotel in Borneo to the rather general user interest into traveling, allows to adapt the scope of the employed personalization scheme to the user's current search mode. Elbassuoni et al. [12, 23] identified three major search modes: *re-finding known information*, *finding out about topics of user interest*, and *following an ad-hoc information need*. By dynamically varying the scope of personalization based on the granularity of the best-matching past user task, our method inherently supports and balances the different search goals a user might pursue. It may assist the user on re-finding a previously searched hotel address, or ameliorate ambiguous search results by respecting her interest in traveling. Our reasoning is cast into a sound theoretical model based on statistical language models for tasks and queries.

The outline of the paper is as follows. In Section 2 we review related work. We introduce the architecture of our client-side personalization system in Section 3.1, and detail our personalization framework in Section 3.2. In Section 4 we present experimental results from an empirical user study showing that our approach achieves statistically significant gains in performance over both opponents, Google and a non-selective personalization approach. Also, we back the benefit of structuring the user profile in terms of tasks by comparing our task-aware personalization scheme to a personalization which is based on a rather monolithic user profile. In Section 4.3 we study a variant of our personalization approach which emphasizes the immediate session context whenever possible. In addition to basing personalization on a re-ranking of the top-50 Google results, we consider applying query expansion when appropriate in Section 4.4. We conclude by pointing out ways of learning optimal parameters to our model in a running system (Section 4.5), and discussing the computational complexity of our method at query-processing time (Section 5).

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-59593-991-3/08/10 ...$5.00.

2. RELATED WORK

Our work touches various fields. In the following we review related work in personalization, query performance prediction, and query expansion.

Personalization. There are a number of attempts on personalizing Web search. Due to space limitations we give a high-level categorization of what has been done along with some exemplary references which are, however, not meant to be exhaustive.

One way of personalizing search is by means of implicit user relevance feedback. Approaches along these lines include [29, 26] which also inspired the work presented in this paper. They achieve personalization by a client-side re-ranking of Web search results based on the previous user search and browse behavior. However, each one tackles a single facet of personalization, either biasing search results to general user-interests [29] or respecting the current search session's context [26], while we unify both aspects and dynamically switch between these two search modes. Elbassuoni et al. [12] also aim at distinguishing the various user search modes, and do so by following a simplistic rule-based decision strategy. While their experimental results show improvements for the search modes of re-finding known information and satisfying an ad-hoc information need, their personalization approach shows query-dependent variations and less impressive performance gains when the user is exploring topics of user interest. This user search mode is especially hard as imposing some general topic bias on queries is error-prone and its usefulness is highly query-dependent. Our approach improves on theirs by more carefully differentiating when a biasing of search results towards general user interests is beneficial. Kim and Chan [19] have published yet another personalization approach employing re-ranking of search results from the client side. Our approach and theirs have in common that both methodologies view the user's profile as a hierarchy of general to specific interests. While we reason on tasks based on the user's search history, they employ a hierarchical clustering of terms based on the user's bookmarks. A recent work by Teevan et al. [30] performs a large-scale query-log analysis on implicit measures for predicting query ambiguity. They correlate implicit ambiguity measures with the level of variation found in explicit relevance assessments of users on the same queries, and point out the potential for personalization on ambigious queries.

Another path of addressing personalization is by the categorization of both user interests and search results and a biasing of search results according to some similarity measure on these categories. Approaches along these lines include [22, 8, 31]. Personal biases inside the state-of-the-art link analysis algorithms such as PageRank [3] and HITS [20] provide a further means to shift search results according to user interests. E.g., Haveliwala [15] has been the first to propose biasing the random jumps inside the PageRank algorithm towards pages of user interests. Qiu and Cho [24] further extend this idea by automatically learning topic preferences from the user search behavior. Some personalization techniques not only consider a single user, but also take the actions of a surrounding group of users into account, e.g., Sugiyama et al. [27] follow a collaborative filtering approach.

Query performance prediction. When deciding whether a query might benefit from personalization, we are actually interested in predicting the performance difference between the non-personalized and the personalized search result ranking. Reasonings in similar vein occur in the traditional web search setting when estimating the difficulty or ambiguity of a query. Cronen-Townsend et al. [9] introduce *query clarity* defined as the Kullback-Leibler divergence between the query and the collection language model. The rationale behind is that a query with high divergence from the collection model is of high coherence and little ambiguity, a query having a similar term distribution to the collection is, however, most probably on more than one topic. A direct translation of *query clarity* to the setting of a client-side personalization would measure the divergence of the query from the user history profile. Then a high divergence score might be either due to the query being on a topic not strongly represented in the profile, or the query being in fact very coherent. Still, such an approach would reason on the user profile in its entirety, not distinguishing between the possibly highly diverse user interests.

A comparative study on the effectiveness of query clarity, query length, and measures reasoning on the inverse document frequencies of query terms for predicting query performance has been done by He and Ounis [16]. They find query clarity and the standard deviation of the idf's of query terms to perform best. Grivolla et al. [13] view query performance prediction as a classification problem. As features they consider, e.g., IR ranking scores, entropy and mean cosine similarity of the top result set documents. None of the features shows a perfect correlation with average precision, however, still most of them are useful, especially in combination with other features.

Query expansion. Query expansion is strictly speaking the process of *adding* terms to the original query. However, in a more general sense, it also refers to methods of query reformulation, i.e., any kind of transformation applied to a query to facilitate a more effective retrieval. Based on the data that serves as the knowledge base for transforming the query, existing methodologies can be categorized based on whether they rely on *relevance feedback*, either explicit, implicit or pseudo, collection-based *co-occurrence statistics*, or *thesaurus information*. Similar to personalization, the performance of query expansion is highly query-dependent. It might improve some queries, while harming others. This lately led to the emergence of approaches that try to automatically determine when query expansion is worthwhile [1, 21, 5, 4]. E.g., Cronen-Townsend et al. [10] study a selective query expansion approach that automatically decides when it is beneficial to expand a query based on a comparison between the ranked lists for the original and expanded query. Chirita et al. [7] consider an adaptive personalized query expansion by varying the number of expansion terms based on an empirically determined function of *query scope* with respect to the user profile and the query clarity on the web. However, this approach always interprets a user profile in its entirety, thus adapting only the *number* of expansion terms dynamically instead of the whole process of term selection.

