An Optimization Framework for Weighting Implicit Relevance Labels for Personalized Web Search

Yury Ustinovskiy
Yandex
Leo Tolstoy st. 16, Moscow, Russia
yuraust@yandex-team.ru

Gleb Gusev
Yandex
Leo Tolstoy st. 16, Moscow, Russia
gleb57@yandex-team.ru

Pavel Serdyukov
Yandex
Leo Tolstoy st. 16, Moscow, Russia
pavser@yandex-team.ru

ABSTRACT

Implicit feedback from users of a web search engine is an essential source providing consistent personal relevance labels from the actual population of users. However, previous studies on personalized search employ this source in a rather straightforward manner. Basically, documents that were clicked on get maximal gain, and the rest of the documents are assigned the zero gain. As we demonstrate in our paper, a ranking algorithm trained using these gains directly as the ground truth relevance labels leads to a suboptimal personalized ranking.

In this paper we develop a framework for automatic reweighting of these labels. Our approach is based on more subtle aspects of user interaction with the result page. We propose an efficient methodology for deriving confidence levels for relevance labels that relies directly on the objective ranking measure. All our algorithms are evaluated on a large-scale query log provided by a major commercial search engine. The results of the experiments prove that the current state-of-the-art personalization approaches could be significantly improved by enriching relevance grades with weights extracted from post-impression user behavior.

Categories and Subject Descriptors

H.3.3 [**Information Systems Applications**]

General Terms

Algorithms, Measurement, Experimentation

Keywords

Personalization; Click Prediction; Gradient Descent

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741105.

1. INTRODUCTION

Most of major search engines record detailed information about users' requests, including timestamps of actions, sets of retrieved documents, and detailed interactions of users with SERPs (search engine result pages), and store petabytes of such data. For these reasons, scalable methods of user logs processing and utilization are gaining an increasing interest in the IR community. The rapid development of these technologies made possible various practical applications of implicit feedback analysis, including evaluation of ranking quality [12], improvement of web search performance [1, 9], and web search personalization [4, 5, 6, 8, 11, 14, 16, 19, 21, 23].

Implicit feedback data are often employed as the source of implicit personalized relevance labels and thus become absolutely indispensable for personalization of web search. Such data allow the collection of a large-scale sample of judgements for a representative set of queries directly from users [14]. These judgements could be further used as the ground truth relevance labels for training and evaluating personalized rankers.

The most straightforward yet still prevalent approach treats clicked documents as relevant and non-clicked ones as totally irrelevant. However, clearly, not every click is an indicator of high personal document relevance and not every impression without a click indicates complete irrelevance. Noisy clicks may mislead learning to rank algorithm decreasing the quality of the trained ranking model. This raises the problem of identifying "reliable" clicks. A step toward solving this problem was made in [3, 4], where the authors considered only *satisfied clicks* (i.e., the last clicks on SERP and clicks followed by a dwell time exceeding some fixed threshold) as the evidence of relevance. Since satisfied clicks are less noisy than all clicks, it is expected that a personalized ranker trained on these gain values results in a better ranking.

In this paper we suggest a far-reaching generalization of this idea. Namely, 'satisfied clicks' approach to the collection of the personalized relevance labels still misses valuable information available in the click-through log. In particular, this approach (1) does not distinguish between skipped documents and not examined documents; (2) does not account for various types of satisfied clicks differently (e.g., multiple clicks, singleton clicks, first clicks, super long clicks); (3) does not take into account positional bias. We aim at improving personalized ranking by interpreting implicit feedback in a more sophisticated way than it has been done in the the current approaches to search personalization, see,

e.g., [4, 11, 16]. The key component of our framework is an additional step which automatically extracts confidence levels of relevance labels from characteristics of user's actions on the SERP *prior* to training a personalized ranker. The relevance labels equipped with these *weights*, utilized by machine learning algorithm, are expected to improve the ranking quality.

To compare our method with the state-of-the-art approaches we need some fixed evaluation methodology. For this reason we use several popular implicit relevance labels to evaluate quality of personalized rankers and refer to these labels as *gains*.

To sum up, we make the following contributions:

- formulate the problem of weighting the implicit relevance labels for personalized search;
- propose a novel optimization framework for automatic training of confidence levels of implicit relevance labels as functions of characteristics of user behavior;
- provide new findings on user behavior modelling for web search personalization. In particular, we analyze the impact of dwell time and original ranking position on the confidence of the implicit personal relevance of the corresponding document.

We evaluate classical personalized relevance labels enriched with our confidence levels on the large-scale search log provided by Yandex[1] for the data mining challenge on web search personalization organized at `Kaggle.com`[2] in the scope of WSCD workshop held at WSDM 2014, [13]. The team that has won the challenge shared with us the dataset they used for training and testing their models. Our experiments prove that even given a very competitive personalization algorithm using a variety of strong state-of-the-art features and a powerful machine learning algorithm one can significantly improve the quality of its ranking by introducing weights of labels.

The remainder of the paper is organized as follows. In Section 2 we review related work on personalization, learning of user behavior models and semi-supervised learning. In Section 3 we fix some notations and give the formal statement of our problem. The core section of the paper is Section 4, where we define our algorithm for general and linear models. In Section 5 we give details of our experimental setup and list the features we use for learning. Results of the experiments are reported in Section 6. Finally, we discuss possible directions of future work and conclude the paper in Sections 7 and 8.

2. RELATED WORK

In numerous studies [4, 5, 6, 11, 14, 16, 23] appeared in the last decade, it has been demonstrated that personal and contextual information extracted from click-through logs has the potential to significantly improve personalized ranking quality. First papers on search personalization employ explicit relevance labels to learn ranking models. In [14] Shen *et al.* employed previous queries and clicks on the returned results to specify the actual information need of a searcher. The authors evaluated several context-sensitive language models on the TREC collection augmented with click-through data and used editorial relevance labels to tune and evaluate their algorithms. Tan *et al.* [17] collected search history from users to get information about users' search preferences. The approach turned out to be especially effective for recurring queries. All the experiments were evaluated on a set of manually labelled queries collected from volunteers. Matthijs and Radlinski [11] studied the effect of a set of personalized features extracted from a browsing history by tuning a few parameters. The authors reported results of both offline experiments on the labelled set of queries and online evaluations. As opposed to these studies, we train and evaluate all our algorithms on user clicks stored in search logs.

