Personalizing Search Results on User Intent

Giorgos Giannopoulos*
supervised by Timos Sellis
NTU Athens -
IMIS, "Athena" R.C.
Greece
giann@dblab.ece.ntua.gr

ABSTRACT

Personalized retrieval models aim at capturing user interests to provide personalized results that are tailored to the respective information needs. User interests are however widely spread, subject to change, and cannot always be captured well, thus rendering the deployment of personalized models challenging. In this doctoral work, we describe our approach where we study ranking models from the aspect of search intent. Our approach is query-centric. That is, it does not rely on separate user search profiles/histories nor it personalizes ranking based on topical similarity of queries. Contrary, it examines the search behaviors/intents induced by queries and groups together queries with similar such behaviors, forming *search behavior clusters*. Specifically, we exploit user feedback in terms of click data to cluster the queries. Each cluster is finally represented by a single ranking model that captures the contained intents expressed by users. Once new queries are issued, these are mapped to the clustering and the retrieval process diversifies possible intents by combining relevant ranking functions.

1. INTRODUCTION

Modern data collections and recordings of historic user interaction pave the way for personalized information retrieval which exploits user profiles and historic usage data to re-rank retrieved documents to serve individual information needs. Personalized retrieval aims at computing a ranking model for every user or groups of similar users. Different approaches, including the impact of short long-term search histories [18, 19], context [12, 18], query categories [7, 21], and search behavior and feedback [1, 8, 11, 13] have been studied. Additionally, collaborative filtering techniques for personalized search [19] and learning to rank-based approaches [1, 6, 11, 14, 16, 23] also proved effective in many scenarios. Many of the above techniques, though, are only applicable to registered users of search engines. So, to have all users benefit from the re-ranking they need to be perfectly disambiguated. This is, particularly on shared computers, an issue and renders personalized web search difficult in practice.

In this work, we study an orthogonal approach to re-ranking for web search which does not share these limitations, so that all users benefit equally from re-ranking the results. Our approach is based on the observation that existing approaches mainly focus on the retrieved content and on users' search histories, thus leaving an important aspect unaddressed: The analysis of user search behavior, induced by queries. Behavior is directly observable by user feedback in form of clicks on result pages and allows to reason about the search intent of the users. The intent therefore acts like an unobserved, latent variable, captured by user behavior.

Figure 1 shows a motivating example based on two users and their respective search histories. User 1 issues a query for a new mobile phone. Her search history so far contains only unrelated queries on IR research and cars. A user-specific personalized model would have to resort to the average user model for processing the query and possibly return pdf documents (due to the user's previous clicks on papers) about phones. On the other hand, even though the two users' search histories are topically unrelated, in this new query, user 1 could take advantage of user's 2 search history on cellphones. Finally, search behavior induced by queries does not always depend on content. For example, although user's 2 queries about cellphones seem to form a solid search history topic, however, they regard two distinct subtopics: (a) searches about reviews/information on specific cellphones and (b) searches on financial information about cellphone companies. In the first case, forum or video results are expected to be more helpful, while in the second case, results from news sites are more suitable. Thus, user's 1 query about "new cellphone reviews" would benefit only from user 2 search history on subtopic (a). This example shows that, capturing search behaviors and exploiting them to rerank search results goes beyond content or users and depends, also, on the latent search intents within queries.

Given the above, our approach does not rely on user-specific models but aims at capturing user search intent by grouping queries that entail similar behavior. User 2 for instance shows a particular interest in watching online

*This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

Figure 1: Motivating example: Personalized user models aggregate search histories by topic (cellphones, cars, etc.). Our approach groups queries based on behavior/intent to form a video cluster (green circle).

videos about movies and mobile phones, that is, she frequently clicks on video results for these types of queries. Our system models this behavior by grouping the video related-queries. This clustering is independent of the user and treats all queries and users the same. The final results proposed to user 1 for her query will now consist of different media types that have been associated with mobile phones in the past (e.g. videos), hence, capturing the broad spectrum of user behavior for queries related to mobile phones.