To the best of our knowledge we are the first to devise a full-fledged personalization model encompassing adaptivity as an integral part of the model.

3. MODEL AND ALGORITHMS

3.1 Architecture

Before we introduce our personalization framework, we introduce the key characteristics of our personalization architecture. As especially the browsing activities beyond search are outside the reach of a search engine, client-side solutions are favorable. Moreover, as all user data is kept locally, user privacy is not violated. We therefore set up a client-side search personalization with the use of a proxy which is running locally. It intercepts all HTTP traffic, extracts queries, *query chains*, i.e., subsequently posed queries, result sets, clicked result pages, as well as the whole *clickstream* of subsequently visited web pages, and stores this information to a local database file which we refer to as the *user profile* in the following. Accordingly, searches with Google (the same approach can be easily applied to any other search engine as well) are intercepted and search results are re-ranked according to personal preferences. We preferred a proxy over implementing a plugin for browser-independence. Moreover, the proxy is broader applicable as it may bundle several users and thus achieve biasing of search towards community interests. The proxy we are using relies on the UsaProxy implementation [2] that enhances all html files passed through by some Javascript code that sends logging information on events such as the load of page, mouse movements, etc back to the proxy.

For the following discussion we define our notion of a *search session* which is based on heuristics about the user's timing as well as the relatedness of subsequent users' actions. User actions are (1) queries, (2) result clicks, and (3) other page visits. All successive actions are considered to be within the same search session as long as their similarity exceeds a certain predefined threshold. When computing the similarity of subsequent actions, a query is represented by the centroid of the top-50 result snippets.

Result re-ranking. Our search agent retrieves more results than the typical user is likely to view (50 results). Whenever a user action allows to update the query representation, unseen results are re-ranked. E.g., this is the case when the user submits a query or when she presses the "Next" link to view more results. Yet we refrain from re-ranking when the user returns to a seen result list using the "Back" button, as we perceived this more as irritating than as advantageous.

Query expansion. For some queries we might even decide to rewrite the query sent to the search engine, thus giving personalization most probably a larger impact by not only re-ordering the original results, but also retrieving a potentially very different result set.

Merging of personalized and original results. In order to incorporate the query-independent web page importance, personalized result ranks and original web ranks (as an approximation for the real page rank) are aggregated to form the final result ranking. Inspired by rank aggregation methods for the web presented in [11], we use Borda's method to combine the two result rankings. Thereby each result item is assigned a score corresponding to the number of results ranked below it. Then the total score of a result is a weighted sum of its scores with respect to each ranking.

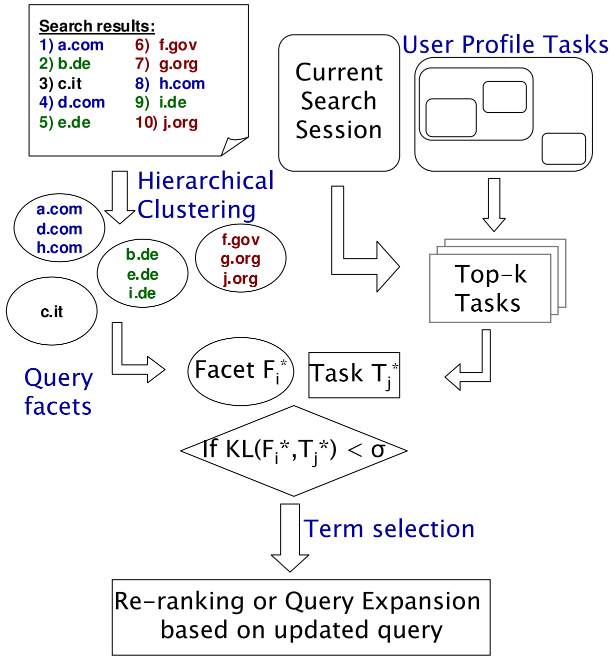

Figure 1: Selective and task-aware personalization framework.

3.2 Personalization Framework

We propose a framework based on fine-grained language models for units of a user's past search and browse behavior, coined *tasks*. They reflect the non-homogeneous, various aspects of user interests which might, e.g., range from the specific task of searching a hotel in Borneo to the general interest in traveling. We obtain tasks by means of a hierarchical clustering of the user's profile. The atomic units to be clustered are past sessions which consist of cohesive query chains, i.e., subsequently posed queries including their result clicks and further browsed documents within the same session, or query-independently browsed documents. Thus we represent the user profile as the full dendogram of tasks which reaches from small tasks consisting only of a single session, to the largest task encompassing the whole user search and browse history.

Similarly, for each query, we perform a hierarchical clustering of the query's result set items (each represented by its title and snippet information) to obtain candidate query *facets* $F_1, \ldots, F_m$ which represent the different aspects the query might span. Then our approach indirectly determines the ambiguity level of a query, and its presence in the user profile as follows (also see Figure 1 for visualization).

Selective Personalization Strategy. We distinguish two cases: either (1) the current query is the first query in the current session, or (2) there exists some query history already, and the current query is a refinement of a previously issued query in the same search session. In the first case we retrieve the *top-k* tasks $T_1, \ldots, T_k$ most similar to the query from the user's profile. In the latter case the tasks present in the user profile are accompanied by a current task made up by all the actions of the currently active session, and represented by the language model T_{k+1} being just a special incarnation of a task language model.