A large body of work has been devoted to search personalization algorithms using clicks for learning and evaluation. Bennett *et al.* [3] personalized web search by leveraging location information from both users and web sites. The authors re-ranked documents by training a location-sensitive model and evaluated it on a proprietary dataset collected via a browser plugin. They used *satisfied clicks* as an indicator of relevance and considered MRR (Mean Reciprocal Rank) of these clicks as the principal performance measure. The same labelling method and performance measures were chosen in [22]. In this paper White *et al.* improved ranking of results for the current user query by using behavioral data of the other users in the same cohort. In this study we consider the same evaluation methodology based on users' clicks. However, unlike the above papers, we propose to use these binary gain values to train a personalization ranker not directly. We develop a framework for systematic learning of numerically valued weights of these gains by analyzing different aspects of user interaction with the result page. We also emphasize that our goal is complementary to the one pursued in the above-mentioned studies on personalization mainly devoted to investigations of different sources of personalization features. Studies on personalization form a small portion of works employing implicit feedback to train machine learning algorithms. One of the pioneer works in this area is [9], where Joachims collects implicit feedback from a set users' clicks to improve the overall retrieval quality of a search engine. Agichtein *et al.* [1] collect features representing user implicit feedback and used them to learn a new ranking model. Authors employed manually assigned relevance labels to train and evaluate their model. Unlike these papers we do not aggregate clickthrough data to estimate document relevance, instead we estimate confidence in a personal relevance label of a document in a particular search session to a particular user.

The idea of our study is related to the one suggested in [26]. In that paper Yue *et. al.* improved conventional interleaving algorithms by interpreting users' actions in a more subtle way than it is defined in the state-of-the-art interleaving techniques. The authors suggested a machine learning approach that solves the *reverse problem* of hypothesis testing — it weights various types of clicks differently in order to improve the statistical power of the interleaving metrics. Our approach can also be considered a solution to the reverse problem of personalization, however, in contrast to that study, we (1) weight not only clicked documents and (2) focus on a completely different task: personalized web search ranking.

[1] `yandex.com`

[2] `kaggle.com/c/yandex-personalized-web-search-challenge`

An interesting theoretical machine learning approach to the general *reverse learning problem* was proposed in [25] by Xu *et al.* Similarly to our study, they combined two classes of features, one of which is known only for a subset of samples. To improve a standard linear machine learning model, authors estimated unknown feature values relying on the values of learning gains. Unlike that paper, in our problem statement, features describing user's interaction with the result page are available in training data only.

3. PROBLEM FORMULATION

3.1 Notations

In this section we fix notations and provide some insights into personalization framework. First, let us describe the data that are available in personalization learning.

Let $\mathcal{S} = \{t_1, \dots, t_S\}$ be the *training dataset*, consisting of S samples t_i. Each sample t_i is a tuple $\{u, q, d\}$ corresponding to a user u, a query q issued by this user, and a document retrieved for the query. Let $\mathcal{Q}$ be the set of user-query pairs occurring in the training dataset $\mathcal{S}$. We assume that in dataset $\mathcal{S}$ for each pair $\{u, q\} \in \mathcal{Q}$, we have exactly 10 tuples t_i in $\mathcal{S}$, corresponding to the top $k = 10$ retrieved documents. These 10 tuples are further denoted as $\mathcal{S}_{u,q} \subset \mathcal{S}$.

Each sample t_i is characterized by an N-dimensional *personalization feature vector* $\mathbf{x}_i = (x_i^1, \dots, x_i^N) \in \mathbb{R}^N$ that can be calculated based on the data available before the result page is shown to the user. Those may be any possible features that can be potentially used as components of a personalized ranking model. Among them, there could be both non-personal features, e.g., text relevance of document d to query q, and personal features, e.g., user-specific popularity of d or the match between the document's text and the user's interest profile. In what follows we denote by $\mathcal{X}$ the map that sends samples to their feature vectors:

$$\mathcal{X}\colon \mathcal{S} \to \mathbb{R}^N, \quad \mathcal{X}\colon t_i \mapsto (x_i^1, \dots, x_i^N) \in \mathbb{R}^N.$$

The goal of the personalized web search is to re-rank for every query $\{u, q\} \in \mathcal{Q}$ the corresponding documents $\mathcal{S}_{u,q}$, on the basis of the personalization feature vectors. Typically, it is done by providing a *scoring function* $\mathcal{F}$, which maps the personalization features $\mathcal{X}(\{u, q, d\})$ to the ranking scores of the documents. Since the values of function $\mathcal{F}$ depend only on the feature vectors $\mathcal{X}(\{u, q, d\})$, we will further consider $\mathcal{F}$ as a function on the N-dimensional feature space $\mathbb{R}^N$:

$$\mathcal{F}\colon \mathbb{R}^N \to \mathbb{R}.$$

Hereby samples $t_i \in \mathcal{S}_{u,q}$ are ranked according to the values $f_i = \mathcal{F}(\mathcal{X}(t_i))$.

To evaluate the quality of a personalized ranker we need *gain values*, that is ground truth relevance labels assigned to all documents, retrieved for the queries $\{u, q\} \in \mathcal{Q}$. So, we define *gain* as a function

$$\mathcal{G}\colon \mathcal{S} \to \mathbb{R}.$$

In the classical information retrieval approach, editorial relevance labels are assigned to the query-document pairs by assessors and could be directly used to define the *gain function* $\mathcal{G}$. For example, the standard mapping used in DCG (Discount Cumulative Gain) or NDCG (Normalized Discount Cumulative Gain) takes values in $\{2^g - 1 | g = 0, 1, 2, 3, 4\}$.

The situation in the personalization of Web search is quite different: collecting a large-scale sample of personalized relevance labels from real users in the same manner is usually significantly more complicated and practically infeasible. The approach widely adopted in the existing studies on personalized search [4, 11, 16] is to use certain user actions as implicit indicators of a document's personal relevance, e.g., to consider clicked documents with dwell time >30 seconds as relevant and the rest of the documents as irrelevant: $\mathcal{G}(\{u, q, d\})$ = longclick(d) := {1 iff d is clicked, and dwell time >30s}. This methodology allows to collect large training datasets comprising data from click-through logs of commercial search engines.

Given a gain function $\mathcal{G}$, one defines the *performance measure* $\mathcal{O}\colon \mathrm{Perm}(\mathcal{S}_{u,q}) \to \mathbb{R}$ on the set of rankings, i.e., permutations, of $\mathcal{S}_{u,q}$, in the following way:

$$\mathcal{O}(\sigma) = \sum_{t \in \mathcal{S}_{u,q}} \mathcal{G}(t) \cdot \mathrm{discount}(\sigma(t)), \tag{1}$$

where $\sigma\colon \mathcal{S}_{u,q} \to \{1, \dots, k\}$ is an ordering of documents in $\mathcal{S}_{u,q}$, $\sigma(\{u, q, d\})$ is the position of document d in the ranking, and discount(k) is a positional discount (e.g., $1/\log_2(k+1)$ for DCG or $1/k$ for MRR).