To build models for user intent, we propose to cluster queries with respect to the user intent and learn a ranking function for every cluster. Firstly, a ranking function is learned for every query to capture the user behavior by adaptation to user feedback given by click data. Secondly, the ranking models are grouped so that the resulting clusters correspond to similar user intents. Thirdly, a ranking function is learned for each cluster to represent the contained intent. At deployment time, queries are mapped to the clustering to compute scores expressing how likely the intent of the query is captured by the respective cluster. The final ranking is then induced by a weighted linear combination of ranking functions that are likely to cover the intent of the user, given the query. Combining the ranking functions of several clusters diversifies the results in terms of the captured intents. Experimentally, our approach captures user intent better than baseline methods on a large sample from the Yahoo! query log, achieving higher precision on top-ranks compared to content-based baselines.

The roadmap of the presented doctoral work follows:

1. We model user search behavior utilizing query clickthrough data (query, result list, result relevance judgments) and a widely used machine learning model (Ranking SVM). Specifically, we approaximate the search intents induced by queries, by representing clickthrough data in the learning model's feature space and recognizing groups of similar search behaviors.
2. We describe a baseline solution for training multiple topic-specific ranking models and we introduce two variations of our proposed framework for training behavior-specific ranking models. We compare our approaches with the topic-specific and other baselines, demonstrating the importance of taking search behavior into account.
3. We are, also, currently working on extending our proposed methods to other search settings. Specifically, we work on adapting the concept of personalized ranking function training on RDF data, where, apart from result content, relations between resources and schema should be taken into account. Also, we plan to transfer the same personalization setting to evolving data, e.g. name changing biological data.

The remainder is organized as follows. Section 2 reviews related work. In Section 3 we present background information and our method's intuition. We describe our proposed framework in Section 4. Section 5 reports shortly on the empirical evaluation of our methods and Section 6 concludes.

2. RELATED WORK

In [9] the author proposes a topic-based refinement of the PageRank algorithm that allows the offline computation of a fixed number of PageRank vectors corresponding to certain topic categories. The final result is a weighted combination of these vectors, with weights depending on the similarity of the query and the respective topic. In [17] the authors utilize concept hierarchies, like ODP, to categorize queries and to generate user profiles. Query results are re-ranked based on those profiles using collaborative filtering techniques. By contrast, our method does not rely on user profiles and is independent of static topic hierarchies.

Another prominent strand of research is based on exploiting historic user feedback. The impact of short-term versus long-term histories has been studied by [19, 20] while [5, 18] aim at capturing the context of the users, for instance by taking documents on the virtual desktop into account. The resulting models are essentially user profiles that are used to expand future queries and to refine the retrieved results. Compared to our method, these approaches focus on content similarity and do not exploit collaborative user data.

Many approaches incorporate state-of-the-art machine learning techniques to improve ranking results. [4] study modifications of ranking support vector machines to reduce the error on top-ranks and to increase the importance of queries with only a few relevant documents in the training sample. In [14], the authors propose to learn multiple ranking functions for different ranks which are aggregated to induce the final ranking. By contrast, we propose to learn different ranking functions for different behaviors and intents. Furthermore, the above approaches do not take the inherent relations between queries/clickthrough data into account.

The closest work to ours is [3] who propose to learn multiple ranking models by clustering queries based on the topical information extracted by their results. They represent queries by aggregating feature vectors which are then clustered to obtain specific ranking models. The final ranking for new queries is being made by combining the models.

Their work differs in several aspects, the two main differences being as follows: Firstly, the method in [3] relies on pseudo feedback to extract the top results of each query and does not distinguish between positive and negative judgements. Secondly, the proposed approach computes the mean feature representation of the results for a given query and uses these averages to group queries. By contrast, we propose to cluster the ranking functions themselves. [25] apply K-Nearest Neighbor to find the closest training queries to a new query. Then, they train a model based on these queries to re-rank the results of the new one. Since this approach is inefficient to run online, they propose approximations that transfer some of the computation offline. However, this method, similar to [3], clusters query results on the feature space and not the search behaviors, as we propose.

3. BACKGROUND/METHOD INTUITION

In this section, we first present some background information on the Ranking SVM model. Then, we discuss how to exploit its properties to identify search behaviors.