Each obtained query facet and each task is represented by a unigram language model. Considering the Kullback-Leibler (KL) divergence between a query facet F_i and a task T_j

$$KL(F_i||T_j) = \sum_w P(w|F_i) \cdot \log \frac{P(w|F_i)}{P(w|T_j)}$$

we determine the facet-task pair (F_i^*, T_j^*) with the lowest Kullback-Leibler (KL) divergence (we approximate $KL(F_i||T_j)$ by considering only those terms present in $F_i \cup T_j$). That way we learn the query facet F_i^* closest to the user's interests, which represents the most probable meaning of the query in case of ambiguity. At the same time, the task T_j^* represents the best-matching part of the user profile to the query facet F_i^*. The Kullback-Leibler divergence between between F_i^* and T_j^* further characterizes the strength of their similarity. If $KL(F_i^*, T_j^*)$ is larger than a threshold σ, we conclude that the current query goes for a previously unexplored task, and thus refrain from biasing the search results. Otherwise we might either reformulate the query sent to Google or re-rank the original search results as described in the next section.

Means of Personalization. In the following we tackle the question how to personalize when personalization has been deemed useful for the current query, that is we are given a facet of the current query F_i^* and a task T_j^* most similar to each other. We update the query representation with terms best discriminating the query facet F_i^* from all other query facets, while being most similar to the task T_j^*, i.e., terms which have the largest impact on the KL-divergence between the union of the chosen facet-task pair and the remaining query facets, for which thus

$$v(w) = P(w|F_i^* \cup T_j^*) \cdot \log \frac{P(w|F_i^* \cup T_j^*)}{P(w|\bigcup_{i \neq i^*} F_i)} < \tau$$

with

$$P(w|\bigcup_{i \neq i^*} F_i) = \frac{1}{|\{i|i \neq i^*\}|} \cdot \sum_{i \neq i^*} P(w|F_i)$$

holds. Based on the KL divergence between the new query representation and the result items' language models estimated from their title and snippet information, we re-rank the original results. Thus in case of ambiguity, the query is biased towards the facet with the best-matching counterpart in the user's profile. We introduce a third threshold δ to allow for an automatic reformulation of the query sent to Google whenever we find strong terms. Terms for which $v(w) < \delta$ qualify for query expansion, while terms with $\delta \leq v(w) < \tau$ and $P(w|\bigcup_i F_i) > 0$ qualify for re-ranking the original top-50 search results. That way the use of the more invasive query expansion is restricted to queries for which there is a certain level of confidence in its success.

Alternatively to selecting terms based on their contribution to the KL-divergence as just described, we consider ranking terms based on their probability in the facet-task pair, i.e., $v(w) = P(w|F_i \cup T_j)$, as well as, thresholding the number of terms to be selected.

3.3 Language models

The language model of a user task is a weighted mixture of its components: queries, result clicks, clickstream documents and query-independent browsed documents. Let Q be a query language model, and B a language model of query-independently browsed documents. Then the task language model T is obtained as

$$P(w|T) \quad = \quad \alpha \cdot P(w|Q) + (1 - \alpha) \cdot P(w|B)$$

where w denotes a word in the vocabulary. While B is the average of the individual browsed documents' language models, Q is the uniform mixture of the task's query chains as follows. Let QC denote a query chain $Q_1, Q_2, \ldots, Q_k$ where Q_k is the last query in the chain. Then

$$P(w|Q) = \frac{1}{|QC|} \cdot \sum_{QC} \frac{1}{\sum_{i=1}^{k} \frac{1}{k-i+1}} \cdot \sum_{i=1}^{k} \frac{1}{k-i+1} \cdot P(w|Q_i)$$

, i.e., a task's queries' language model is the average of all query chains' models. Each query chain is modeled by a weighted sum of its constituent queries, weighted in such a manner that queries later in the chain are considered more important than queries early in the chain. The rationale behind this is the assumption that queries submitted later in the session, are better matches to the user information need, and thus better characterize the search task's intention.

Let $r(\cdot)$ denote the rank of a search result document, and CR be the set of clicked result items of the total set of result items R. Among the non-clicked result items, we further distinguish intentionally non-clicked documents ranked above a clicked one, $NR = \{d \in R, \exists c \in CR : r(d) < r(c)\}$, and unseen results ranked below the lowest-ranked clicked item, $UR = \{d \in R, \forall c \in CR : r(d) > r(c)\}$. Furthermore, let CS be the set of clickstream documents beyond the result documents. Then the individual query models can be estimated as

$$\begin{aligned} P(w|Q_i) \quad = \quad & \lambda_q \cdot P(w|q) + \lambda_c \cdot \sum_{d \in CR} \frac{1}{|CR|} \cdot P(w|d) \\ & + \lambda_n \cdot \sum_{d \in NR} \frac{1}{|NR|} \cdot P(w|d) \\ & + \lambda_u \cdot \sum_{d \in UR} \frac{1}{|UR|} P(w|d) \\ & + \lambda_s \cdot \sum_{d \in CS} \frac{1}{|CS|} \cdot P(w|d) \end{aligned}$$

, i.e., a single query is represented by a mixture of its actual query terms (q denotes the query string), its clicked documents, its intentionally non-clicked documents ranked above clicked ones, its unseen result documents ranked below all clicked ones, and all the documents the user visited starting from clicked result set documents. The constituent language models for both clicked and clickstream documents are a uniform mixture of their document language models, and similarly the language models for unseen and intentionally non-clicked results are uniform mixtures of their individual result models estimated from their title and snippet text. All constituent language models employ Dirichlet prior smoothing, i.e.,

$$P(w|d) = \frac{c(w,d) + \mu \cdot \frac{c(w,Coll)}{|Coll|}}{|d| + \mu}$$

where $c(w, \cdot)$ denotes the frequency of word w in $(\cdot)$, $Coll$ is short for collection which amounts to the user profile in our setting, $|.|$ is the overall length of the document or the collection, and μ is typically 2000 (see [32] for more details).

