Iterative Refinement Methods for Enhanced Information Retrieval

Dong Zhou,[1,*] Mark Truran,[2] Jianxun Liu,[1] Wei Li,[3] Gareth Jones[3]
[1]Key Laboratory of Knowledge Processing and Networked Manufacturing & School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan 411201, People's Republic of China
[2]School of Computing, Teesside University, Middlesbrough TS1 3BA, Tees Valley, UK
[3]School of Computer Science, Dublin 9 City University, Dublin 9, Ireland

Information retrieval (IR) systems exploit relevant information when tailoring search results to individual information needs. However, current search experience becomes poor without considering similar queries entered by previous searchers. In the following paper, we discuss a solution to this problem, which combines collaborative filtering algorithms with traditional IR models to enable *EIR*. We also present various iterative refinement methods for improving the raw performance of this system. We validate our theories in an experiment using queries extracted from the click-through log of a commercial search engine. According to our results, an IR system employing iteratively refined, collaborative retrieval significantly outperforms various baseline retrieval models. © 2014 Wiley Periodicals, Inc.

1. INTRODUCTION

Information retrieval (IR) systems aim to satisfy a user's information needs by retrieving a set of potentially relevant documents from which they can extract the required information. In the majority of search engines, this process does *not* exploit the characteristics or known predilections of the user. As a result, identical queries will generate identical results, even if submitted by wildly dissimilar users. In contrast, personalised information retrieval (PIR) systems exploit user information and provide query-relevant search results that have been adapted to specific user needs.[1,2]

Such PIR systems are dependant upon the ready availability of *user information*. This information is pivotal when tailoring search engine results to the preferences of specific individuals. Unfortunately, user information *is not always available*. For example, Web search engine users are usually anonymous.

*Author to whom all correspondence should be addressed; e-mail: dongzhou1979 @hotmail.com.

View this article online at wileyonlinelibrary.com. • DOI 10.1002/int.21641

Furthermore, users may actively resist the creation of user profiles on the basis of privacy, security, or diversity.[3]

In this paper, we describe an approach to IR that is *not* dependent on the availability of specific user information. We call it enhanced information retrieval (EIR). The intuition behind this approach exploits a fundamental characteristic of search engine usage. Assume a system is confronted by a query q_a from a novel user. We have no user information, so straightforward personalization is not possible. However, *collaborative retrieval* is still viable, provided we have access to a large query log L and click-through record. We find queries in L similar to q_a and compile a list of documents that were clicked by users submitting those queries. We treat these click-through events as "ratings" that "similar" users have awarded. The underlying assumption of the approach is similar to that of the collaborative filtering (CF) methods: if a person p_1 has the same opinion as another person p_2 on one item (here refers to a document), p_1 is more likely to have p_2's opinion on a different item i than to have the opinion on i of a person chosen randomly. We combine this feedback with a set of results from a standard IR engine to improve the search experience on the basis of a single query.

Having outlined the core theory of EIR, we subsequently introduce three different methods for improving the raw performance of our system. These refinement methods suppose that the relevance of a document to a particular user query is mainly determined by two factors: (1) the highly ranked documents for a query are likely to be associated with and influenced by other documents rated by similar queries and (2) there exists various *mutually reinforcing* relationships within our framework that are susceptible to iterative, graph-based analysis. Our first method produces enhanced CF results through iterative updating. Our second method assumes that documents and queries in a enhanced IR system act to reinforce each another. The third method extends the first by regularizing the smoothness of document associations over the connected graph.

The remainder of this paper is organized as follows. In Section 2, we summarize related work from the fields of IR and collaborative filtering/ranking. In Section 3, we describe a CF-based implementation of EIR. Section 4 presents our refinement methods. Section 5 documents the experiments we used to evaluate our algorithms. Section 6 presents our results. Section 7 concludes the paper and proposes future work.

2. RELATED WORK

2.1. Personalized Information Retrieval

Our work relates to PIR in general. It is an extremely active research field.[4–6] To date, a significant amount of effort has been expended examining new sources of user information. Popular sources include query/click-through logs,[1,2] desktop documents,[7] user activity context,[8,9] Web browser logs, social media profiles, and e-mails.[10–12] Once harvested, user information is translated into an individual *profiles*.[12,13] Internally, user profiles employ various data structures to represent the captured data. Representational models employing feature vectors and semantic networks are common.[7,14] These models can be deployed at various points in the search process. They can be manipulated to expand the user's initial query, with the aim of retrieving more relevant results.[7,15] Alternatively, they can be used to rerank and/or filter a given set of search results.[16–18]

Distinct from PIR systems dependent upon individual user models, there is a class of retrieval engine that attempts an *aggregated form* of personalization.[19,20] Aggregate personalization involves the exploitation of collective usage information. A general representation of user behavior is captured by the search engine, e.g., query-document pairs with click-through frequencies. This representation is subsequently manipulated to adapt the search process for *all* users. A common assumption made by aggregate PIR systems is that frequently clicked documents should be assigned higher ranks.

2.2. Collaborative Filtering/Ranking

CF is a technique commonly used by commercial recommender systems. Recommender systems make predictions about the likelihood that a user u will like an item i. A prerequisite for this operation is a matrix-relating items to ratings.[21] These ratings are awarded by u and his/her peers (i.e., the user community). Assuming the availability of this matrix, the recommendation process works like this:

- Find the subset of all users who have awarded ratings to *other* items that agree with the ratings awarded by u
- Filter this subset to find the users who have already rated i.
- Calculate the mean rating awarded to i by these users.
- If the mean rating is positive, recommend i to u.

Given a large enough matrix, this calculation quickly becomes computationally expensive. A number of memory- and model-based algorithms are designed to optimize the process. Memory-based algorithms (e.g., item-based and user-based systems[22,23]) exploit the whole matrix when computing predictions. Generally, the predictions are calculated from the ratings of neighbors (i.e., users or items that are similar to the active user/item). In contrast, model-driven techniques make predictions based on user behavior models. The parameters of the models are estimated offline. Techniques exploiting singular value decomposition (SVD)[24] and probabilistic methods (e.g., latent class models[25]) are common in this context. All of CF algorithms experience difficulties when the matrix is sparsely populated. Accurately recommending products to new users (the "cold start" problem) can also be challenging.[26]

We were able to address both of these issues in our work (see Section 6.4). Even though our system relies upon CF algorithms, it still works even when the rating matrix is sparsely populated. Furthermore, "cold start" scenarios do not automatically lead to a drop in retrieval effectiveness.

CF has exploited ranking facilities to improve the system performance.[27–29] There are methods for improving model-based[27] and memory-based approaches.[28] Exploiting click-through logs has also been investigated.[29]

There exists some work on applying IR principles to the recommender systems or CF systems.[30,31] In these work, the authors exploited connections between IR and

CF and reformulated the CF problem as an IR task. They also developed techniques to provide CF data with a particular structure, to be able to use any IR weighting function with it. However, there is a lack of combining the CF and IR models to produce an enhanced IR method.[32,33]

3. ENHANCED INFORMATION RETRIEVAL METHOD

In this section, we explain our approach to EIR. As mentioned above, this approach is specifically designed for situations in which *no user information is available*. In these situations, straightforward retrieval will generate less accurate results. When our system encounters a user with no user profile, we *automatically construct* one from click-through frequencies and document ranking information opportunistically mined from a large, commercial search log. In doing so, we extend the work of Wei et al., who combined IR with domain-specific recommender models to personalize search, even in the absence of prior information about the user's knowledge of a domain.[34]

We now introduce the notations that will be used throughout this section. Our technique deals with a finite set of queries, $Q = \{q_a, q_1, q_2 \ldots q_m\}$, and a finite set of documents $D = \{d_1, d_2 \ldots d_n\}$ aggregated from documents retrieved by Q. Each query $q_i \in Q$ is associated with a *profile*, which consists of a set of documents retrieved by submitting that query to a standard IR engine (for novel users) or documents clicked by existing users, $D_q \subseteq D$. We also recorded the corresponding retrieval scores and/or ratings (see Table II). Note that we treat retrieval scores as CF *ratings* as well. These ratings, denoted R, will always correspond to real numbers. The first query we send to the IR system is denoted q_a. The subset of queries that have retrieved/clicked a certain document is defined as $Q_d \subseteq Q$.