3.1 Preliminaries

Let S be a training dataset consisting of queries, their results, and relevance judgments. Without loss of generality, we consider here 3 ranks of relevance judgments: $r = 0$ (irrelevant), $r = 1$ (partially relevant) and $r = 2$ (relevant). Let, also, $\mathbf{X}\epsilon\mathbb{R}^d$ a feature space of dimensionality d. Features in this space describe the query results and their matching quality with the corresponding query. Those features may be content-based (describing textual similarity between a query and its result [24]). Some features may involve hyperlink information (i.e., pagerank values), or specific information about the results such as the domain of the url or the rank of the result in several search engines [11]. Also, features may incorporate statistical information on user behaviour, e.g., deviation from average page viewing time [1].

So, our training dataset is of the form $S = \{(\mathbf{x}_1, r_1), (\mathbf{x}_2, r_2), ..., (\mathbf{x}_n, r_n)\}$, where $\mathbf{x}_i = (x_{1i}, x_{2i}, ..., x_{di})$ is the feature vector that characterizes a query-result pair, r_i is the relevance judgment (rank) for the specific result i and n is the number of query-result pairs in the dataset. Ranking SVM tries to find a ranking function $g(\mathbf{x}_i)$ that satisfies:

$$\mathbf{x}_i \succ \mathbf{x}_j \Leftrightarrow g(\mathbf{x}_i) > g(\mathbf{x}_j) \quad (1)$$

where $\mathbf{x}_i \succ \mathbf{x}_j$ denotes that result i has higher rank than result j, i.e. $r_i > r_j$.

Considering function g as a linear function of $\mathbf{x}_i$ then $g(\mathbf{x}_i) = \langle \mathbf{w} \cdot \mathbf{x}_i \rangle$. Thus, Equation 1 is re-written as follows:

$$\mathbf{x}_i \succ \mathbf{x}_j \Leftrightarrow \langle \mathbf{w}, \mathbf{x}_i - \mathbf{x}_j \rangle > 0 \quad (2)$$

Considering every result pair from the training dataset for which $\mathbf{x}_i \succ \mathbf{x}_j$ and introducing non-negative slack variables for misranked results, the problem is formalized as an optimization problem:

$$\begin{aligned} &\text{minimize: } \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum \xi_{ij} \\ &\text{subject to: } \langle \mathbf{w}, \mathbf{x}_i - \mathbf{x}_j \rangle > 1 - \xi_{ij}, \forall \mathbf{x}_i \succ \mathbf{x}_j, \xi_{ij} > 0 \end{aligned} \quad (3)$$

where $\|\mathbf{w}\|$ is the norm of $\mathbf{w}$. Note that $\mathbf{w}$ is a weight vector that quantifies how important is each feature for a specific training on a specific dataset. This vector is normal to hyperplanes (defined by the points b_{pq} where they cut through the normal vector) that separate results of different ranks, as shown in Figure 2. In Equation 3, the first term $\frac{1}{2}\|\mathbf{w}\|^2$ is related to the distance of the *correctly* ranked results that are closest to the hyperplane. The second term $\sum \xi_{ij}$ relates to the error introduced by the *misranked* results. For example, assuming that triangles should be placed (ranked) on the right of the hyperplane defined by b_{11} and the circles on the left, then: (a) $\frac{1}{2}\|\mathbf{w}_1\|^2$ is related to the distance of b_{11} from its closest rightmost triangles and from its closest leftmost circles and (b) $\sum \xi_{ij}$ is related to the distance of b_{11} from its lefttmost triangles and from its rightmost circles. Finally, λ determines the importance of each of the two terms in the training process.

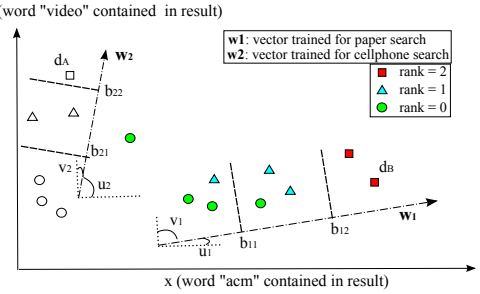

Figure 2: Trained weight vectors on feature space

3.2 Search Behavior and Ranking SVM

The example of Figure 2 gives a geometric interpretation of the model. For simplicity, consider that the feature vectors of the training data include only two features. Let feature x be the frequency of the word "acm" in the result text and feature y be the frequency of the word "video" in the result text. The shaded shapes represent results related to paper searches, while the non-shaded shapes represent results related to cellphone searches. Squares correspond to strongly relevant results, triangles to partially relevant results and circles to irrelevant results.