User	#Queries	#Browsed documents	#Sessions
1	284	1014	593
2	215	749	522
3	200	243	184
4	135	420	377
5	273	1389	1188
6	238	1606	1547
7	290	424	410

Table 1: Size of user log history

Similarly, the facet language model F is the uniform mixture of the result snippets $s \in F$ present in the facet, i.e.,

$$P(w|F) = \frac{1}{|F|} \cdot \sum_{s \in F} \frac{c(w,s) + \mu \cdot \frac{c(w,Coll)}{|Coll|}}{|s| + \mu}$$

4. EXPERIMENTS

4.1 Experimental Setup

In order to evaluate the effectiveness of our proposed approach, we asked 7 volunteers (computer scientists at our institute) to install our proxy on their local machines to log their search and browsing activities for a period of 2 months. Participants were free to opt out of the logging to protect their privacy. Due to holiday seasons, the period actually covered is approximately 6 weeks. The range of users spans from highly active to only sporadically searching and browsing the web. Thus, also the amount of logged user history varies accordingly. Table 1 shows the respective log sizes in terms of number of distinct query strings issued, number of web documents viewed, and number of sessions. These numbers already show the high divergence of web interaction behavior between users, e.g., user 1 uses primarily search to access web content, whereas user 5 has a stronger emphasis on direct or browsing-based navigation.

During a one-week evaluation phase, each participant evaluated 8 self-chosen search tasks, among which at least two were asked to reflect topics searched during the logging period. A search task is a sequence of queries, click and browsing actions until the user's information need is satisfied. We assisted users in remembering information they searched before by presenting them the frequency-ordered terms of their logged query strings. During evaluation, we tried to mimic natural queries by having the participants evaluate their common-day queries at their own pace. The performed evaluation search tasks consisted of 2.3 queries on average, from which the first, the last, and the next to last query if available were subject to evaluation. For each task, the participant was presented with the top-50 Google results for the selected queries, merged and placed in random order in order to avoid result's position bias. Then the participant was asked to mark each result as highly relevant, relevant or completely irrelevant. Furthermore, we asked users to group the top-10 results of each query by giving labels to them. This was to generate the optimal clustering of search results as perceived by the individual user, and to decouple the evaluation of our personalization strategy from the quality of the search result set clustering. In total our experiment comprises 59 search tasks, and 98 individual evaluation queries. We removed queries whose search results were lacking human labels and queries that had no matching tasks in the user profile as no personalization could be carried out on these queries. These considerations left us with a set of 89 evaluation queries.

To measure the ranking quality, we use the Discounted Cumulative gain (DCG) [17], which is a measure that takes into consideration the rank of relevant documents and allows the incorporation of different relevance levels. DCG is defined as follows

$$DCG(i) = \begin{cases} G(1) & \text{if i = 1} \\ DCG(i-1) + G(i)/log(i) & \text{otherwise} \end{cases}$$

where i is the rank of the result within the result set, and $G(i)$ is the relevance level of the result. We used $G(i) = 2$ for highly relevant documents, $G(i) = 1$ for relevant ones, and $G(i) = 0$ for non-relevant ones. Dividing the obtained DCG by the DCG of the ideal ranking we obtain a normalized DCG which accounts for the variance in performance among queries.

	1	2	3	4	5	6
λ_q	0.2	0.3	0.25	0.25	0.5	0.33
λ_c	0.2	0.3	0.25	0.5	0.25	0.33
λ_s	0.2	0.3	0.5	0.25	0.25	0.33
λ_u	0.2	0.1	0	0	0	0
λ_n	0.2	0	0	0	0	0

Table 2: Considered parameter settings for the task query language model.

During our experiments we studied various degrees of freedom of our framework, such as the mean of selecting expansion terms, i.e., whether terms were ranked based on their probability in the chosen facet-task pair as opposed to their contribution to the KL-divergence between the chosen facet-task pair and the remaining query facets. Other parameters we varied are whether to enforce the inclusion of the original query terms in the course of query rewriting, as well as, the weight for aggregating original and personalized results using Borda's method. All these general parameters were fixed among all users during all experiments.

Besides these general variations of our personalization approach, we also experimented with different values for the weights in the task language model for each user separately to account for the diverse user styles of interacting with the Web. We varied the weight of queries with respect to browsed documents. We chose $\alpha \in \{0.5, 0.7, 1\}$, thus testing the extreme points of giving equal importance to queries and independently-browsed documents, as opposed to giving higher weight to queries up to ignoring independently-browsed documents when $\alpha = 1$. Similarly we changed the influence of the individual components of the task query language model by giving different weights to λ_q, λ_c, λ_n, λ_u and λ_s. These regulate the importance of query strings, clicked documents, non-clicked documents, unseen documents and query-dependent clickstream documents with respect to each other. Table 2 gives a summary of the parameter combinations in the task query language model we studied. In total, we restricted ourselves to 18 parameter combinations. Finally, the threshold σ which determines when personalization is employed, as well as, the threshold τ which regulates the number of expansion terms on a query-to-query basis, or alternatively, a parameter fixing the number of expansion terms to be selected, could be tuned.

The rest of this section is divided as follows. We first present the experimental results we obtain when restricting our means of personalization to result re-ranking only. In addition, we study another incarnation of our approach where we enforce the use of the current session task whenever possible to account for the particular importance of the current session context. Then we present the results we obtain by also employing query expansion within our personalization framework. As a first proof-of-concept and as an indicator of the potential of our approach, we report in the following results based on the best performing parameters setting. We conclude by proposing a method to learn and adjust the optimum values for these parameters automatically from the user's profile.