Objective performance measure of a ranker $\mathcal{F}$ is the average of $\mathcal{O}\big(\{rank_{\mathcal{F}}(d)\}_{d \in \mathcal{S}_{u,q}}\big)$ over all user-query pairs $\{u, q\} \in \mathcal{Q}$, where $rank_{\mathcal{F}}(d) \in \{1, \dots, k\}$ is the rank of document d according to the value of the scoring function $\mathcal{F}(\mathcal{X}(\{u, q, d\}))$:

$$\mathcal{M}(\mathcal{F}) := \frac{1}{|\mathcal{Q}|} \sum_{\{u,q\} \in \mathcal{Q}} \mathcal{O}\big(\{rank_{\mathcal{F}}(d)\}_{d \in \mathcal{S}_{u,q}}\big) \to \max, \tag{2}$$

where $\mathcal{O}(\cdot)$ is introduced in Equation (1). The family of performance measures of the form (2) includes, as its special cases, both DCG-like measures in the case of multi-graded editorial relevance labels $\mathcal{G}$ and classical click measures (e.g., MRR or MAP) in the case of click-through-based gains $\mathcal{G}$.

3.2 Our approach

So far, we have the training set $\mathcal{S}$, the gain function $\mathcal{G}$, and the feature map $\mathcal{X}$. Our goal is to produce a scoring function $\mathcal{F}$ as optimal as possible in terms of the objective metric (2). With this formulation of personalization learning framework one can apply the majority of the state-of-the-art machine learning algorithms, including linear regression, polynomial regression, decision trees models, neural networks, etc.

Standard machine learning algorithms previously applied to search personalization assume that all learning samples $t_i \in \mathcal{S}$ carry equally important evidences for a machine learning algorithm. We argue that this straightforward approach leads to a suboptimal ranking. For example, assume, when evaluating ranker $\mathcal{F}$, only the documents, followed by a satisfied click, are treated as relevant answers, i.e., $\mathcal{G}(\{u, q, d\})$ = (last click) OR (dwell time>30s). Let d be a document from the training set such that $20s <$ dwell time(d) $\leq 30s$. A standard machine learning algorithm utilizes the right inequality only. However, intuitively, the information that the click on document d has dwell time larger than 20s may carry an additional signal for the machine learning algorithm and could be also useful for training ranker $\mathcal{F}$. We may think of these documents as of something intermediate between non-clicked documents (or

clicked with an insignificant dwell time) and documents with a sufficiently long dwell time, so they could be treated as potentially relevant by our algorithm $\mathcal{A}$. Hence our confidence in the label that marks them as irrelevant can be lower than for documents with the dwell time around 5s.

Another important distinctive property of the implicit relevance labelling that can adopted in search personalization is the existence of two types of non-relevant samples. The results of the first type are *skipped*, i.e., their snippets or titles were examined by a user and the results were deliberately not clicked. The results of the second type are *not examined*, i.e., neither their snippets, nor the documents themselves, were examined by a user. Clearly, skipped documents have, a priori, larger confidence in their negative label than not examined documents. Since we do not know for sure whether a given result was examined or not by a user, further we follow the widespread assumption: a user examines all snippets up to the last clicked document. In what follows, all not clicked results ranked above the last clicked result are referred to as 'skips' and the remaining not clicked results are said to be 'not examined'.

At this point we would like to stress that we do not deal with the problem of improving the evaluation methodology of personalized search defined by equation (2). On the contrary, we assume that there is some fixed metric $\mathcal{M}$ defined by some state-of-the-art gain function $\mathcal{G}$, e.g., based on the fact of a click or the fact of a satisfied click, as in most of the studies on personalization. Such a labelling is typically established by simple rules, and the gain values it produces turn out to be noisy, as we have discussed in the beginning of this section. That is, not every document with zero/positive label is useless/useful for a user. We argue that direct utilization of these gain values *for training* a personalization algorithm leads to a suboptimal (according to the metric $\mathcal{M}$) ranking.

To address this weakness of the state-of-the-art personalization approaches, we suggest introduction of *weights*, i.e., we equip every sample $t_i \in \mathcal{S}$ with a certain positive number $w_i \in \mathbb{R}$, which reflects the confidence in the corresponding gain value and the need for a machine learning algorithm to take into account the gain for this particular example. Classically this approach is applied when the observed measurements (gain values $\mathcal{G}(t_i)$) have different uncertainties [2]: the closer the weight w_i to zero is, the less confident about the gain value $\mathcal{G}(t_i)$ we are.

For the empirical use, the appropriate weight values w_i are not known and must be estimated. The core idea of the present paper is adding of an extra step in the machine learning, see Section 4, which trains the weight values $w_i \in [0;1]$ in advance.

As we have discussed above, in the traditional personalization learning each sample $t_i \in \mathcal{S}$ is equipped with feature vector $\mathcal{X}(t_i) \in \mathbb{R}^N$. The fundamental advancement of our weight-based personalization learning is the utilization of additional information gathered along with standard gain values $\mathcal{G}(t)$: post-impression features reflecting a more complete picture of the interaction between the user and the search system that resulted into particular gain values. Among them, there are, for instance, the total number of clicks on SERP, the indicator of the fact the document d was clicked by a user, the dwell time of a click, etc. Assume that each sample $t_i \in \mathcal{S}$ in the training dataset is characterized by M-dimensional *weight feature vector* $\mathbf{y}_i = (y_i^1, \dots, y_i^M)$. Let $\mathcal{Y}$ be the corresponding map:

$$\mathcal{Y}\colon \mathcal{S} \to \mathbb{R}^M, \quad \mathcal{Y}\colon t_i \mapsto (y_i^1, \dots, y_i^M) \in \mathbb{R}^M.$$

We emphasize that these features cannot be calculated before the result page is formed, thus do not participate in the ranking model. However, they are possibly useful during the training of the ranker $\mathcal{F}$, since they implicitly carry the information about *degree of relevance* of documents in $\mathcal{S}$. Unfortunately, state-of-the-art search personalization algorithms do not allow employment of features $\mathcal{Y}(t_i)$. For this reason we suggest to learn a *weight function* $\mathcal{W}$ on the space of $\mathcal{Y}$-features.

