first 50 results retrieved for a query. Each of these 50 search results is given a weight and are re-ranked accordingly.

3.2.1 Weighting Methods

When reranking, each candidate document can either be scored, or just the snippets can be scored. We focus on assigning scores to the search snippets as it was found to be more effective for re-ranking search results by Teevan et al. [31]. Also, using search snippets allows a straightforward client-side implementation of search personalization. We implemented the following four different weighting methods:

Matching

For each word in the search snippet's title and summary that is also in the user's list of profile terms, the weight associated with that term will be added to the snippet's weight:

$$score_M(s_i) = \sum_{z=0}^{N_{s_i}} f_{t_i} \times w(t_z) \quad (5)$$

where N_{s_i} represents the total number of unique words within the snippet's title and summary, and f_{t_i} represents the number of occurrences of t_i within the snippet. Words in the snippet title or summary but not in the user's profile do not contribute towards the final weight. This method is equivalent to taking the dot product between the user profile vector and the snippet vector.

Unique Matching

A second search snippet weighting option we consider involves counting each unique word just once:

$$score_{UM}(s_i) = \sum_{z=0}^{N_{s_i}} w(t_z) \quad (6)$$

Language Model

The third weight calculation method attempts to generate a unigram language model from the user profile in which the weights associated to the terms are used as the frequency counts for the language model:

$$\begin{aligned} score_{LM}(s_i) &= log(\prod_{z=0}^{N_{s_i}} \frac{w(t_z)+1}{w_{total}}) \\ &= \sum_{z=0}^{N_{s_i}} log(\frac{w(t_z)+1}{w_{total}}) \end{aligned} \quad (7)$$

where N is the total number of words in the snippet's title and summary, and w_{total} stands for the sum of all the weights within the user profile. The language model estimates the probability of a snippet given a user's profile. To avoid a zero probability for snippets that contain words not in the user's profile, we use add-1 smoothing.

PClick

As a final snippet weighting method we use the PClick algorithm proposed by Dou et al. [8]. It assumes that for a query q submitted by a user u, the web pages frequently clicked by u in the past are more relevant to u. The personalized score for a snippet is:

$$score_{PC}(s_i) = \frac{|Clicks(q,p,u)|}{|Clicks(q,\bullet,u)| + \beta} \quad (8)$$

where $|Clicks(q,p,u)|$ is the number of clicks on web page p by user u for query q in the past, $|Clicks(q,\bullet,u)|$ is the total click number on query q by u, and β is a smoothing factor set to 0.5. Note that PClick makes no use of the terms and weights associated to the user's profile and is solely based on click-through data for a given query. As such, it only affects repeated queries.

3.2.2 Additional Options

Finally, we consider two adjustments to the snippet scores. First, we consider giving additional weight to URLs that have been visited previously. This extends PClick in that it boosts *all* URLs that have previously been visited, while PClick only boosts URLs that have directly been clicked for the current search query. The snippet weight will be boosted by the number of previous visits to that web page (n) times a factor v:

$$\text{finalScore}(s_i) = score(s_i) * (1 + v \times n_i) \quad (9)$$

Second, in the re-ranking framework discussed so far, the original ranking is not taken into account. The original rank can be incorporated into the final snippet weight by multiplying the snippet weight by the inverse log of the snippet's original rank r_{s_i}:

$$\text{finalScore}(s_i) = score(s_i) \times \frac{1}{log(r_{s_i})} \quad (10)$$

We expect both these extensions to be very beneficial.

4. EVALUATION APPROACH

We now consider potential evaluations for personalized search strategies. On the one hand, offline approaches allow the creation of a standard dataset that can be used to optimize personalization parameters. On the other hand, only an online test with actual users can truly reflect how changes to rankings affect user behavior. We now explore the available alternatives, and describe our final strategy.

Relevance judgements

The first offline evaluation approach (e.g. used by Teevan et al. [31]) is based on assembling a group of people that judge the relevance of the top k documents or search snippets for a set of queries. Given these relevance judgements, a standard metric such as (N)DCG or (Normalized) Discounted Cumulative Gain [**?**] can be calculated for a given query and ranking, reflecting the quality of the presented ranking for that user. This approach has the advantage that once the relevance judgements are made, it allows for testing many different user profile and re-ranking parameter configurations. However, due to the long time it takes to judge k documents, this can only be done for a small number of search queries. As volunteers need to be found to sit through this slow and tedious evaluation process, it is also hard to gather a large group of evaluators. The evaluation process also does not reflect a user's normal browsing and searching behavior, which might influence the final results. Moreover, this approach assumes that (N)DCG is the right way to combine a set of relevance judgements into a rank quality score. Finally, the queries evaluated must be representative of a true query load, or offline results may not reflect perhaps poorer performance for non-personalizable queries.