4.2 Evaluation results with re-ranking

We computed the NDCG at top-10 for the original Google results, enforced personalization results where personalization has been carried out for each query, and our *selective personalization* results where personalization has been performed only for queries whose KL divergence is below the threshold σ. We experimented with two incarnations of our selective personalization approach. In the first case, which we refer to as *Fixed*, we adopted the same fixed number of expansion terms among all user queries, whereas in the other, coined *Flexible*, we determined the optimal threshold τ which varies the number of expansion terms between different queries. In both cases, in order to give the enforced personalization a fair chance and make results comparable, we chose the user-specific parameters that would maximize the improvement in NDCG between enforced personalized and original rankings. This would determine all user-specific parameters of the selective personalization approach except for the thresholds σ and τ which we chose such that the performance of selective personalization is maximized while keeping all other parameters fixed.

	Fixed		Flexible	
	Avg	Std dev.	Avg	Std dev.
Original	0.569	0.120	0.569	0.120
Enforced	0.536	0.132	0.594	0.107
Enforced (no facets)	0.473	0.133	0.543	0.108
Enforced (no tasks)	0.454	0.131	0.519	0.136
Selective	**0.586**	0.134	**0.587**	0.117
σ	-0.015	0.015	-0.003	0.022

Table 3: Average NDCG@10 for best parameter setting per user aggregated over all users with human generated query facets.

	Fixed		Flexible	
	Avg	Std dev.	Avg	Std dev.
Original	0.569	0.120	0.569	0.120
Enforced	0.542	0.129	0.594	0.112
Enforced (no facets)	0.472	0.138	0.534	0.126
Enforced (no tasks)	0.454	0.131	0.517	0.134
Selective	**0.593**	0.119	**0.600**	0.114
σ	-0.020	0.024	-0.005	0.023

Table 4: Average NDCG@10 for best parameter setting per user aggregated over all users with automatically generated query facets.

Tables 3 and 4 present the aggregated average performance. We chose the best performing parameters setting for each individual user independently, and then averaged the computed NDCG values over our 7 participants. Table 3 represents the results where we relied on the "ideal" user clustering to construct query facets. Consequently, Table 4 represents the case where the automatic clustering of the search results was used instead.

Our selective personalization outperforms both the original ranking and the enforced personalization in the case of fixed number of expansion terms. We report significantly large improvements over the original ranking with one-tailed paired t-test p-values of 0.026 for human-labeled facets, and 0.024 for automatically generated facets. Also, the improvements over the enforced personalization are statistically significant with p-values of 0.023 (human) and 0.021 (automatic). Similarly, we obtained statistically significant improvements for flexible number of expansion terms over the original Google ranking. The enforced personalization with flexible number of expansion terms already benefits from the selectivity in the number of expansion terms, and also outperforms the original Google ranking significantly with p-values of 0.048 (automatic) and 0.022 (human). We find only small differences between the results obtained when employing a selective personalization with a fixed as opposed to a flexible number of expansion terms. Comparing the hierarchical clustering approach for the generation of query facets to "ideal" query facets as perceived by users, we find the automatic approach to slightly outperform a human grouping indicating the validity of the hierarchical clustering approach for determining query facet candidates.

To evaluate the merit of a task-aware personalization approach, we compare three variants of enforced personalization. As a baseline, we consider a personalization methodology that is unaware of both query facets and tasks, which we refer to as "enforced (no tasks)". As an enhancement thereof, we also consider a variant that serves as an intermediate step and distinguishes between history tasks but still treats a query always in its entirety which we refer to as "enforced (no facets)". Comparing the enforced personalization that utilizes both our notion of query facets and search history tasks to these two baselines indeed proves the concepts of user tasks and query facets beneficial.

Correlating KL-divergence with performance gains. In Table 5, we report the correlation between the improvement in NDCG of the enforced personalization with fixed number of expansion terms over the original ranking and the KL-divergence between the chosen facet-task pair $KL(F_i^*, T_j^*)$. A negative correlation indicates that queries with more relevant information in the local index (i.e., with a low KL divergence) experience higher improvement in NDCG as compared to those that have less relevant information in the local index (i.e., with a high KL divergence). We find that users who benefit the most from personalization, i.e., with strongest improvement of selective personalization over the original ranking, also show a medium negative correlation between the improvement in NDCG of the enforced personalization over the original ranking and $KL(F_i^*, T_j^*)$. This indicates the validity of our approach for predicting whether a query benefits from personalization.

Parameterizing the personalization framework. Tables 6 and 7 show the chosen user-specific parameters setting when using fixed and flexible number of terms, respectively. We only report the case where query facets were automat-

User	1	2	3	4	5	6	7
O	0.537	0.616	0.645	0.35	0.768	0.504	0.564
E	0.547	0.459	0.644	0.398	0.8	0.441	0.506
S	0.572	0.617	0.656	0.409	0.82	0.504	0.572
%	70%	88.8%	50%	100%	88.8%	70.6%	81.8%
π	0.005	0.402	0.347	**-0.5**	**-0.253**	0.037	**-0.33**

Table 5: Pearson correlation (π) between $KL(F_i^*, T_j^*)$ and the improvement of enforced personalization (E) over the original ranking (O). Selective personalization (S) and the percentage (%) of times it decides right. We report NDCG@10 for fixed number of expansion terms and automatically-generated facets.

ically generated. For the majority of users, browsed documents play a role as α equals either 0.7 or 0.5 in most cases. Regarding the task query language models, the weights of its components varied among the different users supporting our intuition that these weights are indeed user-dependent.