Namely, let $\mathcal{W}$ be the function mapping the space of $\mathcal{Y}$-features to the weights:

$$\mathcal{W}\colon \mathbb{R}^M \to [0;1], \quad \mathcal{W}\colon \mathcal{Y}(t_i) \mapsto w_i.$$

Assume that we are given a machine learning algorithm $\mathcal{A}$, which takes into account weights of samples. An example of such algorithm is a linear model minimizing weighted sum of residual squares, discussed in Section 4. Given the weight function $\mathcal{W}$, gain function $\mathcal{G}$ and such a machine learning algorithm, we assume that one uniquely determines the ranker $\mathcal{F}_{\mathcal{W}}$. Our aim is to provide such weight function $\mathcal{W}$ that the ranker $\mathcal{F}_{\mathcal{W}}$ maximizes the overall ranking quality $\mathcal{M}(\mathcal{F}_{\mathcal{W}})$:

$$\widehat{\mathcal{W}} = \operatorname*{argmax}_{\mathcal{W}} \mathcal{M}(\mathcal{F}_{\mathcal{W}}). \tag{3}$$

In the next section we show that, under additional assumptions on the machine learning algorithm $\mathcal{A}$ and on the set of possible weight functions, this problem can be solved via steepest gradient descent.

4. ALGORITHM

4.1 Assumptions

In this section we explain how to learn the weight function $\mathcal{W}$ introduced in Section 3.2. To solve the optimization problem (3), one has to reduce the set of all possible functions $\mathcal{W}\colon \mathbb{R}^M \to \mathbb{R}$ to some finite-dimensional parametric family. In order to learn the parameters of $\mathcal{W}$, we need to know the impact of each parameter on the objective function $\mathcal{M} = \mathcal{M}(\mathcal{F}_{\mathcal{W}})$. If there are few parameters, one can perform an exhaustive search in the parameter space. However, the computational complexity of this exhaustive search grows exponentially with the growth of the number of parameters. Since each step of the exhaustive search requires separate training of an algorithm, which is usually resource consuming, exhaustive search quickly becomes infeasible with the growth of parameter space dimension. Ideally, we would like to start a gradient descent optimization of $\mathcal{M}$ with respect to parameters of the weight function $\mathcal{W}$.

Weight function

First we fix the class of weight functions we are considering. Although our framework is applicable to various parametric families of weight functions, further in this paper we deal only with *logistic* weight functions:

$$\mathcal{W}_\beta(\mathbf{y}) = \operatorname{sigmoid}(\mathbf{y}\cdot\beta), \quad \operatorname{sigmoid}(z) = \frac{1}{1+e^{-z}}, \tag{4}$$

where $\mathbf{y} = (y^1, \dots, y^M) \in \mathbb{R}^M$ is a row vector of $\mathcal{Y}$-features, $\beta = (\beta_1, \dots, \beta_M)^T$ is a column vector of parameters, and $\mathbf{y}\cdot\beta = \sum y^i\beta_i$. Here and below superscript ^{T} stands for

X	$S \times N$ matrix with personalization features, its i-th row is feature vector $\mathbf{x}_i = \mathcal{X}(t_i)$ of sample t_i
Y	$S \times M$ matrix with weight features, its i-th row is feature vector $\mathbf{y}_i = \mathcal{Y}(t_i)$ of sample t_i
β	M-dimensional column vector of weight parameters
W_β	diagonal $S \times S$ matrix with $\mathcal{W}_\beta(\mathbf{y}_i)$ at the i-th position
$\mathbf{b}$	N-dimensional column vector of ranker $\mathcal{F}$ parameters
$\mathbf{f}$	S-dimensional column vector of the ranker values, its i-th coordinate is $f_i = \mathcal{F}(\mathcal{X}(t_i))$
$\mathbf{g}$	S-dimensional column vector of the gain values, its i-th coordinate is $g_i = \mathcal{G}(t_i)$
μ	L_2-regularization parameter

Table 1: Matrix notations.

matrix transposition. The choice of the sigmoid transform is convenient, since it automatically guarantees that the weight function takes its values in the unit interval $(0; 1)$.

Learning-to-rank algorithm

In the most general setting, derivatives $\dfrac{\partial \mathcal{M}}{\partial \beta_i}$ required for steepest descend optimization could not be efficiently computed and even do not necessarily exist. To guarantee the existence of these derivatives and to be able to calculate them, we limit ourselves to the particular class of machine learning algorithms — *linear regression ranking models* — and use a smoothed variant of the performance measure $\mathcal{M}$: SoftRank smoothing [18].

Before we describe this algorithm, we fix *matrix* notations for some objects defined in Section 3 and introduce some additional notions specific for the algorithm, which are summarized in Table 1.

Linear ranker $\mathcal{F}$ is any ranker given by a linear function $\mathcal{F}\colon \mathbb{R}^N \to \mathbb{R}$, i.e., a function of the form

$$\mathcal{F}(\mathbf{x}) = \mathbf{x}\cdot\mathbf{b} = \sum_{i=1}^{N} x^i b_i, \quad \mathbf{x} = (x^1, \ldots, x^N), \tag{5}$$

where $\mathbf{b} = (b_1, \ldots, b_N)^T$ is a column vector of coefficients — parameters of the ranking function. Thus, in the matrix notation, the vector of ranker values is $\mathbf{f} = X \cdot \mathbf{b}$.

Regression learning is one of the possible pointwise approaches to the learning-to-rank problem. This type of algorithms predicts the gain value $g_i = \mathcal{G}(t_i)$ for each sample (point) $t_i \in \mathcal{S}$ separately. Typically, this is done by minimizing the *residual sum of squares*:

$$\mathrm{RSS} = \sum_{t_i \in \mathcal{S}} (g_i - f_i)^2 \to \min.$$

To incorporate weights into RSS optimization problem we use its weighted analogue: WRSS, by multiplying each residual square by the corresponding weight:

$$\mathrm{WRSS} = \sum_{t_i \in \mathcal{S}} w_i (g_i - f_i)^2 \to \min.$$

For the reduction of the overfitting to the training dataset we add L_2 regularization term to the WRSS minimization problem. The impact of the regularization on the final solution is controlled by the parameter $\mu \in \mathbb{R}$ (L_2RSS is defined analogously):

$$L_2\mathrm{WRSS} = \sum_{t_i \in \mathcal{S}} w_i \big(g_i - f_i\big)^2 + \mu ||\mathbf{b}||^2 \to \min \tag{6}$$

Further in the paper we assume that the machine learning algorithm $\mathcal{A}$ solves this minimization problem. We would like to point out that although the framework described below solves the weight optimization problem (3) only for linear regression algorithm, the learned weight function $\mathcal{W}$ itself could be used in a more complicated algorithms. For example, our experimental results in Section 6 prove that the weight function learned for a simple linear regression problem (solution to (3) for ranker (5)) significantly improves a decision tree model.

