Learning to Rank User Intent

Giorgos Giannopoulos[*]
NTU Athens -
IMIS, "Athena" R.C.
Greece
giann@dblab.ece.ntua.gr

Ulf Brefeld
Yahoo! Research
Barcelona, Spain
brefeld@yahoo-inc.com

Theodore Dalamagas
IMIS, "Athena" R.C.
Greece
dalamag@imis.athena-innovation.gr

Timos Sellis
NTU Athens -
IMIS, "Athena" R.C.
Greece
timos@imis.athena-innovation.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. We take a different approach and study ranking models for user intent. We exploit user feedback in terms of click data to cluster ranking models for historic queries according to user behavior and intent. Each cluster is finally represented by a single ranking model that captures the contained search interests 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. Empirical evidence shows that our approach significantly outperforms baseline approaches on a large corporate query log.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval – Relevance feedback, Search process, Clustering

General Terms

Algorithms, Experimentation, Measurement

Keywords

Search engine, ranking, training, clickthrough data, relevance judgement, clustering, search behavior

[*]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.

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 and filter 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- and long-term search histories [21, 22], context [14, 21], query categories [8, 24], and search behavior and feedback [1, 9, 12, 16] have been studied. Additionally, collaborative filtering techniques for personalized search [22] and learning to rank-based approaches [1, 6, 12, 17, 19, 26] also proved effective in many scenarios. Many of the above techniques are also applicable to registered users of search engines, however, 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 paper, we study an orthogonal approach to reranking 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. The user behavior is directly observable by user feedback in form of clicks on the result page and allows to reason about the intent of the users. The intent therefore acts like an unobserved, latent variable and is (partially) captured by user behavior.

Consider a user who issues a query for a new mobile phone. Her search history so far contains only unrelated queries. A personalized model would have to resort to the average user model for processing the query and possibly return text documents about phones. By contrast, our approach does not rely on user-specific models but aims at capturing the user intent by grouping queries entailing similar behavior. The results proposed to the user thus consist of different media types (e.g. reviews, videos, etc) that have been associated with mobile phones in the past. In other words, our system re-ranks the retrieved results, so that they represent the broad spectrum of user behavior for a given query.

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. Optimally, the clustering and the ranking models are optimized jointly to capture interdependencies between the tasks. The corresponding optimization problem however turns out to be a mixed-integer problem with cubic constraints in the number of queries and and renders large-scale deployment infeasible. We therefore present an approximation that consists of three stages: 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.

Empirically, we observe our approach to capture user intent better than baseline methods on a large sample from the Yahoo! query log. Our method achieves higher precision values on top-ranks compared to content-based baselines. Additionally, the underlying clustering is observed to effectively group queries with similar intents together while content-based baselines do not exhibit interpretable clusterings.

The remainder is organized as follows. Section 2 reviews related work. We present our main contribution, the joint optimization problem and its approximation, in Section 3. Section 4 reports on the empirical evaluation and Section 5 concludes.

2. RELATED WORK

In [10] 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, where weights are proportional to the similarity of the query and the respective topic. In [20] the authors utilize concept hierarchies, like ODP[1], to categorize queries and to generate user profiles. Query results are reranked 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 [22, 23] while [5, 21] 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 [17], 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 behavior and intents. Furthermore, the above approaches do not take the inherent relations between queries and their 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.

Finally, clustering methods are studied in combination with learning to rank strategies. [15] propose to cluster results to discard probably redundant examples from a large training sample to render the resulting optimization feasible, while [7] cluster personalized ranking functions to group users for recommendation purposes.

3. RANKING MODELS FOR USER INTENT

In this section we present our main contribution, ranking models for user intent. The following section introduces the problem setting and notation. Section 3.2 presents a joint optimization problem that directly solves the problem in theory but is infeasible in practice. In Section 3.3 we devise an efficient approximation that can be solved on large scales. Section 3.4 details the application of the model for new queries at execution time.