User	1	2	3	4	5	6	7
α	0.7	0.7	0.7	1	0.5	0.5	0.7
λ_q	0.25	0.2	0.5	0.2	0.2	0.25	0.5
λ_c	0.5	0.2	0.25	0.2	0.2	0.25	0.25
λ_s	0.25	0.2	0.25	0.2	0.2	0.5	0.25
λ_u	0	0.2	0	0.2	0.2	0	0
λ_n	0	0.2	0	0.2	0.2	0	0

Table 6: Chosen parameter settings per user with automatically generated query facets and flexible number of terms

User	1	2	3	4	5	6	7
α	1	0.7	0.5	0.7	0.7	0.5	0.5
λ_q	0.2	0.3	0.3	0.25	0.25	0.2	0.2
λ_c	0.2	0.3	0.3	0.25	0.25	0.2	0.2
λ_s	0.2	0.3	0.3	0.5	0.5	0.2	0.2
λ_u	0.2	0.1	0.1	0	0	0.2	0.2
λ_n	0.2	0	0	0	0	0.2	0.2

Table 7: Chosen parameter settings per user with automatically generated query facets and fixed number of expansion terms

For the general parameters that were fixed among all the users, we found the exclusion of the original query terms, term selection according to terms' facet-task probabilities and the aggregation of original and personalized results with an equal weight of 0.5 to be the best performing parameter setting for selective personalization with both fixed and flexible number of expansion terms.

4.3 Considering the current session's context

When comparing the performance of queries across the various time points in a session (see Figure 2), we find a large gap between the first query in a session and both the last and the query before the last in terms of performance of the original Google ranking (first: 0.33, last: 0.7 average NDCG@10). The difference in performance between the first and the last query in a session is significant with a one-tailed paired t-test p-value of 0.005. This indicates that indeed the user succeeds in improving result quality solely based on her query reformulation skills which motivates us to study a stronger consideration of session context within our personalization framework.

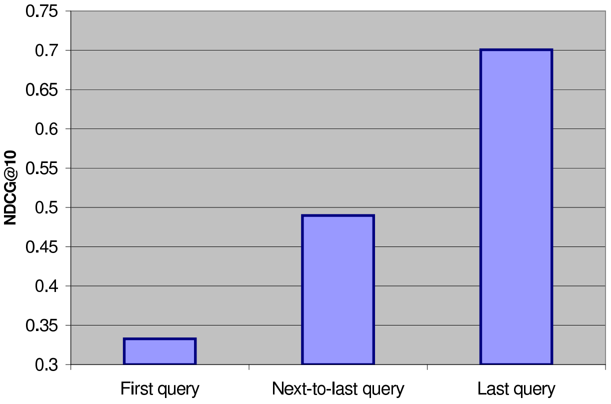

Figure 2: Query performance gaps with respect to the query's position within a search session.

Here, we report the results for a variant of our model where we enforce the use of the current session context as the chosen task whenever available. We again report the results where for each user, we chose the setting that optimizes the enforced personalization NDCG, restricting the set of evaluated queries to those that have some current session context associated with. Table 8 represents the average performance aggregated over all users. We find statistically non-significant improvements of *selective personalization* over both the original ranking and the enforced personalization for both the variant with fixed and the one with flexible number of expansion terms. The correlation between the improvement in NDCG and the KL divergence is negative in both cases. We again highlight the importance of considering the different query facets by comparing the enforced personalization that takes into consideration these query facets, with the one that treats a query always in its entirety. As indicated in Table 8, the former enforced personalization outperforms the personalization without facets. Comparing the selective personalization that freely chooses an appropriate user profile task to the one that enforces the use of the current session task on the reduced set of queries for which current session information is available, shows clearly the benefit of this simple heuristic to aid the selective personalization in selecting the right user task.

	Fixed		Flexible	
	Avg	Std dev.	Avg	Std dev.
Original	0.555	0.161	0.555	0.161
Enforced	0.437	0.187	0.537	0.174
Enforced (no facets)	0.353	0.190	0.486	0.165
Selective	**0.584**	0.151	**0.576**	0.161
Selective with current session enforced	**0.588**	0.171	**0.590**	0.167
σ	0.026		0.030	
Pearson correlation	-0.358		-0.436	

Table 8: Average NDCG@10 for best parameter setting per user aggregated over all users with current session enforced and automatically generated query facets.

4.4 Adding Query Expansion

The results we have presented so far all rely on result reranking of the original user's query result set. One drawback of result re-ranking is the high dependence on the performance of the original query. A query with a small recall, would not stand a high chance of improvement. On the other hand, automatic query expansion, if not applied carefully, could harm the query performance. Our framework allows for judicious query expansion, reasoning on how well the expansion terms selected from the user's facet-task pair are believed to present the current user's search actions.

Revisiting Section 3.2, our personalization strategy allows for automatic query expansion, by augmenting the original user's query with additional terms before forwarding it to the search engine. We only expand queries that are believed to benefit from personalization, i.e., queries for which the kl-divergence between the chosen facet-task pair is below the threshold value σ as set during the re-ranking phase. For each qualifying query, we experimented with two variants of expansion methods. The first method, which we refer to as ”enforced expansion” expands each qualifying query with kl-divergence below σ with terms for which $v(w) < \tau$. Next, we evaluate our selective personalization scheme as described in Section 3.2, where we combine both the merits of query expansion and result re-ranking. Queries that have expansion terms with scores $v(w) < \delta$ qualify for query expansion. If there are no such terms, we re-rank the original results using terms that have $v(w) < \tau$, otherwise we refrain from personalization completely. We contrast these two variants employing query expansion, with the selective personalization which employs only re-ranking.

To evaluate query expansion, we only considered queries that were chosen for personalization by our selective approach. We expanded each such qualifying query adding terms in incremental manner, and fetching the top-5 results for the expanded query. We only added terms whose $v(w)$ were below the chosen threshold τ, i.e, terms that were chosen by our selective re-ranking approach with flexible number of terms, and presented the user with the merged set of results in random order. Similar to the case of re-ranking, each user had to assess each result being highly relevant, irrelevant or completely irrelevant. We restricted ourselves to only 5 results per query, in order to avoid overwhelming users with a large number of results. We also believe that it is sufficient to test the effect of query expansion on top-5, since query expansion substantially changes the result set.