4.2 Gradient computation

In the case of L_2WRSS regression problem (6), we have $f_i = \mathbf{x}_i \cdot \mathbf{b}$ and linear learning has the explicit solution:

$$\mathbf{b} = (X^T W_\beta X + \mu \mathrm{Id})^{-1} X^T W_\beta \mathbf{g}; \tag{7}$$

the vector of ranker values is:

$$\mathbf{f} = X\mathbf{b} = X(X^T W_\beta X + \mu \mathrm{Id})^{-1} X^T W_\beta \mathbf{g}. \tag{8}$$

For our choice of the weight function, see (4), the elements of the matrix W_β smoothly depend on the parameter vector β, so together with equation (8) this gives smooth dependence of the vector $\mathbf{f}$ on parameters β, and we can compute all the derivatives $\dfrac{\partial \mathbf{f}}{\partial \beta_k}$ for $i = 1, \ldots, M$. Setting $Z_\beta = X^T W_\beta X + \mu \mathrm{Id}$ and differentiating the equality $Z_\beta \mathbf{b} = X^T W_\beta \mathbf{g}$ with respect to β_i, one obtains:

$$\frac{\partial \mathbf{b}}{\partial \beta_i} = Z_\beta^{-1} X^T \frac{\partial W_\beta}{\partial \beta_i}\mathbf{g} - Z_\beta^{-1}\frac{\partial Z_\beta}{\partial \beta_i}\mathbf{b} = \\ = Z_\beta^{-1} X^T \frac{\partial W_\beta}{\partial \beta_i}(\mathbf{g} - \mathbf{f}). \tag{9}$$

Here $\dfrac{\partial W_\beta}{\partial \beta_i}$ is the elementwise derivative of matrix W_β, i.e., the diagonal $S \times S$ matrix with $y_k^i \,\mathrm{sigmoid}'(\mathbf{y}_k \cdot \beta)$ at k-th position.

To finish the computation of the derivatives $\dfrac{\partial \mathcal{M}}{\partial \beta_i}$ we need to find $\dfrac{\partial \mathcal{M}}{\partial f_j}$. Although in the straightforward interpretation of the metric $\mathcal{M}$ described above, this derivative never exists, there are effective methods to cope with this problem. Indeed, ordering of documents $\mathcal{S}_{u,q}$ defined by the values of the ranking function $\mathcal{F}$ is locally constant and changes discontinuously as scores $f_i = \mathcal{F}(\mathcal{X}(t))$ vary, so does the objective function $\mathcal{M}$. A standard method of turning $\mathcal{M}$ into a smooth function is a probabilistic smoothing of the ranking measure, see [18, 20, 24]. Further we employ *SoftRank* approach from [18]. In a nutshell, this method replaces each score f_i with a random variable with a normal distribution with the mean f_i and computes the expectation value $E\mathcal{M}$ instead of the true value $\mathcal{M}$. We refer to [18, Section 3] for technical details and explicit formulas. By abuse of notation we continue writing simply $\mathcal{M}$ instead of $E\mathcal{M}$.

Now we have all ingredients to start steepest descend in the space of parameters β. Let $\beta^j \in \mathbb{R}^M$ be the vector of weight function parameters at j-th iteration. The pseudocode in Algorithm 1 summarizes iterative steepest descend learning of the weight function $\mathcal{W}_\beta$ in the case of linear ranker $\mathcal{F}$, see equations (5) and (6).

Input: feature matrices X, Y, vector of gain values $\mathbf{g}$
Parameters: number of iterations J, step size regularization ϵ, parameter of L_2 regularization μ
Initialization: $\beta^0 = (0, \dots, 0)^T$
for $j = 0$ *to* J **do**
  $\mathbf{f} = X(X^T W_{\beta^j} X + \mu \mathrm{Id})^{-1} X^T W_{\beta^j} \mathbf{g}$ (see (8));
  $\text{step} = \dfrac{\partial \mathcal{M}}{\partial \mathbf{f}} \cdot \dfrac{\partial \mathbf{f}}{\partial \beta}\Big|_{\beta=\beta^j} = \sum_{i=1}^{S} \dfrac{\partial \mathcal{M}}{\partial f_i} \cdot \dfrac{\partial f_i}{\partial \beta}\Big|_{\beta=\beta^j}$ (see (9));
  $\beta^{j+1} = \beta^j + \epsilon\,\text{step}$;
end
Output: optimized vector β^J

Algorithm 1: Steepest descend optimization of β.

4.3 Computational complexity

In this section we give an upper bound on the complexity of one iteration of Algorithm 1. In what follows, we assume that $M, N \ll S$, since in the personalization framework it is usually very easy to collect a training set of arbitrarily large size, and, typically, in studies on personalization [3, 4, 15], the number of samples in the training set is several orders of magnitude larger than the number of features. Under these assumptions we estimate the complexity of our algorithm with a sigmoid weight function $\mathcal{W}$ and a linear ranker $\mathcal{F}$ (see equations (4) and (5)).

To calculate the derivatives (9) we first compute the $S \times S$ matrix with the numbers $\text{sigmoid}'(\mathbf{y}_k \cdot \beta), k = 1 \dots, S$, on the diagonal. Then compute M matrices $\dfrac{\partial W_\beta}{\partial \beta_i}$ multiplying diagonal by numbers y_k^i and matrix W_β, which requires $O(MS)$ operations. After this, using the fact that matrix W_β is diagonal, we compute matrix Z_β, in $O(SN^2)$ operations.

Recall that computation of a product of two matrices of sizes $A \times B$ and $B \times C$ requires $O(ABC)$ operations, and finding a solution of a linear system of a size $A \times A$ requires $O(A^3)$ operations. Hence, to multiply right-hand sides of equations (8) and (9) from the right to left one requires

$$\begin{aligned} &O(S) + O(NS) + O(N^3) + O(NS) + \\ &\quad + M\big(O(S) + O(NS) + O(N^3) + O(NS)\big) = \\ &\qquad = O(SNM) + O(N^3 M) \end{aligned}$$

operations. Taking into account the fact that $M, N \ll S$, we obtain the estimate $O(SMN) + O(SN^2)$ operations to compute the vectors $\dfrac{\partial \mathbf{f}}{\partial \beta_i}$, see (9).