Using the query profiles, we construct a rating matrix V. V will contain $|Q|$ rows and $|D|$ columns. Each element of V, $v_{qd} \in R \cup \varnothing$, denotes the rating given by query $q \in Q$ to document $d \in D$. A value of $\varnothing$ for v_{qd} indicates that query q has not retrieved document d yet. We process this matrix using a CF algorithm. The goal is to predict the value of v for documents that have not been retrieved/clicked. Let us denote the prediction for $d \in D$ by query $q \in Q$ as $p_{qd} \in R \cup \varnothing$ (p_{ad} for q_a). If our CF algorithm is not able to make this prediction, then we set $p_{qd} = \varnothing$. For later use, we define the subset of query ratings as $v_{q\cdot} = v_{qd} \in V/d \in D_q$, and the subset of document ratings as $v_{\cdot d} = v_{qd} \in V/q \in Q_d$. We also denote the query mean rating as $\overline{v_{q\cdot}}$ ($\overline{v_{a\cdot}}$ for q_a) and document mean rating as $\overline{v_{\cdot d}}$.

Our basic approach works as follows: Assume our EIR system has a novel user (no profile) who submits a query q_a. First, we get a set of results for this query from a standard IR engine. Next, we find a certain number (m) of queries similar to q_a in L such that $Q = \{q_1, q_2 \ldots q_m\}$ (see Algorithm 1 and Table I).

Having found m similar queries, we examine L to find the documents that were clicked on by the users who submitted those queries. We use this information to populate a matrix V (see Algorithm 2 and Table II). In the algorithm, we retrieve several documents lists for each query excluding the query q_a. Each list consists of documents clicked by the corresponding query (denoted as CF-RETRIEVE). For query

Algorithm 1 Finds similar queries $Q = q_1, q_2 \dots q_m$

```
Require: $q$ A TEST QUERY
Require: $L$ A SEARCH LOG
  Set $Q = \Phi$
  Tokenise $q$ into terms $Q = t_1, t_2 \dots t_o$
  for all $t_i \in q$ do
    $Q \leftarrow$ Find queries containing $t_i$
    for all $q' \in Q$ do
      $\overrightarrow{S_{q'}} \leftarrow$ SIMILARITY MEASURES $(q, q')$ #See Table I.
    end for
  end for
  $Q = q_1, q_2 \dots q_m \leftarrow$ sort $\overrightarrow{S_{q'}}$ in descending order
  return $Q$
```

Table I. Query similarity measurements. W_q and W_s are the sets of words in queries q and q'.

Dice	$2\left\|W_q \cap W_{q'}\right\| / \left(\left\|W_q\right\| + \left\|W_{q'}\right\|\right)$
Jaccard	$\left\|W_q \cap W_{q'}\right\| / \left\|W_q \cup W_{q'}\right\|$
Overlap	$\left\|W_q \cap W_{q'}\right\| / min\left(W_q, W_{q'}\right)$
Cosine	$\left\|W_q \cap W_{q'}\right\| / \sqrt{\left\|W_q \times W_{q'}\right\|}$
Matching	$\left\|W_q \cap W_{q'}\right\|$

q_a, we output first k documents of the IR ranked list (denoted as IR-RETRIEVE) in the document corpus C. We then aggregate the documents generated by all above lists into a single set of documents. To populate the rating matrix V, a rate will assign to each document according to their ranking position in the search logs and click position as recorded in the search logs (see Table II).

$$V = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ q_a & (\times) & (\times) & - & - & - & - \\ q_1 & - & \times & \times & - & - & - \\ q_2 & - & - & \times & - & \times & \times \\ q_3 & \times & \times & \times & \times & \times & \times \end{bmatrix}$$

The populated matrix maybe something shown above, q_1 has two entries indicating click-through events against $\{d_2, d_3\}$, q_2 has three click-through events and q_3 has 6. Note that the ratings against q_a are *pseudoratings* taken from the IR results. We take the top-k results from IR. Here, we assume that $k = 2$. The parameter m is set to 3.

Having built the rating matrix V, we predict the relevance p_{ad} of each document $d \in D$ to q_a using one of the common CF algorithms.[26] This procedure is described in Algorithm 3. In this algorithm, we iterate through all documents in D (excluding those documents $\{d_1, \dots, d_k\}$ which were retrieved using the query q_a) to produce a vector of predictions $\overrightarrow{p_{a\cdot}}$. In our experiment, we compare four state of the art CF algorithms as follows:

User-based:

$$p_{ad} = \overline{v_{a\cdot}} + \sigma_a \frac{\sum_{q \in neigh_a} \left[\left(\frac{V_{ad} - \overline{V_{a\cdot}}}{\sigma_q} \right) s(a,q) \right]}{\sum_{q \in neigh_a} s(a,q)}$$

Item-based:

$$P_{ad} = \frac{\sum_{d'} (s(d',d) v_{ad})}{\sum_{d'} |s(d',d)|}$$

SVD:

$$P_{ad} = \overline{V_{a\cdot}} + U_r \cdot \sqrt{S_r^T}(u) \cdot \sqrt{S_r} \cdot R_r^T(d)$$

SlopeOne:

$$p_{ad} = \frac{\sum_{d' \in D_q - \{d\}} \left(\sum_{x \in S_{dd'}} \frac{v_{xd} - v_{xd'}}{S_{dd'}} + v_{qd} \right) |S_{dd'}|}{\sum_{d' \in D_q - \{d\}} |S_{dd'}|}$$

Algorithm 2 Populates the rating matrix V

```
Require: $q$ A TEST QUERY
Require: $L$ A SEARCH LOG
Require: $C$ A DOCUMENT CORPUS
  $Q \leftarrow$ SIMILAR $(q, L)$ #See Algorithm 1
  for all $q_i \in Q$ do
    $D \leftarrow$ CF-RETRIEVE $(q_i, L)$ #ranked by CF scores
  end for
  $(d_1, \ldots, d_k) \leftarrow$ IR-RETRIEVE $(q, C)$
  $D \leftarrow D \cup (d_1, \ldots, d_k)$
  for all $q_i \in Q \cup q$ do
    for all $d_j \in D$ do
      $v_{ij} \leftarrow$ RATE $(q_i, d_j)$ #See Table II
      $V \leftarrow v_{ij}$
    end for
  end for
  return $V$
```

Table II. Three different methods for calculating a rating (see Algorithm 2). The function $max(click)$ returns the maximum number of clicks in the search logs for this query. The function $min(click)$ returns the minimum number of clicks in the search logs for this query. The function $similar()$ uses a query similarity measurement from Table I.