Table 9 represents the performance of query expansion aggregated over all 7 users. The reported NDCG values are all at top-5. Both our selective personalization scheme, and the selective re-ranking outperform enforcing query expansion and the original Google results, yet the selective re-ranking has the largest gains.

	Automatic	
	Average	Std deviation
Original	0.558	0.208
Enforced Expansion	0.475	0.253
Selective (Expansion with Re-ranking)	**0.578**	0.185
Selective (only Re-ranking)	**0.637**	0.234
δ	0.047	0.021

Table 9: Average NDCG@5 for query expansion with flexible number of expansion terms and automatically generated query facets.

4.5 Parameters learning

In a running system, it is crucial to automatically determine the values of the different system parameters. In [28], Taylor et al. discuss the lack of efficient learning algorithms for retrieval functions, and compare a greedy line search approach that directly optimizes NDCG to a gradient descent approach optimizing the order of document pairs induced by the ranking function. Both of these methods are approximations to doing an exhaustive search in the parameter space which becomes infeasible with a large number of parameters. Our case is even more involved since some of the parameters we want to learn are not a direct component of the ranking function, but indirectly influence the retrieval function by influencing the number and choice of terms added to the query representation.

The experimental results we presented so far are based on doing an exhaustive search on our set of test queries, thus providing us with an upper bound on the performance of our approach indicating its potential. In the following we discuss methods for determining optimal parameters in a running system. Most of the aforementioned work assumes that we know an objective performance function, such as NDCG, for each parameter setting we might want to consider. This, however, requires to ask users for explicit relevance assessments which the user usually is reluctant to give. Thus methods for implicitly determining the optimal parameters from the user's search behavior are needed.

In the following, we present our experimental results for learning parameters from explicit user relevance assessments, as well as, implicit user feedback. For each participant, we extracted the last 8 queries that were recorded during the logging period, and used these queries as a training set. For each selected query, we ensured that the user had clicked on at least one result ranked below the first one. The rationale behind this is to ensure that there is a potential for improvement over the original ranking. We also asked our participants for relevance assessments of the top-10 results for each query in the training set.

Explicitly learning parameters. We determine the parameter values which maximize the improvement in NDCG between enforced personalized and original rankings on the training set. Applying the parameters learnt on this training set on our test queries set, we obtain evaluation results as reported in Table 10. Even though, the learnt parameters do not fully generalize, selective personalization still improves over enforced personalization when using fixed (p-value = 0.046) or flexible (p-value = 0.055) number of terms.

	Fixed		Flexible	
	Avg	Std dev.	Avg	Std dev.
Original	0.569	0.120	0.569	0.120
Enforced	0.499	0.160	0.481	0.189
Enforced (no facets)	0.496	0.164	0.455	0.182
Enforced (no tasks)	0.468	0.161	0.473	0.178
Selective	**0.517**	0.150	**0.494**	0.186
σ	-0.005	0.021	0.005	0.026

Table 10: Average NDCG@10 for best parameter setting per user aggregated over all users with explicitly learnt parameters.

Implicitly learning parameters.

We utilize the user click information to learn parameters implicitly on the same training set. As most queries occur only once, methods such as the one by Carterette et al. [6] which estimate DCG based on click rates of a significant number of users are not applicable to our client-side personalization setting. Thus we experiment with a naive heuristic which chooses parameters in such a way that clicked results would be ranked the highest. We obtain an estimate of the expected performance of this heuristic by comparing the NDCG of the original rankings in our training set, with the ones we obtain by re-shuffling results in such a way that either (1) all clicked results are ranked highest followed by the remaining ones in original order, or (2) clicked results are ranked top, followed by unseen results, and intentionally non-clicked results at the bottom. Averaging the NDCGs over the training queries of all users yields an average NDCG of 0.863 for the original ranking, 0.897 for the first re-ordering heuristic and 0.857 for the second one. The differences between original and heuristic 1 are statistically significant with a one-tailed paired t-test p-value of 0.031, and also the performance gap between the two heuristics are statistically significant with a p-value of 0.004. This led us to choose the first heuristic for learning parameters. Its performance improvement over the original ranking opens up potential for learning parameters that improve over the original ranking. At the same time, these numbers raise awareness of the much smaller margin of potential improvement as opposed to what might be learnt from explicit ratings which optimize parameters towards a targeted NDCG of 1. This indeed indicates the challenge of deducing relevance assessments from clicks, which suffers from result position bias [18], and noisy user click behavior [25].

Table 11 shows the results of using the parameter settings learnt by relying on implicit user feedback on our test set. Similarly as with explicit learning, our selective approach does not fully generalize to the test evaluation set, however, still outperforms enforced personalization in both cases when using fixed (p-value = 0.013) or flexible (p-value = 0.002) number of terms. Increasing the size of the training set, we expect to be less susceptible to such problems. However, the moderate sizes of our user profiles limited us to experiment with a rather small training set.

	Fixed		Flexible	
	Avg	Std dev.	Avg	Std dev.
Original	0.569	0.120	0.569	0.120
Enforced	0.381	0.176	0.510	0.186
Enforced (no facets)	0.391	0.169	0.476	0.183
Enforced (no tasks)	0.381	0.142	0.487	0.173
Selective	**0.466**	0.213	**0.534**	0.182
σ	-0.011	0.014	0.005	0.024

Table 11: Average NDCG@10 for best parameter setting per user aggregated over all users with implicitly learnt parameters.