The complexity of the procedure of smoothing the objective function is described in [18, Section 3.6]. The computation of the derivatives $\dfrac{\partial M}{\partial f_j}$ for one user-query pair requires $O(k^3)$ operations, where k is the number of documents retrieved for a given query q (assumed to be constant across different queries in our case). So, the computation of all the derivatives of the objective function $\mathcal{M}$ over all scores $f_i, i = 1, \dots, S$ requires $\dfrac{S}{k} O(k^3) = O(Sk^2)$ operations.

All in all the computation of the gradients $\dfrac{\partial \mathcal{M}}{\partial \beta}$ requires:

$$O(SNM) + O(SN^2) + O(Sk^2)$$

operations. Note that the complexity of our framework is linear with respect to the number of samples in the training dataset and the number of iterations.

5. EXPERIMENTAL SETUP

In this section we describe the dataset and the features that are used to train both the personalized ranker and the weight function used in our learning framework.

5.1 Data

In the experiments we verify our theoretically grounded framework on the large-scale log of a popular commercial search engine. Namely, we use a sample from the publicly available dataset shared by Yandex with the participants of 'Personalized Web Search Challenge' on `Kaggle.com`.This dataset contains user sessions collected during 30 days including user ids, their queries, query terms, SERPs, and clicks. All queries with no clicks on the result page were eliminated. The sessions from the first 27 days were intended for training and are supplied with all available data, while the sessions from the last three days are not accompanied by information about clicks and were used to find the winners of the challenge. While the clicks for the latter sessions had not become publicly available even after the challenge, we use the former sessions for our experiments, also used by all participants of the challenge to train and test their solutions. The stream of user actions stored in the dataset is pre-segmented into *search sessions*. As described below, we use this segmentation to compute some of the weight features. All data including texts of queries are fully anonymized and encoded by numerical values. An original production ranking recorded in this dataset, which was presented to real users, will be further referred to as *non-personalized baseline*.

5.2 Personalization features

The core component of any personalization algorithm is the set of features $\mathcal{X}$ used for training a personalized ranker. Note that the specific choice of features $\mathcal{X}$ is out of the scope of our study, and the framework we suggest could be applied regardless of a feature set. However, to prove effectiveness of our methodology on the real data, we conduct our experiments by using a competitive collection of personalization features. Namely, we use the evaluation dataset provided by team `pampampampam`, the winner of the Personalized Web Search Challenge.

This evaluation dataset $\mathcal{S}$ consists of 280K queries uniformly sampled from the 27th day in the original raw Kaggle log for training and testing purposes. For each query-document pair, personalization features are computed based on the information available prior to the moment this query was issued (including the information from the first 26 days of the Kaggle dataset).

We briefly enlist the types of features extracted from the Yandex dataset. For every sample $\{u, q, d\} \in \mathcal{S}$ we used 165 features covering most of the existing sources of signal for personalization investigated in previous studies on personalization [4, 16, 21]. Since some data used in these studies is unavailable in our case, as we use the above mentioned publicly available dataset, we use simplified versions of some features, e.g., we do not have any information on *topics* of documents, *stopwords* in queries and IDF's (inverse document frequencies) of words.

- Features based on the user behavior aggregated over all users, e.g., various click-through rates of document d, the number of times document d was shown, the number of times document d was skipped.
- Features reflecting the short-term search context of a given query, e.g., the number of times the document was shown/clicked/skipped by the user in the current search session. These features are analogous to the **Query-Doc-User Features** from [4] based on URL-match with decay.
- Features intended for long-term personalization (see [4, 16]) computed during the first 26 days, e.g., the number of times the document/host was shown/clicked/skipped by the user in the past. These features are analogous to the **Query-Doc-User Features** from [4] based on URL-match without decay.
- Characteristics of query q, e.g., word count, query popularity, click entropy, sum of inverse frequencies of its terms. These features include all **Query features** from [4].
- The original position of document d assigned by the default non-personalized ranker (The **Query-Doc** feature from [4]).

During evaluation of our personalization system we perform the following version of cross-validation by proceeding the following splitting of the dataset five times. We divide the whole dataset $\mathcal{S}$ containing 2.8M samples into two parts on the basis of unique user id: (1) part $\mathcal{S}_1$ containing 1/3 of query-document pairs is used for learning of personalization models and weights as it is described in Algorithm 1; (2) the remaining part $\mathcal{S}_2$ is used for evaluation. We did this type of splitting following the state-of-the-art studies on personalization [3, 15, 19] to prevent user-specific overfitting and to ensure that the model we learned is applicable to users not seen in the training dataset. All performance values reported below are averaged over these five splittings.

5.3 Weight features

The core part of our experiments is the choice of the set of weight features $\mathcal{Y}$. These features characterise user behavior on the result page and are used only as components of the function $\mathcal{W}_\beta$ trained with the procedure described in Section 4. They do not participate in training a personalization ranker, since these features cannot be calculated before the search results are shown to the user. The basic features of user interaction with SERP are reported in Table 2. There are two types of features: the first ones depend on the particular query-document pair and the second ones depend on the whole SERP. All the non-binary features are

Feature	Description
Document level features	
click*	1 iff d was clicked
dwell	dwell time of a click, 0 if click(d)=0
long*	1 iff dwell(d) > 30
last*	1 iff click(d) = 1 and d is the last clicked document
first*	1 iff click(d) = 1 and d is the first clicked document
sat*	last(d) = 1 or long = 1
position	original position of the document
skip*	d was skipped
skipprev*	the previous document in the ranking list was skipped
skipabove	the number of skipped documents above d
SERP level features	
topclick	the highest position among all clicks
bottomclick	the lowest position among all clicks
numclick	total number of clicks on the result page
numclick3	number of clicks at top 3 results
numskips	total number of skipped documents on the result page
lastquery*	query q is the last query in the session
examtime	time to the first click on SERP

Table 2: Basic weight features (* indicates binary)

Feature	#	Discretization levels
dwell	10	0–5–15–30–50–90–150–300–750–3000–∞
position	10	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
skipabove	3	0, 1, 2–∞
numclick	2	1, 2–∞
numclick3	4	0, 1, 2, 3
numskips	3	0, 1, 2–∞
examtime	10	0–5–15–30–50–90–150–300–750–3000–∞

Table 3: Discretization of non-binary features.