In this feature space, we train two ranking models expressed by weight vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. These vectors correspond to searches for papers and searches for cellphones, respectively. The direction of each $\mathbf{w}_i$, i.e., the angle between the vector and one of the axes, indicates how important each training feature is for the process of result ranking. For example, the angle u_1 between vector $\mathbf{w}_1$ and x-axis is smaller than the angle v_1 between the vector and y-axis. This means that a change in the value of feature x (frequency of word "acm" in results) is more probable to induce a change in the result's rank than a change in the value of feature y (frequency of word "cellphone" in results). So, for the particular training performed on clickthrough data from paper searches, feature x is more important than y when ranking query results. The opposite stands when training a vector $\mathbf{w}_2$ on cellphone searches: feature y is more important than x, as shown by the direction of $\mathbf{w}_2$.

The above example describes two different search behaviors, that is, specific search patterns followed by users for specific categories of searches. We can see that search behaviors are not expressed in terms of content, but through the feature space $\mathbf{X}\epsilon R^d$ selected to represent the clickthrough data (query results and their ranks/relevance judgments). So, we can capture and exploit such search behaviors by utilizing the distribution of the training clickthrough data in the feature space. Next section describes our framework.

4. RANKING MODELS FOR USER INTENT

In this section we present our main contribution, ranking models for user intent. Our work consists in three continuous approaches that revolve around the idea of capturing search behaviors/intents and training ranking models on them. Due to lack of space, we shortly describe the first two and then we describe in detail the latest one.

In [26] we followed a first cut approach to the problem, where we assumed that clustering historic queries **on content similarity** can capture search behaviors. To this end, our method trained separate ranking functions, each one using clickthrough data from one cluster. Then, each new query was matched content-wise on the clusters and its results were reranked using the respective ranking functions, weighted according to their cluster's similarity to the query.

In [27] we exploited the geometric characteristics of the Ranking SVM model in order to capture search behaviors. Specifically, we approximated the weight vectors to be trained by the models and we performed clustering on these approximations. The rest of the process is similar to [26], in terms of per cluster ranking function training and reranking. This method was, essentially, the precursor of the current method, that we present next ([28]).

4.1 Method Analysis

In a nutshell, we aim at learning ranking functions for *similar* queries, where similar refers to the latent user intent. Figure 3 shows a simple two-dimensional visualization of the problem setting, focusing on pdf (dimension x_1) and video (dimension x_2) results. Different queries (e.g., *racing cars videos*, *web search*) are visualized by relevant clicked (red squares) and not clicked results (green circles) documents. The task is to cluster the queries so that similar intents are close with respect to some distance measure in the feature space and, then, to train one ranking function per cluster.

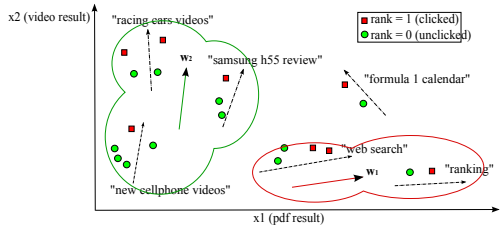

Figure 3: Visualization of the problem setting.

Since there is no ground-truth for the intrinsic clustering, the respective error of the ranking functions serves as a makeshift for the missing performance measure at the clustering stage. That is, if the error-rate of a ranking function is high, the queries in the respective cluster are too diverse to allow for a good fit; the goal is therefore to find a grouping of the queries such that the ranking models are well adapted. Thus, a natural approach is to jointly optimize the clustering *and* the ranking models.