3.1 Preliminaries

We are given n historic queries $q_1, \ldots, q_n$ and their top-m retrieved documents $(x_1^{(q)}, y_1^{(q)}), \ldots, (x_m^{(q)}, y_m^{(q)})$ where $y_j^{(q)} = 1$ if $x_j^{(q)}$ was clicked and 0 otherwise. The click feedback induces a partial ranking on the documents such that

$$x_i^{(q)} \text{ is preferred over } x_j^{(q)} \;\Leftrightarrow\; y_i^{(q)} > y_j^{(q)}$$

holds. We collect the preference relations for query q in the index set $\mathcal{P}_q = \{(i,j) : y_i^{(q)} > y_j^{(q)}\}$, see also [12, 18]. A ranking function $f : (q,x) \mapsto \mathbb{R}$ can now be adapted to the pairwise preferences $\mathcal{P} = \bigcup_q \mathcal{P}_q$. In this paper we focus on linear models of the form $f(q,x) = \langle \vec{w}, \phi(q,x) \rangle$, where $\phi(q,x)$ denotes a joint embedding of query and document in some feature space. To avoid overloading the notation, we'll use $\phi(q,x) = x$ in the remainder and note that generalizations are straight forward, see for instance Table 2 for the features we used in the experiments. Following a large-margin approach leads to the optimization problem [13]

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

where $\lambda > 0$ determines the trade-off between margin maximization and error minimization. The latter is the sum of individual losses ξ_{ij} and constitutes an upper bound on the 0/1-loss of mistaken preference relations. The constraints enforce $\langle \vec{w}, x_i \rangle > \langle \vec{w}, x_j \rangle$ whenever possible and penalize violations thereof. Once optimal parameters $\vec{w}^*$ have been

[1]http://www.dmoz.org/

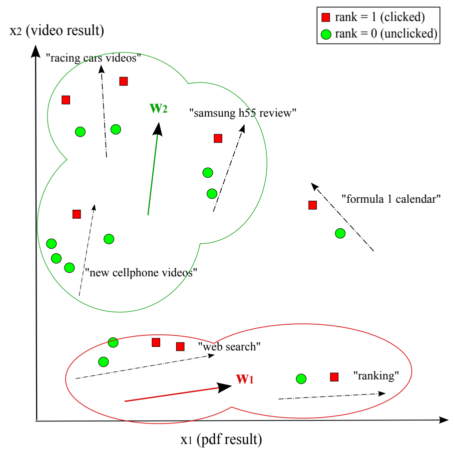

Figure 1: Visualization of the problem setting.

found, these are used as plug-in estimates to induce rankings of the documents for new queries.

3.2 Joint Optimization

In a nutshell, we aim at learning ranking functions for *similar* queries, where similar refers to the latent user intent. Figure 1 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 group the queries so that similar intents are close with respect to some distance measure in the feature space so that they are clustered together.

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,

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

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. (1) is cubic in the number of queries and renders the optimization infeasible at larger scales. We present an efficient approximation and propose a pipelined approach in the next section.

Table 1: Ranking Models for User Intent

Require: n queries q_j with preference relations $\mathcal{P}_{q_j}$

```
1: for $1 \leq j \leq n$ do
2:     learn ranking function $\vec{w}_j$ for $q_j$ using $\mathcal{P}_{q_j}$
3: end for
4: cluster $w_1, \ldots, w_n$
5: for $1 \leq k \leq K$ do
6:     learn ranking function $\vec{w}_k$ using $\bigcup_{j:c_j=k} \mathcal{P}_{q_j}$
7: end for
```

Ensure: ranking models $\vec{w}_1, \ldots, \vec{w}_K$

3.3 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. The algorithm in pseudo-code is depicted in Table 1.

3.3.1 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. Analogously to Section 3.1, the ℓ-th optimization problem can either be solved by quadratic programming or online gradient-based approaches [12, 18, 13] and is given by

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

In general, the trade-off parameter λ 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.

3.3.2 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

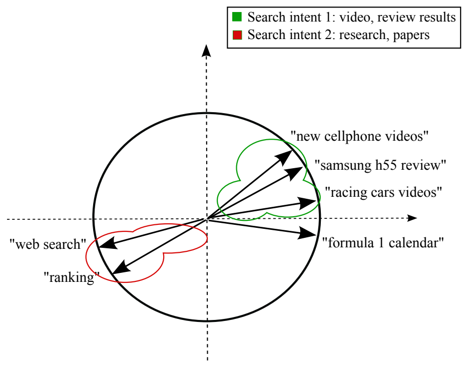

Figure 2: Query-specific models on the unit sphere.

functions $\vec{w}$ and $\vec{w}'$ can now be measured by their cosine 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 $p(\cdot|\vec{\mu}_k, \kappa_k)$.[2] 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.

3.3.3 *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

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

3.4 Application

Once the ranking functions are adapted to the clusters, our method can be deployed to re-rank 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.