$RATE_1$	(rank/max(rank)) × (max(click) + min(click) − click)
$RATE_2$	(rank/max(rank)) × (max(click) + min(click) − click) × similar()
$RATE_3$	(max(click) + min(click) − click) × similar()

Algorithm 3 Generates predictions p_{ad} for each document

```
Require: $q_a$ A TEST QUERY
Require: $V$ THE RATING MATRIX
Require: $D$ A SET OF DOCUMENTS
  for all $d_i \in D \setminus (d_1, \ldots, d_k)$ do
    $p_{ai} \leftarrow$ CF-CALCULATION $(q_a, d_i)$
    $\overrightarrow{p_{a\cdot}} \leftarrow p_{ai}$
  end for
  return $\overrightarrow{p_{a\cdot}}$
```

In the user-based algorithm, we use Pearson's correlation coefficient to measure the similarity between q_a and $q \in Q$ (denoted as $s(a, q)$) as follows:

$$s(a,q) = \frac{\sum_{d \in D_a \cap D_q} (V_{ad} - \overline{V_{a\cdot}})(V_{qd} - \overline{V_{q\cdot}})}{\sqrt{\sum_{d \in D_a \cap D_q} (V_{ad} - \overline{V_{a\cdot}})^2 \sum_{d \in D_a \cap D_q} (V_{qd} - \overline{V_{q\cdot}})^2}}$$

where D_a denotes documents retrieved for q_a and D_q denotes documents retrieved for $q \in Q$. After calculating the similarity between different queries, we calculate predictions by considering the contribution of each neighbor in the matrix, weighted by its similarity to q_a $(neigh_a)$. We use the technique suggested by Herlocker et al.,[35] taking into account the mean $\overline{v_{a\cdot}}$, as well as the standard deviation of queries σ_a and σ_q. Similarly, we define the similarity between different documents (denoted as $s(d', d)$) for the item-based algorithm as

$$s(d',d) = \frac{\sum_{q \in Q} (V_{qd} - \overline{V_{q\cdot}})(V_{qd'} - \overline{V_{q\cdot}})}{\sum_{q \in Q} (V_{qd} - \overline{V_{q\cdot}})^2 \sum_{q \in Q} (V_{qd'} - \overline{V_{q\cdot}})^2}$$

In the weighted *SlopeOne* algorithm, $S_{dd'}$ is the set of queries that have "rated" both documents d and d'. In the SVD algorithm, we use a matrix factorization technique that converts V into three matrices:

$$V = U \cdot S \cdot R^T$$

where U and R are orthogonal matrices and S is a diagonal matrix of size $g \times g$ (where g is the rank of V). This matrix is iteratively reduced by discarding the smallest values, to produce a matrix S_γ with $\gamma < g$. The reconstructed matrix, $V = U_\gamma \cdot S_\gamma \cdot R_\gamma^T$ is the best $rank - \gamma$ approximation of the rating matrix V. We calculate CF predictions from this (reduced dimension) matrix using the formula stated above.

The memory-based algorithms use Pearson's correlation coefficient. In the weighted SlopeOne algorithm, $S_{dd'}$ is the set of users that have rated both documents d and d'. In the SVD algorithm, $V = U_\gamma \cdot S_\gamma \cdot R_\gamma^T$ is the best $rank - r$ approximation of the rating matrix V. The output of the CF operation is another ranked set of documents ranked.

In the final stage of our procedure, we combine a set of IR-generated results with the CF-generated results. This procedure is described in Algorithm 4, where we combine the top c documents returned for q_a by the IR engine with the vector of predictions produced in Algorithm 3 (c is arbitrarily set to 1000). We tried a variety of combinatorial methods, from the simple (e.g., CombMAX, CombSUM) to the complex (e.g., CombRSV%, CORI, Z-score).[36,37] CombRSV% seems to work the best. It is usually defined in the following way:

$$SUM(RSV_i/MAX_{RSV})$$

where RSV denotes the retrieval status value (i.e., the score). Now we have *three* sets of document rankings (IR scores, CF scores, and $combRSV$ scores). We sort the documents using all three scores (sort precedence as listed above, descending order) to produce a final ranking (the cutoff number is also set to 1000 in D_{final}).

Algorithm 4 Integrates IR-generated and CF-generated results to produce a final ranking list

```
Require: $q_a$ A TEST QUERY
Require: $C$ A DOCUMENT CORPUS
Require: $\overrightarrow{p_{a\cdot}}$ CF PREDICTIONS
  $(d_1, \ldots, d_c) \leftarrow$ IR-RETRIEVE $(q, C)$
  for all $d_i \in (d_1, \ldots, d_c)$ do
    if $p_{ad} \in \overrightarrow{p_{a\cdot}}$ then
      COMBRSV-SCORE $(d_i) \leftarrow$ COMBRSV $(p_{ad_i}$, IR-SCORE $(d_i))$
    end if
  end for
  for all $d_j$ THAT $p_{aj} > 0$ do
    $D_{final} \leftarrow (d_1, \ldots, d_c) \cup d_j$
  end for
  SORT BY IR SCORE, CF SCORE, COMBRSV-SCORE
  return $D_{final}$
```

As mentioned before, the CF algorithms often suffer from the following two problems: *sparsity* of the rating matrix and *cold start*. The sparsity problem occurs because in a typical recommender systems each user rates only a small subset of the available items, so most of the cells in the rating matrix are empty. This problem has very limited impact on our methods. This is because of two reasons. For the IR-based pseudoratings, we can enlarge the top-rank list, to prevent this happens. While for the CF-based runs, we merely use ratings to calculate final scores, not for prediction of accurate items. In the experiments presented below, we can verify this issue. The "cold start problem deals with the difficulty in making recommendations for users recently introduced into the system. In our methods, this means the coverage of the search logs is insufficient to generate enough similar queries to the submitted query. However, because of the procedure of locating similar queries, we have minimized the problem to the smallest scale. In the experiments below we also showed that even for some queries that could not find enough similar queries, our methods work well. We will come back to these two issues later on in the experiment sections. This concludes the description of our basic approach to EIR. In the next section, we will present three adjusting methods based on this approach.

4. REFINEMENT METHODS

In this section, we describe three refinement methods that can be used to improve the performance of the basic EIR system described in Section 3.

4.1. IterHITS

This algorithm assumes that documents and queries in a CF-based EIR system *mutually reinforce* each other, i.e., the predictions "made" by the queries depend on the predictions "made" by the documents and vice versa. This being the case, we can use an iterative, HITS-like algorithm[38] to produce the CF scores instead of raw CF-based methods. This method, which we call *IterHITS*, is described in Algorithm 5. After we obtain the prediction scores for the test query $\overrightarrow{p_{a\cdot}}$, we merge this CF-generated and IR-generated results (as shown in Algorithm 4).

Algorithm 5 The *IterHITS* refinement method, which automatically adjusts the CF-generated scores

```
Require: $\overrightarrow{p_{a\cdot}}$ CF PREDICTIONS
Require: $V$ THE RATING MATRIX
Require: $z$ THE NUMBER OF ITERATIONS REQUIRED
  SET $\overrightarrow{p_{\cdot d}}$ TO BE THE VECTOR $(1, 1, \ldots, 1) \in \mathbb{D}$
  SET $\overrightarrow{p_{\cdot a}}$ TO BE THE VECTOR $(1, 1, \ldots, 1) \in \mathbb{A}$
  for all $t = 1$ TO $z$ do
    $\overrightarrow{p_{\cdot d}}^{t+1} \leftarrow \overrightarrow{p_{a\cdot}}^{t} \times V$
    $\overrightarrow{p_{a\cdot}}^{t+1} \leftarrow \overrightarrow{p_{\cdot d}}^{t+1} \times V^T$
    NORMALISE $\overrightarrow{p_{\cdot d}}^{t+1}$
    NORMALISE $\overrightarrow{p_{a\cdot}}^{t+1}$
  end for
  return $\overrightarrow{p_{a\cdot}}^{t+1}$
```

4.2. IterCo

The *IterHITS* algorithm only considers one aspect of the results (i.e., the CF-generated scores). Our second refinement method goes one step further by assuming that the original CF-generated results, and the IR-generated results are mutually influence each other. This assumption derives from the fact that we use IR-generated scores as pseudoratings for the test query. Therefore, updating one set of scores should iteratively propagate (via the connected document graph) to the other set of scores.

Let $G = (N, E)$ be a connected graph, wherein nodes N correspond to the n documents, and edges E correspond to the associational strengths between documents (through the ratings of queries). Furthermore, assume an $n \times n$ symmetric weight matrix B on the edges of the graph, so that b_{ij} denotes the weight between documents d_i and d_j. The *IterCo* algorithm (iterative co-updating) is described in Algorithm 6. In each iteration, the CF- and IR-generated results are updated through a two-step algorithm, with one influences another. The operations of the two equations presented in the algorithm are linear combination with two parameters α and β. The entire algorithm will converges to a fixed point finally (see further in Refs. 39–41).