Let K be the number of desired clusters. We intend to find (i) K ranking models $\vec{w}_1, \ldots, \vec{w}_K$, one for each cluster, and (ii) find a clustering $\vec{c}_1, \ldots, \vec{c}_K$ with $c_{kj} = 1$ if query q_j belongs to cluster k and $c_{kj} = 0$ otherwise, that gives rise to an optimal fit of the ranking models. The following optimization problem realizes this task straight forwardly:

$$\begin{aligned}
\min_{\vec{w}_k, \vec{c}_k, \xi_{ij}} \quad & \sum_{k=1}^{K} \left[\|\vec{w}_k\|^2 + \lambda_k \sum_{\ell=1}^{n} c_{k\ell} \sum_{(i,j)\in\mathcal{P}_{q_\ell}} \xi_{ij}^k \right] \\
\text{s.t.} \quad & \forall k, \forall (i,j) \in \mathcal{P}(k) : \langle \vec{w}_k, x_i \rangle \geq \langle \vec{w}_k, x_j \rangle + 1 - \xi_{ij}^k \\
& \forall k, \forall (i,j) \in \mathcal{P}(k) : \xi_{ij}^k \geq 0 \\
& \forall i, j, \ell : \; c_{ki} c_{kj} + c_{ki} c_{k\ell} \leq c_{kj} c_{k\ell} + 1 \\
& \forall k, \forall j : \; c_{kj} \in \{0, 1\}
\end{aligned} \tag{4}$$

where we defined $\mathcal{P}(k) = \bigcup_{j:c_{kj}=1} \mathcal{P}_{q_j}$ as the union of all members of cluster k, and trade-off parameters $\lambda_k > 0$.

The above optimization problem suffers from major drawbacks. Firstly, the optimization interweaves real and integer variables; that is, directly solving the mixed-integer program is expensive and one usually resorts to relaxing the binary variables to the interval $[0, 1]$ to obtain an approximate solution. Secondly and more severely, the number of triangle inequalities guaranteeing a proper clustering in Eq. (4) is cubic in the number of queries and renders the optimization infeasible at larger scales. Next, we present an efficient approximation of the problem.

4.2 Learning to Rank User Intent

We now present a sequential model that approximates the infeasible optimization problem and that can be solved efficiently on large scales. The novel approach consists of three stages and generates the desired ranking models for each cluster of queries: Firstly, we learn a ranking function for every query. Secondly, these ranking functions are clustered, and thirdly, we learn a ranking function for each cluster using the original queries and documents.

4.2.1 Ranking Models for Queries

The initial step of the approximation consists in learning a ranking model for every query. To this end we solve the standard Ranking SVM for every query and the respective preference relations assembled from the click data:

$$\begin{aligned}
\min_{\vec{w}_\ell, \xi_{\ell ij} \geq 0} \quad & \langle \vec{w}_\ell, \vec{w}_\ell \rangle + \lambda \sum_{ij} \xi_{\ell ij} \\
\text{s.t.} \quad & \forall (i,j) \in \mathcal{P}_{q_\ell} : \langle \vec{w}_\ell, x_i \rangle \geq \langle \vec{w}_\ell, x_j \rangle + 1 - \xi_{\ell ij}.
\end{aligned}$$

In general, the trade-off parameter λ (Eq. 3) needs to be set appropriately to obtain optimally adapted models. In our large-scale experiments, tuning the parameters manually or deploying model selection techniques like cross-validation is not feasible due to the large amount of data. Anecdotal evidence, however, shows that for binary representations and features in the interval $[0, 1]$, values around $\lambda \approx 1$ are often a reasonable choice. We thus use $\lambda = 1$ for the initial Ranking SVM models and note that there is potentially room for improvement. The result of this step is n ranking functions $\vec{w}_1, \ldots, \vec{w}_n$, one for each query. The training features we applied are presented in Table 1.

4.2.2 Clustering Ranking Functions

The goal of the second step of our approach is to group similar ranking models together as they capture similar intents. As the absolute locations of the $\vec{w}_i$ are negligible and only the direction of the vectors is of interest, the ranking functions are ℓ_2-normalized by $\vec{w} \leftarrow \vec{w}/\|\vec{w}\|$ so that they lie on the unit hyperball. The similarity of two ranking functions $\vec{w}$ and $\vec{w}'$ can now be measured by their cosine