Feature	#
click=x AND numclick=y AND position=1	4
click=x AND numclick=y AND position>1	4
last=1 AND numclick=x	2
first=1 AND numclick=x	2
skip=0 AND click=0	1
skipprev=1 AND click=1	1

Table 4: Composite weight features and their counts.

turned into the binary ones as it is reported in Table 3. The set of binary features we choose was motivated by the basic binary features used in [26] for improving the statistical power of interleaving algorithms. From the binary features we combine more complex ones as described in Table 4. To sum up, we take binary features from Tables 2 and 3 and the ones presented in Table 4 (64 features in total).

Besides improving the quality of the personalized ranker, we expect that the values of weight function will help answering the following questions:

Q1 How does the time the user has spent on the clicked document d affect the confidence in its positive gain value?

Q2 How does *positional bias* of users' clicks affect the weights of the gain values?

Q3 Do all the non-clicked documents indicate non-relevance to the same extent?

6. RESULTS

In this section we report the results of various experiments we conduct on training weight functions $\mathcal{W}$ on the Yandex dataset. Below we compute re-ranking performances on all the queries in the test set, which, in particular, includes many of the queries not affected by the personalization. Although averaging improvements over all queries, instead of affected ones, decreases change in MRR*, it also provides more insight into the overall effect of our method.

We conclude this section by analyzing the learned weight function in the scope of the stated research questions and provide some insight into interpreting the user behavior.

6.1 Evaluation methodology

To evaluate the quality of our framework we used five simplest and widely used [1, 9, 22], 'basic' binary gains $\mathcal{G}$:

click, first click, last click, long click, satisfied click.

We fix the positional discount function: $\text{discount}(k) = 1/k$ and compute the corresponding objective measures as in equations (1) and (2) for each of the gains. Further these five performance metrics are referred to as MRR* (where * stands for one of the gains: click, first, last, long or sat).

Relying on the unique user identifier, we randomly divide part $\mathcal{S}_1$ into three equal parts: $\mathcal{S}_1 = \mathcal{S}_1^1 \sqcup \mathcal{S}_1^2 \sqcup \mathcal{S}_1^3$. First, we adjust the regularization parameter and use it in the subsequent experiments: $\mu = \text{argmax}_\mu \mathcal{M}(\mathcal{F})$, where the performance metric $\mathcal{M}$ is computed over $\mathcal{S}_1^2$ and the ranker $\mathcal{F}$ is trained via L_2-regularized RSS linear model on S_1^1. Step regularization parameter ϵ was set to the trivial default value $\epsilon = 1$ and was not tuned. Smaller values of ϵ could possibly improve the quality of our framework, however, at the same time, they would increase the optimal number of iterations and thus, the runtime of the steepest descent algorithm. After that we run Algorithm 1 and find the best iteration count according to the performance $\mathcal{M}(\mathcal{F}_\mathcal{W})$ measured on the set $\mathcal{S}_1^2$. The resulting ranker trained on the set $\mathcal{S}_1^1$ is evaluated on the set $\mathcal{S}_2$.

The remaining set $\mathcal{S}_1^3$ is further used to tune the parameters of the decision tree model, see Section 6.3.

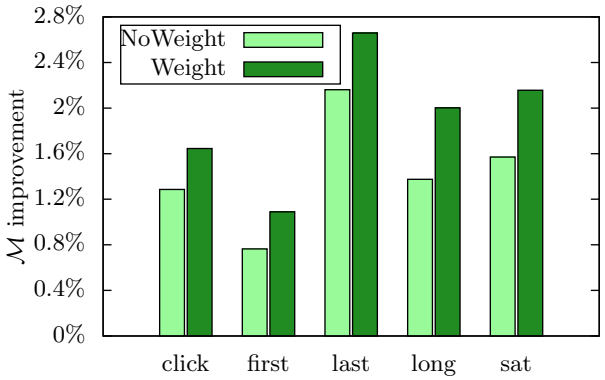

Figure 1: Relative improvements of MRR* of various gains for linear models trained without and with weights over the non-personalized baseline.

	MRR* improvement in %				
WRSS \ gain $\mathcal{G}$	click	first	last	long	sat
$\mathcal{W}$ for 'first'	+1.17	**+0.98**	+1.54	+1.24	+1.32
$\mathcal{W}$ for gain $\mathcal{G}$	**+1.68**	**+0.98**	**+2.66**	**+2.00**	**+2.16**

Table 5: MRR* improvements over the non-personalized baseline of the linear rankers learned with weights trained for 'first' gain and for the corresponding gain.

6.2 Linear model

In this section we compare two personalization algorithms: linear regression model learned with the identical weights (L_2RSS) and linear regression model learned with the weight function optimized with our framework (L_2WRSS). The improvements of both learning algorithms over non-personalized baseline according to five binary gains are demonstrated in Figure 1. For every gain, weighted L_2WRSS model significantly outperforms the corresponding L_2RSS model.

To demonstrate that the weight function $\mathcal{W}$ trained with our framework essentially depends on the choice of the evaluation metric $\mathcal{M}$, and, therefore, on the gain function $\mathcal{G}$, we also conduct the following experiment. We trained weight function $\mathcal{W}$ relying on the 'first' gain, and the corresponding ranker $\mathcal{F}_\mathcal{W}$ was evaluated according to the other basic gains.

In Table 5 we report the performance of this ranker in comparison with the rankers trained by utilizing weight functions optimized by Algorithm 1 for the corresponding gains. One can see that for 'click' 'last', 'long' and 'sat' gains the weight function trained for the 'first' gain is significantly outperformed. This proves that the choice of the performance measure strongly affects the weight function provided by our algorithm. Another argument supporting this claim will be discussed in Section 6.4.

6.3 Decision tree model

To understand whether the weights provided by our framework are specific for linear machine learning algorithms only or could be used for more complicated models, we used them at the top of a proprietary implementation of Friedman's gradient boosted decision tree-based machine learning algorithm [7] with the WRSS loss function.

Alg.\ gain $\mathcal{G}$	MRR* improvement in % click	first	last	long	sat
RSS	+1.95	+1.50	+2.75	+2.11	+2.22
WRSS	**+2.02**	**+1.60**	**+2.81**	**+2.23**	**+2.34**

Table 6: MRR* improvements over the non-personalized baseline of the decision-tree rankers learned without (RSS) and with (WRSS) trained weights. All WRSS rankers significantly ($p<0.05$) outperform RSS rankers according to the Student t-test.

Alg.	Fraction of queries in% meas diff	plus	minus	plus/minus
RSS	26.94	15.65	11.30	1.38
WRSS	28.50	16.67	11.83	1.41

Table 7: Fractions of affected ('meas diff'), improved ('plus'), deteriorated ('minus') queries and robustness rate ('plus/minus') for the decision-tree rankers learned for 'sat' gain without (RSS) and with (WRSS) trained weights.