Algorithm 6 The *IterCo* refinement method, which automatically adjusts the CF and IR generated scores

```
Require: $\overrightarrow{p_{a\cdot}}$ CF PREDICTIONS
Require: $\overrightarrow{IR_{a\cdot}}$ THE IR SCORE VECTOR
Require: $V$ THE RATING MATRIX
Require: $z$ THE NUMBER OF ITERATIONS REQUIRED
  $B \leftarrow V^T V$
  ROW NORMALISE $B$
  for all $t = 1$ TO $z$ do
    $\overrightarrow{p_{a\cdot}}^{t+1} \leftarrow \alpha \cdot \overrightarrow{p_{a\cdot}}^{0} + (1-\alpha) \cdot B \cdot \overrightarrow{IR_{a\cdot}}^{t}$
    $\overrightarrow{IR_{a\cdot}}^{t+1} \leftarrow \beta \cdot \overrightarrow{IR_{a\cdot}}^{0} + (1-\beta) \cdot \overrightarrow{p_{a\cdot}}^{t+1}$
  end for
  return $\overrightarrow{p_{a\cdot}}^{t+1}, \overrightarrow{IR_{a\cdot}}^{t+1}$
```

4.3. Regu

Our third refinement method extends the iterative coupdating algorithm described above. For this method, we calculated a weight for each document based on its relationship with other documents in the pool (as with *IterCo*). After calculation, both the CF- and IR-generated scores are adjusted using a function that regularizes the smoothness of document associations over a connected graph. The intuition behind this method is the prior assumption of graph consistency—similar documents are likely to have similar ratings with respect to the test query. In other words, the CF and IR scores for the test query are adjusted by context enhancing and weighting propagation. The neighbors of a document in a connected graph should have similar rating scores as the document. In addition, the refined CF and IR scores should be partially restrained by the initial CF and IR scores (see Algorithms 3 and 4).

So, in addition to the connected graph constructed in Section 4.2, we further define M as a diagonal matrix with entries

$$M_{ii} = \sum_j b_{ij}$$

We also define a $n \times 2$ matrix F with

$$F = \left[\overrightarrow{P_a\cdot}\overrightarrow{IR_a\cdot}\right]$$

We use $f(a, d)$ to denote ratings of q_a assigned to d.

Then we develop a regularization framework for adjusting the CF- and the IR-generated scores. Formally, the cost function $\Re(F, a, G)$ in a joint regularization framework is defined as

$$\Re(F, a, G) = \frac{1}{2}\sum_{i,j=1}^{n} b_{ij}\left\|\frac{f(a, d_i)}{\sqrt{M_{ii}}} - \frac{f(a, d_j)}{\sqrt{M_{jj}}}\right\|^2 + \mu\sum_{i=1}^{n}\|f(a, d_i) - f^0(a, d_i)\|^2)$$

where $\mu > 0$ is the regularization parameter, $f^0(a, d_i\cdot)$ is the initial rating matrix of the test query q_a and the document d_i. F, and f^0 are the refined matrix and the initial matrix, respectively.

The first term on the right-hand side of the cost function is the *global consistency constraint*. This constraint ensures that the weighting function does not change too much between nearby points. In this experiment, nearby points are the refined rating scores reflecting the initial relationships between documents and context information. The second term on the right-hand side of the cost function is the *fitting constraint*. This constraint ensures that the ratings assigned to documents "fit" the initial ratings. The trade-off between these two variables is controlled by the parameter μ.

Given as the above, the final weighting function is defined as

$$F^* = arg \min_{F \in \Re^{+n}} \Re(F, a, G)$$

After simplification, we can derive the following closed form solution:

$$F^* = \mu_2(I - \mu_1 S)^{-1}F^0$$

where

$$\mu_1 = \frac{1}{1+\mu}$$

$$\mu_2 = \frac{\mu}{1+\mu}$$

$$S = M^{-\frac{1}{2}}AM^{\frac{1}{2}}$$

and I is an identity matrix (see further in Refs. 41, 40). Note that S is a normalized graph Laplacian matrix. Given the refined weighting matrix F, we can extract refined $\overrightarrow{p_{a\cdot}}$ an $\overrightarrow{IR_{a\cdot}}$ scores. It is worth noting that μ_2 could be eliminated as it does not change the ranking. This refinement method, which we call *Regu*, is described in Algorithm 7.

Then we apply CF-based EIR algorithm to merge the CF- and IR-generated results as shown in Algorithm 4. This concludes our discussion of the proposed regularization framework.

Algorithm 7 The *Regu* refinement method, which automatically adjusts the CF and IR generated scores

```
Require: $\overrightarrow{p_{a\cdot}}$ CF PREDICTIONS
Require: $\overrightarrow{IR_{a\cdot}}$ THE IR SCORE VECTOR
Require: $V$ THE RATING MATRIX
Require: $z$ THE NUMBER OF ITERATIONS REQUIRED
  $B \leftarrow V^T V$
  SET $M$ AS DIAGONAL MATRIX WITH ENTRIES $M_{ii} = \sum_j b_{ij}$
  $S \leftarrow M^{-\frac{1}{2}} A M^{\frac{1}{2}}$
  $F^0 \leftarrow \left[\overrightarrow{p_{a\cdot}}^0 \overrightarrow{IR_{a\cdot}}^0\right]$
  for all $t = 1$ TO $z$ do
    $F(t+1) = \mu S F(t) + (1-\mu) F^0$
  end for
  return $\overrightarrow{p_{a\cdot}}^{t+1}, \overrightarrow{IR_{a\cdot}}^{t+1}$
```

5. EVALUATION

In the following section, we describe a series of experiments designed to answer the following questions:

1. Does our CF-based EIR system perform better than traditional IR models?
2. How effective are the refinement methods outlined in §4?
3. Is a CF-based EIR system vulnerable to low matrix density or 'cold-start'?
4. Which CF technique performs the best?

5.1. Experimental Data

We evaluated our system using a publicly available Web page collection called *SogouT*.[a] Released for the NTCIR-9 *Intent* task, this corpus contains 135.4 million Web pages crawled from 5.3 million Chinese Web sites during 2008. The total uncompressed storage size of this collection is approximately 5 TB. We also used a publicly available click-through log called *SogouQ*. This log contains all of the click-through data collected by a Chinese search engine during June 2008. In this log, each click event is represented by a single line of tab-separated data containing the following fields:

- the ID of the user performing the search,
- the user's query,
- the ranking of the clicked URL,
- the ordering of the user click, and
- the URL that was clicked.

Note, in this experiment, we assume that a click event on a specific URL indicates that the clicked Web page is relevant to the submitted query, i.e., we are discounting accidental/erroneous click events. All of the Chinese documents and

[a] See further, http://www.sogou.com/labs/dl/t-e.html.

Table III. Summary of the experimental data set after the filtering operation.

Unique users	196,891
Unique queries	25,648
Unique documents	60,392
Maximum number of queries per user	30
Maximum number of docs per user	93
Minimum number of queries per user	1
Minimum number of docs per user	1

queries used in this experiment were preprocessed using a segmentation tool.[b] No other filtering was applied.

To create our experimental data set, we randomly sampled 7 days worth of data from the *SogouQ* log. We then attempted to map click-through events documented in *SogouQ* to documents cached the *SogouT* Web crawl. This reduced the data set significantly (see Table III). We randomly sampled 200 users from the *SogouQ* log who had submitted *less than three queries* (where no user profiles could be built for these users). We used these individuals as our test users.

5.2. Evaluation Metrics

We use the following evaluation metrics in this experiment:

- the precision of the top 5, top 10, and top 20 documents (P@5, P@10 and P@5),
- normalized discounted cumulative gain (NDCG),[42]
- the recall of the top 5, top 10, and top 20 documents (R@5, R@10 and R@20) and
- mean average precision (MAP).