Table 1: Feature categories

	Textual similarity features
4	Sum of TFs of query terms in result title—URL—text—all
4	Lucene score between query and result title—URL—text—all
	Result characteristics features
1	Result initial rank
4	Number of words in result title—url—text—all
1	Result URL length in characters
72	Result URL domain (boolean values)
83	Popular sites the result might belong to (boolean)
200	Top most frequent urls in the dataset
	Result special words features
10	Special words in result URL ("forum", "pdf", etc.)
10	Result site category (news, search, blog etc)
200	Top most frequent words in the dataset

which reduces to the inner product for normalized vectors, $\cos(\vec{w}, \vec{w}') = \langle \vec{w}, \vec{w}' \rangle$. Unit vectors are usually modeled by a von Mises-Fisher distribution [2], given by $p(\vec{x}|\vec{\mu}, \kappa) = Z_d(\kappa) \exp\{\kappa \langle \vec{\mu}, \vec{x} \rangle\}$ where $\|\vec{\mu} = 1\|$ and $\kappa \geq 0$ and $d \geq 2$ and partition function $Z_d(\kappa) = \kappa^{d/2-1}/(2\pi)^{d/2} I_{d/2-1}(\kappa)$ where $I_r(\cdot)$ denotes the modified Bessel function of the first kind and order r. Applied to the n ranking functions $\vec{w}_1, \ldots, \vec{w}_n$, a mixture model of von Mises-Fisher distributions with K components (clusters) has the density

$$f(\vec{w}_i|\vec{\mu}_1, \ldots, \vec{\mu}_K, \vec{\kappa}) = \sum_{i=1}^{n} \alpha_{c_i} p(\vec{w}_i|\vec{\mu}_{c_i}, \kappa_{c_i})$$

with mixing parameters α_i with $0 \leq \alpha_i \leq 1$ and $\sum \alpha_i = 1$. The latent variables $c_i \in \{1, \ldots, K\}$ indicate the generating components for the $\vec{w}_i$; that is, $c_i = k$ indicates that the ranking function $\vec{w}_i$ is sampled (generated) from the k-th component $p(\cdot|\vec{\mu}_k, \kappa_k)$. If the latent variables were known, finding maximum likelihood estimates for the parameters $\vec{\mu}_1, \ldots, \vec{\mu}_k$ and $\kappa_1, \ldots, \kappa_k$ would be trivial. Since this is not the case, we resort to a constrained Expectation Maximization approach to jointly optimize the log-likelihood.

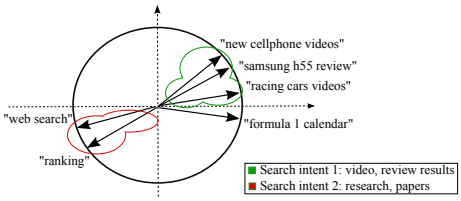

Figure 4: Query-specific models on the unit sphere.

4.2.3 *Ranking Models for Clusters*

Given the clustering induced by the latent variables c_i of the previous section, we now learn a ranking function for each cluster. The approach is similar to learning the initial ranking models for the queries, however, this time, all queries in the cluster have to be taken into account. The optimization for the k-th cluster can again be solved with the Ranking SVM and is given by:

$$\min_{\vec{w}_k, \xi_{ij} \geq 0} \quad \langle \vec{w}_k, \vec{w}_k \rangle + \lambda \sum_{ij} \xi_{ij}$$

$$\text{s.t.} \quad \forall (i,j) \in \bigcup_{\ell : c_\ell = k} \mathcal{P}_{q_\ell} : \langle \vec{w}_k, x_i \rangle \geq \langle \vec{w}_k, x_j \rangle + 1 - \xi_{ij}.$$

4.3 Reranking

Once the ranking functions are adapted to the clusters, our method can be deployed to rerank retrieved documents for new queries. Our approach aims at diversifying possible intents as the same query might end up in more than just one cluster, for instance if users clicked on different media types (e.g., videos, pdfs, etc.). Thus, the goal is to map a new query to the clustering and combine the respective ranking functions of the top matching clusters.

To this end, we represent historic queries together with their positively judged results as pseudo documents which are indexed and made searchable by a search engine. In our implementation we used the Lucene[1] IR engine, however, other choices are straightforward. Given a new query q, the Lucene scoring function is used to obtain historic queries which are similar to q.

We select the top-u most similar historic queries and the clusters they belong to. By doing so, we compute a weighted mapping of the new query to the clustering as follows. Let v_j, $1 \leq j \leq u$, be the scores for the top-u historic queries q_j, these are ℓ_1-normalized and translated into cluster-scores s_k, $1 \leq k \leq K$, such that $s_{qk} = \sum_{j:c_j=k} v_j / \sum_{i=1}^{u} v_i$, where the c_j are the latent cluster memberships. That is, if a cluster occurs more than once, the respective scores are aggregated. Due to the normalization, the scores s_{qk} act like probabilities, quantifying the likelihood that cluster k contains the intent expressed by query q.