5. EFFICIENCY

Tasks are computed offline. In a running system, the hierarchical clustering of past user sessions would be recomputed periodically. To fold in new sessions without making a complete re-computation necessary, incremental clustering approaches such as [14] could be employed. To analyze the complexity of our approach at query-processing time, we empirically compare the relative contribution of each query-processing step to the overall latency incurred by our personalization approach. We picked an exemplary user, and averaged the runtime of each processing step over her evaluation queries (see Table 12).

Clearly, the hierarchical clustering of the top-10 search results, the retrieval of the top-k tasks best matching the query string and the construction of their language models, as well as, the pairwise KL-divergence computations between all candidate facet-task pairs are the largest bottlenecks. We considered replacing the hierarchical clustering here with a flat clustering approach which requires to specify an optimal number of clusters n. We experimented with methods for determining the optimal number of clusters. These, however, mostly involve heavy computations as well such as computing clusterings for all possible values of n. Also, the predictions of n we obtained exhibited only medium Pearson correlation with the number of clusters obtained from human labellings.

The time complexity of the retrieval and construction of the top-k tasks can be traded against additional space complexity for fully precomputing all possible task language models. In our experiments we pre-computed term scores only at the session level, and at query-time computed the task language model by aggregation over the constituent sessions.

The complexity of the pairwise KL-divergence computation can be reduced by limiting the number of candidate pairs to be considered. This might be either achieved by choosing a smaller k or reducing the number of query facet candidates. At query-time we perform a hierarchical clustering of the top-10 result snippets. Instead of considering the full hierarchical cluster dendogram, we could remove some of its levels. Yet another option would be to reduce the number of terms in the summation of the KL-divergence computation by approximating the task and facet language models by their most contributing terms, and ignoring terms below some probability threshold. We would, however, need to carefully study the effect of these approximations on personalization performance.

	Query-processing step	% Total runtime
1	Top-k tasks and task LM	38.6%
2	Clustering of search results	43.7%
3	Facet LM computation	0.27%
4	KL-divergence computation	17%
5	Term selection - Maximum Probability	0.002 %
7	Term selection - Maximum Divergence	1.26%
8	Result LM computation	0.29%

Table 12: Breakdown of query run-times.

6. CONCLUSION AND FUTURE WORK

We have proposed a thorough language modeling framework that explicitly addresses user tasks and carefully matches the current user needs with appropriate past user tasks for an amelioration of search results. Our novel task model takes into consideration more user centered feedback information, which does not only rely on past viewed documents, but also incroporates past queries, as well as clickstream behavior. We introduced the notion of selectivity that enables us to avoid the pitfalls of most personalization systems that omit irregularities in users needs. Our framework also combines both result re-ranking and query expansion, selectively choosing the appropriate means of personalization. We presented an empirical user study where our method achieved statistically significant gains over both the original Google ranking and traditional personalization approaches.

During our experiments, we relied on single terms to present the vocabulary of both the task and the facet language model. However, we encountered limits of a mere bag-of-words model to truly capture user interests. Currently, we are analyzing term correlations, and experimenting with ways of augmenting the language models with correlated compounds of terms to capture dependencies between words to further improve the selection process of expansion terms.

Furthermore, we currently always look for a single best-matching task-facet pair. However, we actually could also consider how close the next most matching pairs are and allow more than one pair to qualify. For example, consider the following scenario: a user is both a Java programmer and has traveled to the island Java. However, the programming makes up a larger part of the history. Then in our framework, the query "Java" is biased towards the programming interest, where it actually should recognize that the user has two diverse almost equally strong interests.

7. REFERENCES

[1] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *ECIR*, 2004.

[2] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move - user activity tracking for website usability evaluation and implicit interaction. In *WWW*, 2006.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.

[4] D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer. Automatic query refinement using lexical affinities with maximal information gain. In *SIGIR*, 2002.

[5] C. Carpineto, R. de Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. In *ACM Transactions on Information Systems*, volume 19, 2001.

[6] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS*, 2007.

[7] P.-A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR*, 2007.

[8] P.-A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using odp metadata to personalize search. In *SIGIR*, 2005.

[9] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR*, 2002.

[10] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. A framework for selective query expansion. In *CIKM*, 2004.

[11] C. Dwork, S. R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, 2001.

[12] S. Elbassuoni, J. Luxenburger, and G. Weikum. Adaptive personalization of web search. In *1st SIGIR Workshop on Web Information Seeking and Interaction*, 2007.

[13] J. Grivolla, P. Jourlin, and R. de Mori. Automatic classification of queries by expected retrieval performance. In *SIGIR*, 2005.

[14] K. Hammouda and M. Kamel. Incremental document clustering using cluster similarity histograms. In *WI*, 2003.

[15] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW*, 2002.

[16] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *SPIRE*, 2004.

[17] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, 2000.

[18] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, 2002.

[19] H. Kim and P. Chan. Personalized ranking of search results with learned user interest hierarchies from bookmarks. In *WebKDD*, 2005.

[20] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the ACM-SIAM symposium on Discrete Algorithms*, 1997.

[21] G. Kumaran and J. Allan. Selective user interaction. In *CIKM*, 2007.

[22] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. In *IEEE Trans. on Knowledge and Data Eng.*, 2004.

[23] J. Luxenburger, S. Elbassuoni, and G. Weikum. Task-aware search personalization. In *SIGIR*, 2008.

[24] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *WWW*, 2006.

[25] F. Scholer, M. Shokouhi, B. Billerbeck, and A. Turpin. Using clicks as implicit judgements: Expectations versus observations. In *ECIR*, 2008.

[26] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM*, 2005.

[27] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW*, 2004.

[28] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM*, 2006.

[29] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, 2005.

[30] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *SIGIR*, 2008.

[31] Y. Xu, B. Zhang, Z. Chen, and K. Wang. Privacy-enhancing personalized web search. In *WWW*, 2007.

[32] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, 2001.