Unless otherwise stated, the results given indicate *average performance* across all test users. Statistically significant differences in performance were determined using a paired *t* test at a confidence level of 95%.

5.3. Retrieval/Baseline Systems

All IR functions in our system were handled by the Terrier open source platform.[c] The first baseline we used was BM25, a popular and robust probabilistic retrieval model. We also used BM25PRF, a probabilistic retrieval function utilizing pseudorelevance feedback based on divergence from randomness theory.[43d] Finally,

[b]Available from http://code.google.com/p/paoding/
[c]http://terrier.org/.
[d]This function is included in the Terrier distribution.

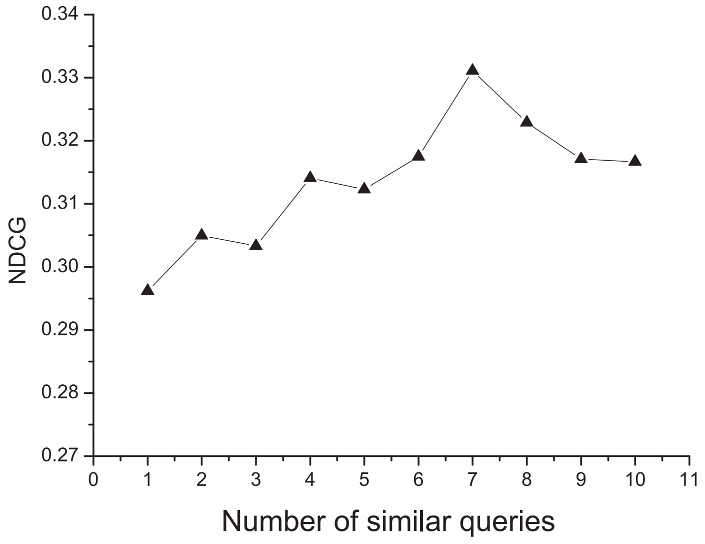

Figure 1. Tuning parameter m.

we used the raw output of our collaborative EIR system as a strong baseline for the evaluation of our iterative refinement methods.

5.4. Parameter Settings

All of the parameters used in this experiment were empirically set. Before the experiment proper, a subset of users was randomly selected from the search logs for this purpose. There was no overlap between the training set of users and the test set. According to our training runs, the optimal value for parameter m (the number of similar queries mined from the search log) is 7. We did not try any values for m beyond 10 (see Figure 1). We wanted to keep m as low as practicable to avoid the possibility of query drift.

We conducted a number of runs with different values for k (the number of top-ranked documents used when generating pseudoratings). As shown in Figure 2, NCDG was fairly consistent between 2 and 10 documents, but the optimal value was 5.

We tried three different methods when computing each rating in the matrix (see Algorithm 2 and Table II). $Rate_3$ scored highest in terms of NCDG (see Figure 3). We selected the Jaccard similarity measurement (see Table I), which led by a small margin (see Figure 4). We kept the default values set by the Terrier platform for the BM25 retrieval function, as this achieved the highest retrieval effectiveness. We tried a range of parameters for the BM25PRF retrieval function during setup, eventually settling on two expansion documents and ten expansion terms (see Figures 5 and 6). Our iterative refinement methods used $\alpha = 0.5$, $\beta = 0.5$, and $\mu = 0.9$, respectively. These parameters were set empirically.

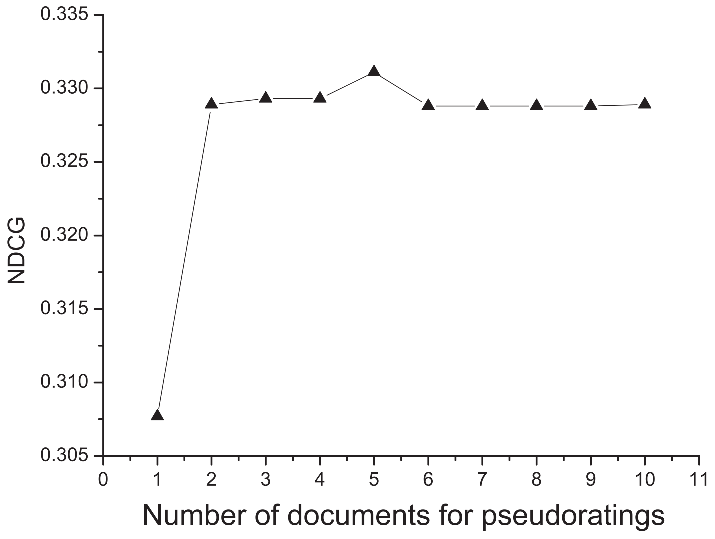

Figure 2. Tuning parameter k.

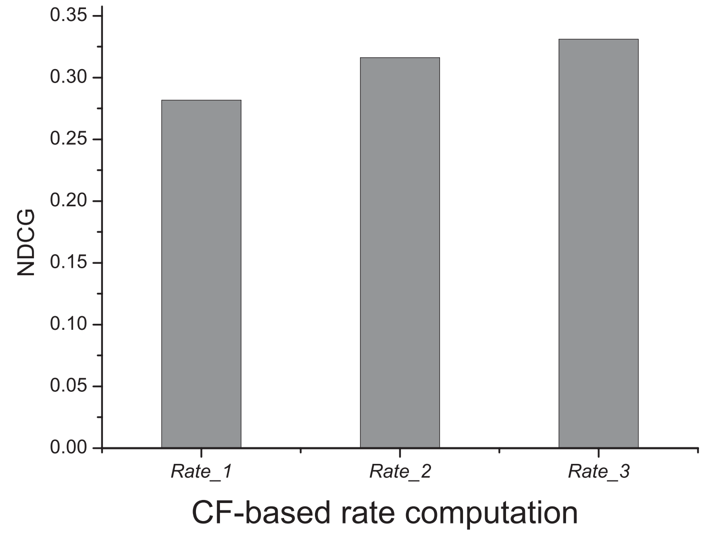

Figure 3. Selecting a rate computation method.

6. RESULTS

6.1. Baselines

In our first evaluation, we compared the performance of the unrefined system described in Section 3 (henceforth known as CF-base) with the baselines listed

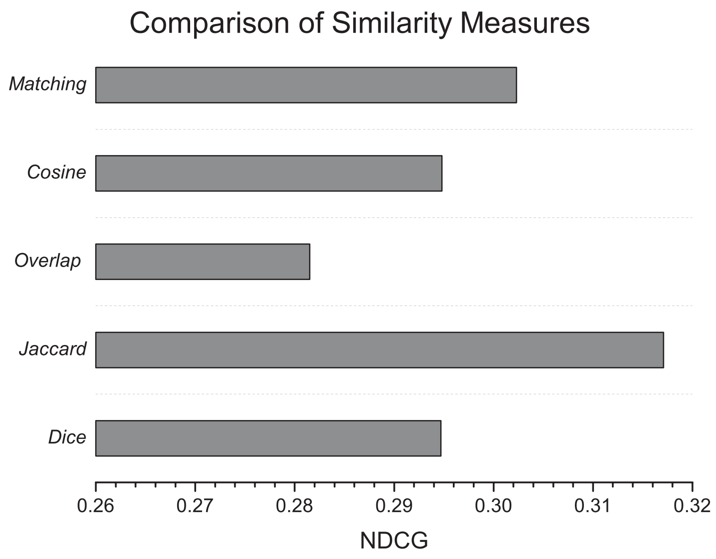

Figure 4. Selecting a similarity measure.