Finally, the ranking of the documents for the query q is assembled from the clustering by weighting the contribution of each cluster k by its score s_{qk}. Let r_{kj} denote the ranking of the j-th document by the ranking function of cluster k, the final ranking score is given by linearly weighting the cluster rankings r_{kj} with the cluster importance s_{qk} for query q:

$$score(q, j) = \sum_{k=1}^{K} s_{qk} r_{qkj}.$$

5. EVALUATION

We briefly report on the experimental results of our current method, described above. More information can be found in [26, 27, 28]. We have sampled, from the Yahoo! query log, 76,037 queries posed by 453 distinct users. We split the obtained data, that is query and top-10 results, chronologically into 30,053 (40%) queries for training and 45,984 (60%) queries for test set. Ground-truth is given by user clicks in terms of relevance judgments as described in [11, 15]. This gives us a total of 96,030 relevance judgments for the training dataset and 144,021 for the test set, averaging to about 3.2 relevance judgments per query.

5.1 Baselines

We compare our method, denoted as *Intent* with four alternative approaches for re-ranking search results: Firstly, we deploy a single ranking SVM (*Single*) for all users which is trained on all available training data and used to rank the documents for the test queries. Secondly, we train an SVM for every user (*User*) to capture state-of-the-art personalization approaches. According to [19] we expect the *User* baseline to perform best while the *Single* baseline is expected to be too simple to capture diverse behaviors.

Furthermore, we apply *Content-1* which clusters queries in the training set based on their content similarity and learns a RSVM for each cluster which are finally combined to re-rank documents for the test queries. Note that, except for the clustering, the processing pipeline remains the same

[1] http://lucene.apache.org/

as ours; at the clustering stage, queries are grouped based on their textual similarity including text from their positive results (clicked documents). Finally, we apply a variant of topical RankSVMs [3] (*Content-2*). The document representation is extended by incorporating means and variances as dimensions for each feature; the new representation is computed by using the top-5 results per query. Note however that this baseline is not identical to [3] in the sense that we use the standard RSVM for the optimization problems.

5.2 Ranking Performance

Table 2 presents MAP results. Unsurprisingly, learning user specific models performs best, achieving about 14% precision increase compared to the a single model that serves everyone. The setting resembles an ideal scenario and the baselines *Single* and *User* constitute the lower and upper performance bound, respectively. Note that a real-world deployment of personalized user models would require perfect disambiguation of users which is still an open problem.

By contrast, *Content-1*, *Content-2*, and *Intent* are user independent and form groups of similar content or intent, respectively. In that sense, they constitute realizable approaches. However, they differ significantly in terms of predictive performance: *Content-2* is the weakest method although it still increases the performance over the *Single* baseline by 3.5%. *Content-1* allows for improvements about 5.5%, while *Intent* achieves the highest increase, by 6.3%.

Table 2: Mean average precision.

Method	MAP	Increase
Single	0.709	-
User	0.806	13.7%
Content-1	0.748	5.5%
Content-2	0.734	3.5%
Intent	0.754	6.3%

At first sight our method seems to be outperformed by a personalized solution. However, the latter is not always applicable. Consider, e.g., scenarios such as web search where only a fraction of all users are registered and can be disambiguated only after the login. Including the personalized user model thus mirrors an ideal but unrealistic scenario. As an alternative for scenarios that do not allow personalized methods, we propose to deploy ranking models for user intent, since our method significantly increases MAP.

6. CONCLUSION

In this work, we propose a methodology for improving the quality of ranking functions for web search by capturing and exploiting latent search behaviors. The underlying idea grounds on the observation that search behavior is not necessarily content-dependent and we show that it can be used to train more effective ranking models.

Our method clusters ranking models trained on search queries and their results. The produced clusters represent implicit search behaviors and are used to train ranking models for user intent. The experimental evaluation demonstrates the effectiveness of our method compared to content-based baselines. Also, our method is generally deployable and does not rely on user disambiguation. It, thus, proves a valid alternative for scenarios in which personalized models cannot always be applied, such as web search.

