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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

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 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. 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 opti-
mization 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 authors propose 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 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 [22, 23] while [5, 21] are given historic queries \(q_1, \ldots, q_n\) and their top-\(m\) retrieved documents \((x^{(q)}_1, y^{(q)}_1), \ldots, (x^{(q)}_m, y^{(q)}_m)\) where \(y^{(q)}_j = 1\) if \(x^{(q)}_j\) was clicked and 0 otherwise. The click feedback induces a partial ranking on the documents such that \(x^{(q)}_i\) is preferred over \(x^{(q)}_j \iff y^{(q)}_i > y^{(q)}_j\) holds. We collect the preference relations for query \(q\) in the index set \(\mathcal{P}_q = \{(i, j) : y^{(q)}_i > y^{(q)}_j\}\), 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 straightforward. Following a large-margin approach leads to the optimization problem [13]

\[
\min_{\vec{w}, \xi_{ij} \geq 0} \langle \vec{w}, \vec{w} \rangle + \lambda \sum_{ij} \xi_{ij}
\]

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

where \(\lambda > 0\) determines the trade-off between margin maximization and error minimization. The latter is the sum of individual losses \(\xi_{ij}\), 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

\[^1\text{http://www.dmoz.org/} \]
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 makeshift for the missing performance measure at the clustering, the respective error of the ranking functions serves as a plug-in estimate to induce ranking models.

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 makeshift for the missing performance measure at the clustering, the respective error of the ranking functions serves as a plug-in estimate to induce ranking models. This 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.

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.

### Table 1: Ranking Models for User Intent

<table>
<thead>
<tr>
<th>Require: $n$ queries $q_j$ with preference relations $P_{q_j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: for $1 \leq j \leq n$ do</td>
</tr>
<tr>
<td>2: learn ranking function $w_j$ for $q_j$ using $P_{q_j}$</td>
</tr>
<tr>
<td>3: end for</td>
</tr>
<tr>
<td>4: cluster $w_1, \ldots, w_n$</td>
</tr>
<tr>
<td>5: for $1 \leq k \leq K$ do</td>
</tr>
<tr>
<td>6: learn ranking function $w_k$ using $\bigcup_j c_{kj} = kP_{q_j}$</td>
</tr>
<tr>
<td>7: end for</td>
</tr>
</tbody>
</table>

Ensure: ranking models $w_1, \ldots, w_K$

The goal of the second step of our approach is to group similar ranking models together as they capture similar intents. The absolute locations of the $w_i$ are negligible and only the direction of the vectors is of interest, the ranking functions are $\ell_2$-normalized by $w \leftarrow w/\|w\|$ so that they lie on the unit hyperball. The similarity of two ranking models

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.

#### 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 $\ell$-th optimization problem can either be solved by quadratic programming or online gradient-based approaches [12, 18, 13] and is given by

$$
\min_{w_k, \xi} \sum_{k=1}^{K} \|\tilde{w}_k\|^2 + \lambda_k \sum_{(i,j) \in P_{q_j}} \xi_{ij}
$$

s.t. $\forall k, \forall (i,j) \in P(k): \langle \tilde{w}_k, x_i \rangle \geq \langle \tilde{w}_k, x_j \rangle + 1 - \xi_{ij}$

where we defined $P(k) = \bigcup_j c_{kj} = P_{q_j}$ 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 inequalities guaranteeing a proper clustering in Eq. (1) is
functions $\vec{w}$ and $\vec{w}'$ can now be measured by their cosine which reduces to the inner product for normalized vectors, $\cos(\vec{a}, \vec{a}') = \langle \vec{a}, \vec{a}' \rangle$. Unit vectors are usually modeled by a von Mises-Fisher distribution [2], given by $p(\vec{\mu}|\kappa) = Z_d(\kappa) \exp\{\kappa \langle \vec{\mu}, \vec{x} \rangle\}$ 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 $I_\cdot(\cdot)$ denotes the modified Bessel function of the first kind and order r. Applied to the n ranking functions with $\vec{w}_1, \ldots, \vec{w}_n$, a mixture model of von Mises-Fisher distributions with K components (clusters) has the density

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

with mixing parameters $\alpha_i$ 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 $p(\vec{w}_k|\vec{\mu}_k, \kappa_k).$ \(^2\) 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

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

s.t. $\forall(i,j) \in \bigcup_{k} P_{ij} : \langle \vec{w}_k, \vec{x}_i \rangle \geq \langle \vec{w}_k, \vec{x}_j \rangle + 1 - \xi_{ij}.$

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

\(^2\) Note that the variables $c_i$ 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 $c_{i,j} = 1$ and $c_{j,k} = 0$, respectively. We overloaded the notation to indicate that both represent the actual clustering.

### Table 2: Feature categories

<table>
<thead>
<tr>
<th>Feature category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual similarity features</td>
<td>For each query, score between query and result titles/abstacts</td>
</tr>
<tr>
<td>Result characteristics features</td>
<td>For each result, score between query and result title/abstact</td>
</tr>
<tr>
<td>Result metadata features</td>
<td>For each result, score between query and result URL metadata</td>
</tr>
<tr>
<td>Result meta data features</td>
<td>For each result, score between query and result URL meta data</td>
</tr>
<tr>
<td>Result special words features</td>
<td>For each result, score between query and result special words</td>
</tr>
<tr>
<td>Result domain category</td>
<td>For each result, score between query and result domain category</td>
</tr>
<tr>
<td>Top most frequent words in the dataset</td>
<td>Top most frequent words in the dataset</td>
</tr>
<tr>
<td>Top most frequent words in the dataset</td>
<td>Top most frequent words in the dataset</td>
</tr>
</tbody>
</table>

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 $y_i = 0$ which is also used for unclicked results occurring right after a clicked result. This

\(^3\) http://lucene.apache.org/
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 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 judgments (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>
<thead>
<tr>
<th>Method</th>
<th>MAP</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>0.709</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>0.806</td>
<td>13.7%</td>
</tr>
<tr>
<td>Content-1</td>
<td>0.748</td>
<td>5.5%</td>
</tr>
<tr>
<td>Content-2</td>
<td>0.734</td>
<td>3.5%</td>
</tr>
<tr>
<td>Intent</td>
<td>0.754</td>
<td>6.3%</td>
</tr>
</tbody>
</table>

Table 3: Mean average precision.

![Figure 3: Precision@k and NDCG@k.](image)
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


Table 4: Exemplary results of the clustering.