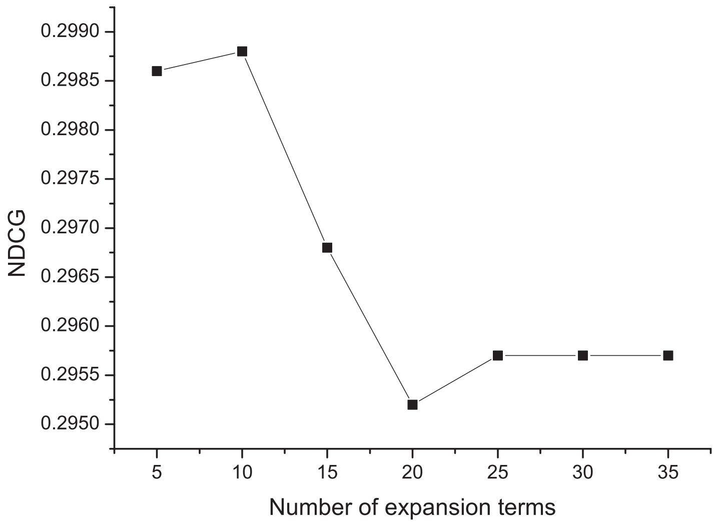

Figure 5. Selecting the number of expansion terms for the BM25PRF retrieval function.

in Section 5.3. The results are shown in Table IV. CF-base achieved statistically significant improvements over BM25 (marked with †) and BM25PRF (marked with ∗). This means our basic assumption in the beginning of the paper is verified that even without the user¡¯s history, we can still use other similar search interests to enhance the performance. However, the gains were modest. It shows that it

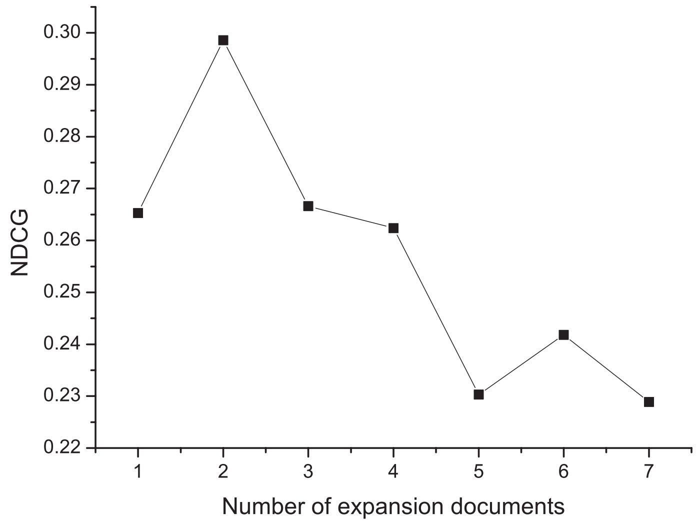

Figure 6. Selecting the number of expansion documents for the BM25PRF retrieval function

Table IV. A comparison of all retrieval systems.

	P@5	P@10	P@20	NDCG	MAP
BM25	0.0676	0.0464	0.0300	0.2776	0.1872
BM25PRF	0.0694	0.0473	0.0297	0.2790	0.1880
CF-base	0.0757†∗	0.0554†∗	0.0354†∗	0.2899†∗	0.1915∗
IterHITS	0.0784	0.0559	0.0380‡	0.3092‡	0.2000‡
IterCo	0.0814‡	0.0592‡	0.0396‡	0.3213‡	0.2092‡
Regu	0.0966‡	0.0681‡	0.0478‡	0.3406‡	0.2229‡

is necessary to further improve this basic approach. Henceforth, we will use this approach as a baseline in our later experiments.

6.2. Refinement Methods

In our second evaluation, we compared the refinement methods described in Section 4. As shown by Table IV, all three methods improved the retrieval effectiveness of CF-base (statistically significant improvements over CF-base are marked with a ‡). In terms of NDCG, *IterHITS* and *IterCo* improved the retrieval performance of CF-base by 6.66% and 10.83%, respectively, but *Regu* dominated with a 17.49% increase. The performance of *Regu* at P@5 and P@10 was also strong, showing improvements of 27.61% and 22.92% with respect to CF-base, respectively. The improvements of *Regu* over the nonenhanced baselines reached 42.9%/39.2% with respect to to BM25/BM25PRF (P@5). These results confirm our assumption earlier in the paper that the CF- and IR-generated results mutually reinforce each other.

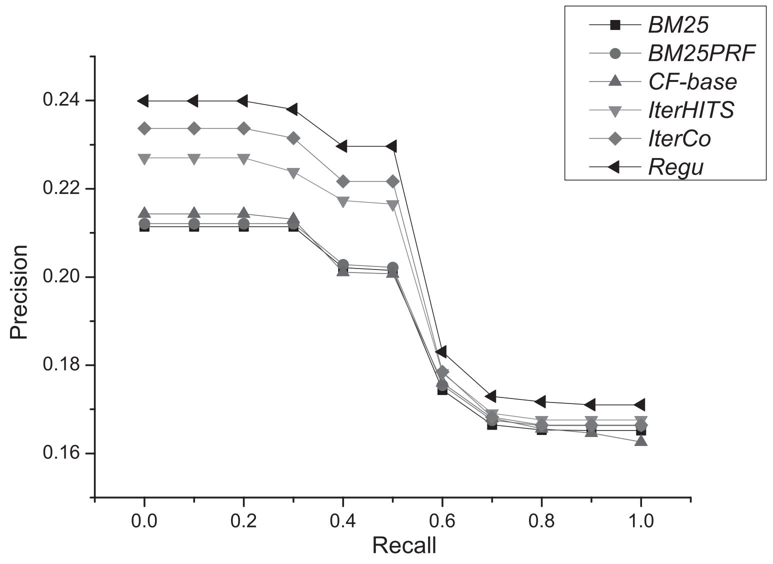

Figure 7. Precision–recall curve (all systems).

Figure 7, which plots the precision–recall curves for the various systems, suggests that the gains achieved with our methods are consistent. Interestingly, the refinement methods outscore the baselines on almost all of the evaluation metrics, despite being specifically tuned for NCDG (see Figure 8). As a whole, these results vindicate our earlier assumption that CF- and IR-generated results will act to mutual reinforce each other.

6.3. Collaborative Feedback Algorithms

In our third evaluation, we studied the performance of various collaborative feedback algorithms. The results are shown in Figure 9. The model-based CF algorithms (SVD and weighted SlopeOne) produced better results than the memory-based alternatives (user-and item-based). This was the result we expected. Memory-based CF algorithms tend to perform poorly when the rating matrix is sparse. Model-based algorithms are generally less sensitive to matrix density. It has been reported previously that the memory-based algorithms generally work better under relatively high density conditions (in terms of the rating matrix), but their accuracy decreases considerably under sparsity conditions given that they only use a small part of the available information. In the contrast, the model-based algorithms are less sensitive to density changes. The models tend to capture information that is hidden, difficult to extract using the memory-based methods. This is particularly useful in our experiments as the rating matrix maybe sparse when the cold-start problem exists.

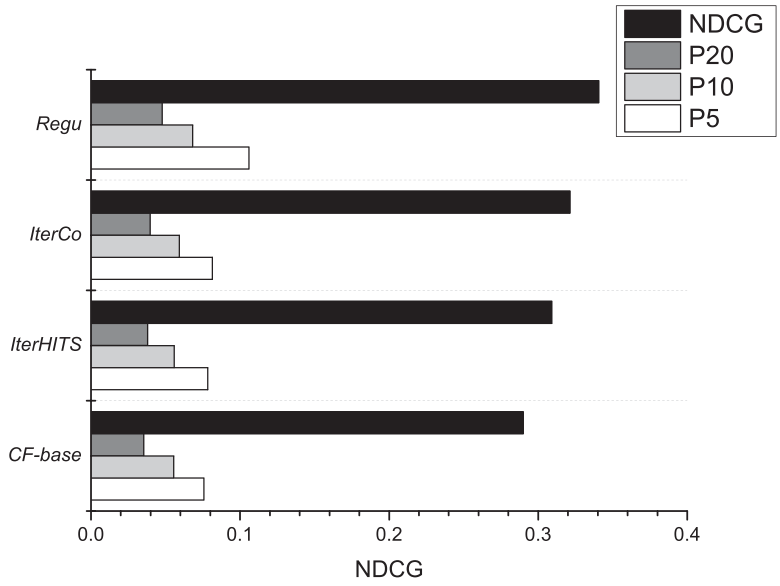

Figure 8. Comparison of the three refinement methods.