There is room for improvements and extensions on several aspects of our method. Our future work involves experimenting on the clustering dimensions construction process, the clustering algorithm and metrics and on the query-cluster matching and weighting process. Also, we want to study how individual user search behaviors can be exploited in our framework, thus integrating our query-centric method with user-centric and content-centric approaches. Finally, after enhancing the current methods, we aim to deploy/adapt them to different search scenarios, such as keyword search on RDF graphs, where training and reranking should take into account the structure (relations between resources) and the schema of the data.

7. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proc. of the ACM SIGIR Conference*, 2006.

[2] A. Banerjee, I. Dhillon, j. Ghosh and S. Sra. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions. *Journal of Machine Learning*, 38(6):1345–1382, 2005.

[3] J. Bian, X. Li, F.-Li. Liu, Z. Zheng, and H. Zha. Ranking Specialization for Web Search: A Divide-and-Conquer Approach by Using Topical RankSVM. In *Proc. of the ACM WWW Conference*, 2010.

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proc. of the ACM SIGIR Conference*, 2006.

[5] P.-A. Chirita, C.-S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In *Proc. of the ACM CIKM Conference*, 2006.

[6] W. Chu, and S.-S. Keerthi. Support Vector Ordinal Regression. *Neural Computation*, 19:792–815, 2007.

[7] Z. Dou, R. Song, J.-R. Wen, and X. Yuan. Evaluating the Effectiveness of Personalized Web Search. *IEEE TKDE*, 21:1178–1190, 2008.

[8] S. Fox, K. Karnawat, M. Mydland, S. Dumais and T. White. Evaluating implicit measures to improve web search. *ACM TOIS*, 23(2):147–168, 2005.

[9] T.-H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the ACM WWW Conference*, 2002.

[10] R. Herbrich, T. Graepel and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers, MIT Press*, 2000.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of the ACM SIGKDD Conference*, 2002.

[12] J.-W. Kim, and K.-S. Candan. Skip-and-prune: cosine-based top-k query processing for efficient context-sensitive document retrieval. In *Proceedings of the ACM SIGMOD Conf.*, 2009.

[13] S. Pandey, S. Roy, C. O. J. Cho, and S. Chakrabarti. Shuffling a stacked deck: the case for partially randomized ranking of search engine results. In *Proceedings of the VLDB Conference*, 2005.

[14] T. Qin, X.-D. Zhang, D.-S. Wang, T.-Y. Liu, W. Lai, and H. Li. Ranking with multiple hyperplanes. In *Proceedings of the ACM SIGIR Conference*, 2007.

[15] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *Proceedings of the ACM SIGKDD Conference*, 2005.

[16] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proc. of the ACM SIGKDD Conference*, 2007.

[17] U. Rohini and V. Ambati. Improving Re-ranking of Search Results Using Collaborative Filtering. *Information Retrieval Technology, AIRS*, 2006.

[18] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the ACM SIGIR Conference*, 2005.

[19] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the ACM WWW Conf.*, 2004.

[20] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the ACM SIGKDD Conference*, 2006.

[21] J. Teevan, S.-T. Dumais, and D.-J. Liebling. To Personalize or Not to Personalize: Modeling Queries with Variation in User Intent. In *Proceedings of the ACM SIGIR Conference*, 2008.

[22] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings of the ACM CIKM Conference*, 2004.

[23] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the ACM SIGIR Conference*, 2007.

[24] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Information Retrieval Journal*, 2010.

[25] X. Geng, T.-Y. Liu, T. Qin, A. Arnold, H. Li, and H.-Y. Shum. Query dependent ranking using K-nearest neighbor. In *Proceedings of the ACM SIGIR Conference*, 2008.

[26] G. Giannopoulos, T. Dalamagas and T. Sellis. Collaborative Ranking Function Training for Web Search Personalization. In *Proceedings of the PersDB Workshop*, 2009.

[27] G. Giannopoulos, T. Dalamagas, and T. Sellis. Search behavior-driven training for result re-ranking. In *Proceedings of the TPDL Conference*, 2011.

[28] G. Giannopoulos, U. Brefeld, T. Dalamagas, and T. Sellis. Learning to rank user intent. In *Proceedings of the CIKM Conference*, 2011.