[2]Note that the variables $\vec{c}_k$ in Section 3.2 are analogous binary encodings of the latent variables c_i. That is, if the j-th query is in the k-th cluster, we have $c_{kj} = 1$ and $c_j = k$, respectively. We overloaded the notation to indicate that both represent the actual clustering.

Table 2: 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

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[3] IR engine, however, other choices are straight forward. 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,

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

4. EMPIRICAL EVALUATION

For the experimental evaluation, we sample queries from the Yahoo! query log. From the sample, we discard queries with less than 5 results, queries without clicks, and queries from users with less than 100 searches. This leaves us with 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 [12, 18] as follows: If a document x_i has been clicked, the relevance judgment equals $y_i = 1$. Unclicked documents that are higher ranked than clicked results receive a relevance judgment of $y_j = 0$ which is also used for unclicked results occuring right after a clicked result. This

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

Table 3: **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%

process results in a total of 96,030 relevance judgments for the training dataset and 144,021 for the test set. This gives an average of about 3.2 relevance judgments per query on the data. The query-result pairs are represented by feature vectors. The respective features are depicted in Table 2.

4.1 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 [22], short- and long-term search histories are well captured by personalized, user-specific models and we thus expect the *User* baseline to perform best while the *Single* baseline is expected to be too simple to capture the diverse behavior in the data.

Furthermore, we apply *Content-1* which clusters queries in the training set based on their content similarity and learns a ranking SVM for each cluster which are finally combined to re-rank documents for the test queries. Note that – except for the clustering – the processing pipeline is exactly the same as in our method; at the clustering stage, queries are grouped based on their textual similarity including text from their positive results (the 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 of each query. Note however that this baseline is not identical to [3] in the sense that we use the standard ranking SVM for solving the optimization problems.

4.2 Ranking Performance

The first experiment aims at measuring the performance of the algorithms in a static environment. We use the complete training set for the learning processes and all available test queries for evaluation. We report on MAP, Precision@n, and NDCG@n.

Results for MAP are shown in Table 3. 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 expected lower and upper bound on the performance, respectively. Note that a real-world deployment of the personalized user model 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. Among these three, *Content-2* is the

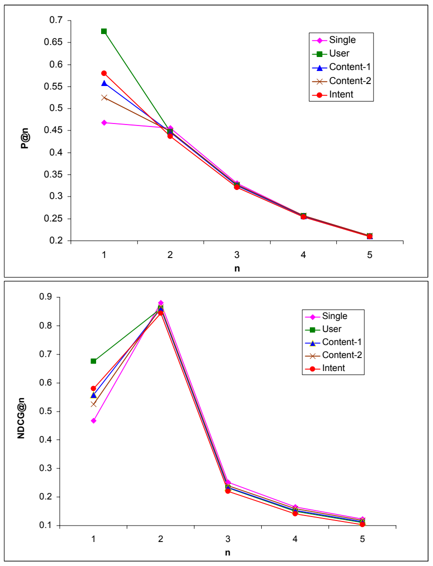

Figure 3: Precision@k and NDCG@k.

weakest method although it still increases the performance over the *Single* baseline by 3.5%. *Content-1* allows for improvements about 5.5% and *Intent* even by 6.3%.

A similar picture is drawn by the precision at n scores that are displayed in Figure 3 (top). The methods are indifferent for $n > 1$ due to the relatively small number of relevance judgments (on average 3.2 per query). More specifically, for the 45,984 test queries, there are 51,089 positive relevance judgements (user clicks) which translate to about 1.1 clicks per query on average. At P@1, however, we observe significant differences in performance that confirm the previous findings. *Single* and *User* establish lower and upper bounds and *Intent* performs better than *Content-1/2*. Figure 3 (bottom) corroborates the observations for NDCG@n.

4.3 Cluster Analysis

To shed light on the nature of intent- and content-based methods, we analyze and compare respective clusterings for *Intent*, *Content-1*, and *Content-2* in Table 4. We picked clusters with queries for which the respective methods perform well.

The qualitative results are as follows. Firstly the approaches differ significantly in the amount of clusters, where the optimal number of clusters is determined by model selection for each method. While the content-based methods generate between 20 (*Content-1*) and 32 (*Content-2*) clusters, the solution of *Intent* consists of 75 distinct clusters. Though clusterings of this size are generally difficult to interpret, the numbers already indicate that the solution found by *Intent* is more specialized than the content-based ones due to the, on average, smaller clusters. In fact, it turns out that the *Intent* performs well in many specific information needs as Table 4 (left) shows. The first set of queries corresponds to a cluster that contains information needs in textual form, perhaps enriched with pictures while the second group contains specific questions which are probably best answered by appropriate text documents, too.