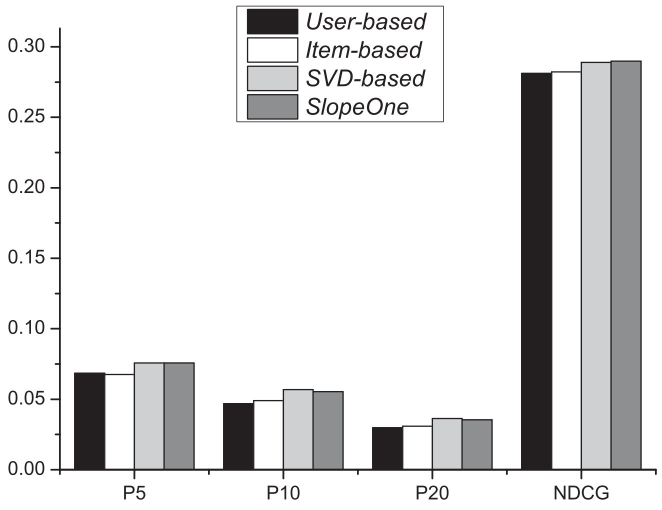

Figure 9. A comparison of CF algorithms.

Another observation is that the SVD-based method and the weighted SlopeOne-based method demonstrate similar performance. This further shows that the data sparsity problem is not a critical issue in our approach.

Table V. 'Cold start' query set.

	P@5	P@10	P@20	NDCG
BM25	0.1375	0.1125	0.0625	0.5577
BM25PRF	0.1125	0.0625	0.0375	0.3480
CF-base	0.1500	0.0812	0.0469	0.5762
IterHITS	0.1250	0.0812	0.0469	0.5808
IterCo	0.1500	0.0812	0.0500	0.6281
Regu	0.2000	0.1375	0.0875	0.7336

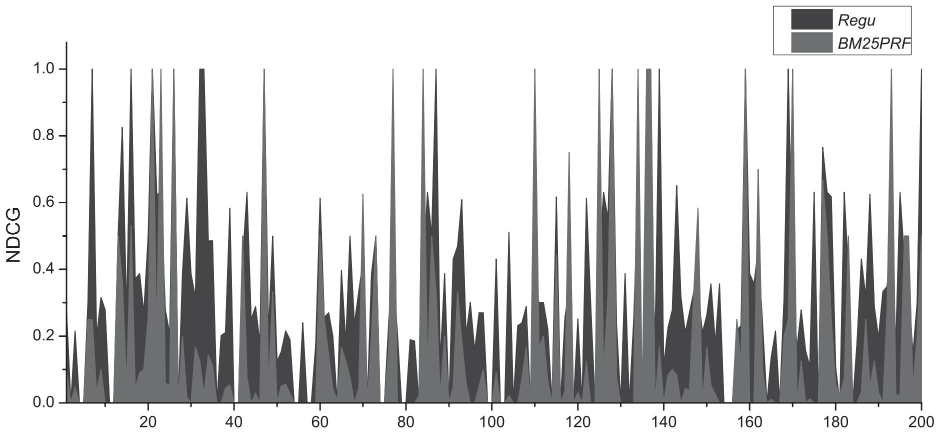

Figure 10. Per-user analysis.

6.4. "Cold Start"/Low-Density Problems

In our fourth evaluation, we wanted to examine the viability of EIR given a "cold start." We extracted a set of "problem case" queries from the data set. These queries had three or fewer matches when passed to the similarity function. We reran our experiment using these queries. Overall, refined EIR still outperformed nonenhanced retrieval overall (see Table V).

Matrix sparsity had a very limited impact on our results overall. We think there are two reasons for this. First, we can prevent low density in the rating matrix by simply enlarging the IR-based pseudoratings. Second, we only use the CF-generated results to calculate the final retrieval scores and *not* to predict the accurate items.

6.5. Per-user Analysis

Figure 10 provides a per-user analysis, which is useful as an overview. It compares the results obtained by BM25PRF, which was the highest scoring nonenhanced baseline, with the results returned by *Regu*, the highest scoring refinement method. 64.5% of all users (129 of 200) were better off with refined enhanced retrieval. Only 11% of users (22 of 200) were better off without the enhanced version.

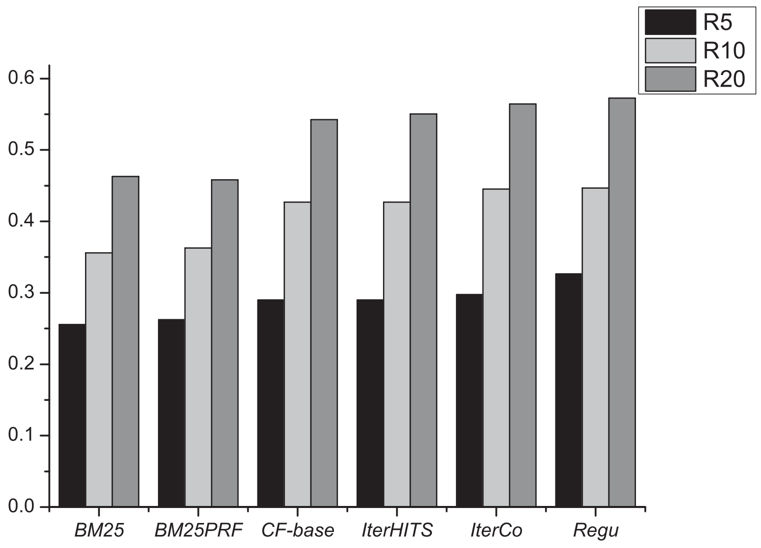

Figure 11. Recall-based evaluation.

6.6. Recall

In addition to the precision-based measurements used above, we measured the performance of our algorithms using various recall-based metrics. Enhanced retrieval achieved better recall than the baseline systems, although the improvements were less dramatic than seen elsewhere (see Figure 11). This above average recall performance has an intuitive explanation. Relevant documents are being crowd-sourced, i.e., "found" by other users.

7. CONCLUSIONS AND FURTHER WORK

In this paper, we have described a method to IR that combines CF algorithms with traditional IR models to enable *EIR*. We have also proposed various iterative methods for refining its performance. In a multistage evaluation using real-world data, our experimental system delivered statistically significant improvements over various baseline retrieval models. Crucially, these gains persisted even when the system was given a "cold start."

In future work, we plan to explore additional sources of collective user information beyond search logs, e.g., social media. We also plan to evaluate the applicability and performance of kernel methods when mining for similar queries.[44]

Acknowledgements

This research was supported by the National Natural Science Foundation of China under grant numbers 61300129 and 61272063, the Science Foundation Ireland under grant number

07/CE/I1142 (Centre for Next Generation Localisation), and a project Sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, China under grant number [2013] 1792 (Research on results re-ranking techniques for cross-language IR).

References

1. Cao H, Jiang D, Pei J, Chen E, Li H. Towards context-aware search by learning a very large variable length hidden Markov model from search logs. In: Proc 18th Int Conf on World Wide Web, WWW ’09. New York: ACM; 2009. pp. 191–200.
2. Liu F, Yu C, Meng W. Personalized web search for improving retrieval effectiveness. IEEE Trans Knowl Data Eng 2004;16(1):28–40.
3. Pariser E. The filter bubble : what the Internet is hiding from you. New York: Penguin Press; 2011.
4. Haveliwala T, Kamvar S, Jeh G. An analytical comparison of approaches to personalizing pagerank. Technical Report 2003-35. Stanford InfoLab; June 2003.
5. Li L, Otsuka S, Kitsuregawa M. Finding related search engine queries by web community based query enrichment. World Wide Web 2010;13(1–2):121–142.
6. Xu G, Li L, Zhang Y, Yi X, Kitsuregawa M. Modeling user hidden navigational behavior for web recommendation. Web Intell Agent Syst 2011;9(3):239–255.
7. Chirita PA, Firan CS, Nejdl W. Personalized query expansion for the web. In: Proce 30th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, SIGIR ’07. New York: ACM; 2007. pp. 7–14.
8. Dou Z, Song R, Wen J-R. A large-scale evaluation and analysis of personalized search strategies. In: Proc 16th Int Conf on World Wide Web, WWW ’07. New York: ACM: 2007. pp. 581–590.
9. Luxenburger J, Elbassuoni S, Weikum G. Matching task profiles and user needs in personalized web search. In: Proc 17th ACM Conf on Information and Knowledge Management, CIKM ’08. New York: ACM; 2008. pp. 689–698.
10. Gauch S, Speretta M, Chandramouli A, Micarelli A. User profiles for personalized information access. In: The adaptive web. Berlin, Germany: Springer-Verlag; 2007. pp. 54–89.
11. Sontag D, Collins-Thompson K, Bennett PN, White RW, Dumais S, Billerbeck B. Probabilistic models for personalizing web search. In: Proc 5th ACM Int Conf on Web Search and Data Mining, WSDM ’12. New York: ACM; 2012. pp. 433–442.
12. Speretta M, Gauch S. Personalized search based on user search histories. In: Proc 2005 IEEE/WIC/ACM Int Conf on Web Intelligence, WI ’05. Washington, DC: IEEE Computer Society; 2005. pp. 622–628.
13. Zhang Y, Koren J. Efficient Bayesian hierarchical user modeling for recommendation system. In: Proc 30th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, SIGIR ’07. New York: ACM; 2007. pp. 47–54.
14. Shen X, Tan B, Zhai C. Implicit user modeling for personalized search. In: Proc 14th ACM Int Conf Information and Knowledge Management, CIKM ’05; New York: ACM; 2005. pp. 824–831.
15. Zhou D, Lawless S, Wade V. Improving search via personalized query expansion using social media. Inform Retr 2012;15(3–4):218–242.
16. Wang Q, Jin H. Exploring online social activities for adaptive search personalization. In: Proc 19th ACM Int Conf on Information and Knowledge Management, CIKM ’10. New York: ACM; 2010 pp. 999–1008.
17. Xu S, Bao S, Fei B, Su Z, Yu Y. Exploring folksonomy for personalized search. In: Proc 31st Annual Int ACM SIGIR Conf Research and Development in Information Retrieval, SIGIR ’08. New York: ACM; 2008. pp. 155–162.
18. Li L, Xu G, Zhang Y, Kitsuregawa M. Random walk based rank aggregation to improving web search. Knowl-Based Syst 2011;24(7):943–951.

19. Smyth B, Balfe E. Anonymous personalization in collaborative web search. Inform Retr 2006;9(2):165–190.
20. Yin Z, Shokouhi M, Craswell N. Query expansion using external evidence. In: Proc 31th Eur Conf on IR Research on Advances in Information Retrieval, ECIR '09. Berlin, Germany: Springer-Verlag; 2009. pp. 362–374.
21. Shardanand U, Maes P. Social information filtering: algorithms for automating 'word of mouth'. In: Proc SIGCHI Conf on Human Factors in Computing Systems, CHI '95. New York: ACM Press/Addison-Wesley; 1995. pp. 210–217.
22. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J. Grouplens: an open architecture for collaborative filtering of netnews. In: Proc of the 1994 ACM Conf on Computer Supported Cooperative Work, CSCW '94. New York: ACM; 1994. pp. 175–186.
23. Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: Proc 10th Int Conf on World Wide Web, WWW '01. New York: ACM; 2001. pp. 285–295.
24. Billsus D, Pazzani MJ. Learning collaborative information filters. In: Proc 15th Int Conf on Machine Learning, ICML '98. San Francisco, CA: Morgan Kaufmann; 1998. pp. 46–54.
25. Hofmann T. Latent semantic models for collaborative filtering. ACM Trans Inform Syst 2004;22(1):89–115.
26. Cacheda F, Carneiro V, Fernández D, Formoso V. Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender systems. ACM Trans Web 2011;5(1):2:1–2:33.
27. Weimer M, Karatzoglou A, Le Q, Smola A. Cofirank maximum margin matrix factorization for collaborative ranking. In: Advances in neural information processing systems, MIT Press Cambridge, MA; 2007, pp. 1593–1600.
28. Liu NN, Yang Q. Eigenrank: a ranking-oriented approach to collaborative filtering. In: Proc 31st Annual Int ACM SIGIR Conf on Research and sevelopment in information retrieval, SIGIR '08. New York: ACM; 2008. pp. 83–90.
29. Cao B, Shen D, Wang K, Yang Q. Clickthrough log analysis by collaborative ranking. In: Twenty-Fourth AAAI Conf on Artificial Intelligence, AAAI 2010, Atlanta, GA, July 11-15, 2010. Palo Alto, CA: AAAI Press; 2010. pp. 224–229.
30. Bellogín A, Wang J, Castells P. Text retrieval methods for item ranking in collaborative filtering. In: Proc 33rd Eur Conf on Advances in Information Retrieval, ECIR'11. Berlin, Germany: Springer-Verlag; 2011. pp. 301–306.
31. Costa A, Roda F. Recommender systems by means of information retrieval. In: Proc Int Conf on Web Intelligence, Mining and Semantics, WIMS '11. New York: ACM; 2011. pp. 57:1–57:5.
32. Zhou D, Truran M, Liu J, Zhang S. Collaborative pseudo-relevance feedback. Expert Syst Appl 2013;40(17):6805–6812.
33. Zhou D, Truran M, Liu J, Zhang S. Using multiple query representations in patent prior-art search. Inform Retr In Press.
34. Li W, Ganguly D, Jones GJ. Enhanced information retrieval using domain-specific recommender models. In: Amati G, Crestani F, editors. Advances in information retrieval theory. Lecture Notes in Computer Science Vol 6931. Berlin, Germany: Springer; 2011. pp. 201–212.
35. Herlocker J, Konstan JA, Riedl J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inform Retr 2002;5(4):287–310.
36. Savoy J. Combining multiple strategies for effective monolingual and cross-language retrieval. Inform Retr 2004;7(1–2):121–148.
37. Savoy J. Comparative study of monolingual and multilingual search models for use with Asian languages. ACM Transactions on Asian Language Information Processing. 2005;4(2):163–189.
38. Kleinberg JM. Authoritative sources in a hyperlinked environment. J ACM 1999;46(5):604–632.
39. Wang F, Zhang C. Label propagation through linear neighborhoods. In: Proc 23rd Int Conf on Machine Learning, ICML '06. New York: ACM; 2006. pp. 985–992.

40. Zhu X, Ghahramani Z, Lafferty J. Semi-supervised learning using Gaussian fields and harmonic functions. In: Twentieth Int Conf on Machine Learning, August 21-24, 2003. pp. 912–919.
41. Zhou D, Bousquet O, Lal TN, Weston J, Scholkopf B. Learning with local and global consistency. In: Advances in Neural Information Processing Systems, MIT Press Cambridge, MA; 2004, pp. 321–328.
42. Järvelin K, Kekäläinen J. IR evaluation methods for retrieving highly relevant documents. In: Proc 23rd Annual Int ACM SIGIR Conf on Research and development in Information Retrieval, SIGIR '00. New York: ACM; 2000. pp. 41–48.
43. Amati G, Van Rijsbergen CJ. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans Inform Syst 2002;20(4):357–389.
44. Sahami M, Heilman TD. A web-based kernel function for measuring the similarity of short text snippets. In: Proc 15th Int Conf on World Wide Web, WWW '06. New York: ACM; 2006. pp. 377–386.