By contrast, Table 4 (center and right) show exemplary clusters for the two content-based methods. The former shows two clusters for *Content-1*. While the top cluster

Table 4: Exemplary results of the clustering.

Intent	Content-1	Content-2
1968 yamaha trailmaster 100 yl2 value	austro diesel gmbh schwechat	mila kunis photo
spendor s3 5 system	skeleton reference of human muscle	marie osmond classical beauty doll margot
sonic video game 2011	double din dash facia for pt cruiser	mickey mouse pictures
85 mustang ignition module harness	keilwerth tenor ex90	batman action ↓gure power pack
owner of gold 39 s gym in wichita	seiko ladies watch bracelet elegant	lego star wars 2
72 chevy fuel tank swap	conn 37m tenor sax	↓ghter jets
artist lessons mountain painting	new jersey animal shelters	dental o¡ce for sale in california
who makes jet skis	best food to sell for pro↓t	barrio indios puerto rico house rentals
why does spray paint come o↑	fbi national academy 2010 boston	tv shows solar power
where can i buy centrum materna in us	passport renewal	logo design
why is the order of operations for algebra	oprah wearing philip stein watches	hotel dei mellini rome
shooting a wedding without a 'ash	top scottish baby names	gem kitchens and bath dublin

is similar to corresponding one of the *Intent*, the bottom is more or less a random collection of queries expressing a diverse set of information needs. Finally, the right column of Table 4 shows examples for well performing clusters for *Content-2*. The baseline exhibits typical content-based clusters formed by common tokens. The noisy membership can be explained by keywords which are central for the cluster and only occur on the result documents and not in the query.

4.4 Discussion

At first sight our method seems to be outperformed by a personalized solution. However, the latter is not always applicable. Consider, for instance, 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. Our method significantly increases MAP and also outperforms traditional content-based baselines for P@n and NDCG@n.

In our setting, the increase in P@n and NDCG@n performance is achieved by a significant increase in P@1, that is, *Intent* performs well in ranking relevant result on top. This observation is explained by the model itself: by grouping queries into clusters with similar intent, multiple ranking models are established, each one based on queries with similar user clicks in terms of the resulting types of documents. Results for new queries are re-ranked using the clustering; the final ranking score is computed by a linear mixture of relevant ranking functions. In case the textual matching is inaccurate, for instance because textual similarity does not necessarily imply similar search intentions, the final score diversifies the most likely intents and counterbalances possible errors at earlier stages.

5. CONCLUSION

In this paper, we presented a methodology for improving the quality of ranking functions for web search by capturing and exploiting latent search behavior. 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 behavior and are used to train ranking models for user intent. The experimental evaluation demonstrates the effectiveness of our method compared to traditional content-based baselines, leading to significant increases in MAP, P@1 and NDCG@1. An analysis of the resulting clusterings revealed that the novel method groups similar queries together while the content-based baselines suffer from noise that is incorporated by additional content from the documents. Although our approach cannot compete with personalized methods, we note that it is generally deployable and does not rely on user disambiguation. It thus proved a valid alternative for scenarios in which personalized models cannot be applied such as web search.

Acknowledgments

The authors wish to thank B. Barla Cambazoglu for helping us with data extraction and experimental evaluation.

6. 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 *Proceedings of the ACM CIKM Conference*, 2006.

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

[7] J. Diez, J. J. del Coz, O. Luaces, and A. Bahamonde. Clustering people according to their preference criteria. *Expert Systems with Applications: An International Journal*, 34:1274–1284, 2008.

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

[9] 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.

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

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

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

[13] T. Joachims. Training Linear SVMs in Linear Time. In *Proceedings of ACM SIGKDD Conference*, 2006.

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

[15] X. Li, N. Wang, and S.-Y. Li. A fast training algorithm for svm via clustering technique and gabriel graph. In *Proceedings of the International Conference on Intelligent Computing*, 2007.

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

[17] 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.

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

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

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

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

[22] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user pro↓le constructed without any e↑ort from users. In *Proceedings of the ACM WWW Conference*, 2004.

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

[24] 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.

[25] 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.

[26] 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.