Namely, given the gain function $\mathcal{G}$, we train a weight function, as in Section 6.2. For the trained weights, we learn the gradient boosted decision tree model on the set $\mathcal{S}_1^1 \sqcup \mathcal{S}_1^2$. Note that in Section 6.1 we have trained weights w_i using the same dataset $\mathcal{S}_1^1 \sqcup \mathcal{S}_1^2$, this ensures that the model trained with the use of our weights has accessed the same set of labelled samples as the model trained using baseline identical weights.

We would like to point out that the decision tree training model based on RSS loss function we use in this section coincides with the training model used by the winners of the Personalized Web Search Challenge. Thus it represents very strong personalization benchmark and any statistically significant improvement is valuable and important.

The resulting improvements are summarized in Table 6. Although the corresponding weight functions were trained under the assumption that machine learning algorithm $\mathcal{A}$ is linear (see Section 4.1), the utilization of these weights inside the decision tree model significantly improves its performance for all five basic gains. This indicates, that trained weights are universal and, being applicable to any algorithm based on minimizing WRSS loss function, they improve various machine learning algorithms.

For a better understanding of the magnitude of our improvements, we compare them with other studies on personalization. Most papers [3, 4, 5, 15, 22] report improvement of the order of a tenth of a percent up to a few percent *over the non-personalized baseline*. However, all these papers employ some *new sources of data* for personalization, e.g., short-term and long-term history, task-aggregation, geographical data, browsing data, etc. On the contrary, in our experiments we (1) achieve the improvements by changing the machine learning framework only, rather then by incorporating new signals for personalization and (2) evaluate our methodology against a very strong personalization baseline. This observation proves that weight learning framework indeed a notable improvement of the existing approaches.

Another important property of any re-ranking algorithm is its *robustness*. The fractions of affected queries, i.e., measurably different from the baseline performance, improved, deteriorated queries are common characteristic of re-ranking models, see, e.g, [3, 11, 16]. A robust algorithm not only improves the average quality, but also does not "hurt much", i.e., has low fraction of deteriorated and improved queries. From Table 7 one can see that the personalization model learned with weights affects more queries and has a better improvement/deterioration ratio.

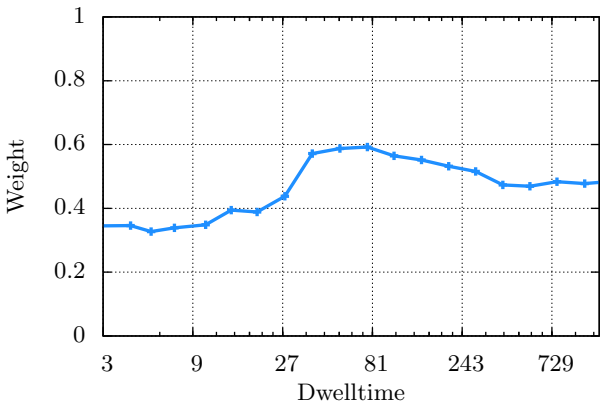

Figure 2: Average trained weight (for satisfied clicks gain) of clicked documents as a function of dwell time.

To conclude this section, we would like to mention that web search personalization is known as one of the most challenging learning tasks. For instance, the additional improvement of 0.12% over the baseline (e.g., the same as the WRSS achieved over RSS according to MRR with 'sat' gain) would allow the team that took the 3rd place in Kaggle competition become a winner[3]. Given that, by applying our framework, we managed to improve even the ranking system that won the challenge, we consider this improvement as an important and convincing motivation for opening a new research direction of optimizing weight functions of personalized labels introduced in our paper.

6.4 Analyzing weight function

Now we address the research question Q1 by analyzing the effect the dwell time has on the trained weight values w_i. Dwell time is a common measure of conversion, so it has been widely studied from different perspectives [10]. Figure 2 depicts the average weight of documents with a given interval of dwelltime on the logscale. One could conclude that the average weight values increase in the neighbourhood of point 30, justifying this threshold for satisfied clicks. In fact, it works in the opposite direction. Namely, these weights dramatically increase at 30, since they were optimized for a metric MRR-SAT, where only satisfied clicks are relevant. This observation again proves (as Table 5) that the weight function essentially depends on the choice of the evaluation metric $\mathcal{M}$. We stress that choice of the 'right' metric $\mathcal{M}$ lies out of the scope of our study and could not be done solely using data available in our experiments.

To answer both questions Q2 and Q3, we demonstrate the average of the trained weight values over clicked/skipped/not examined documents ranked at the various fixed positions in Figure 3. The weights of a skipped

[3]http://www.kaggle.com/c/yandex-personalized-web-search-challenge/leaderboard

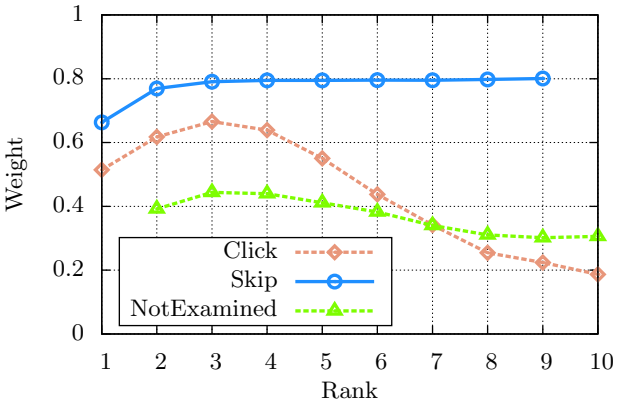

Figure 3: Average trained weight (for satisfied clicks gain) of clicked, skipped and not examined documents as a function of ranking position.

document at position 10 and a not examined document at position 1 are not presented, since (by definition) there are no such documents.

As we have expected, the average confidence in the negative label of 'skipped' documents is noticeably larger than of 'not examined' ones. Interestingly, confidence in the positive label of clicked documents increases up to position 3 and decreases after it. There are two possible explanations of this behavior. The first one is related to *positional bias*: users tend to click on the top-ranked documents despite their relevance, so clicks at first positions are often noisy. The second effect is that the queries with clicks at low ranked documents are typically complex and ambiguous so do not have a single relevant result, unlike, e.g., navigational queries. Confidence in the relevance of a particular result retrieved in a response to such complex query is lower.

7. FUTURE WORK

Improving personalized ranking is the focus of our study. However, our framework and the tuned weight function itself could be applied in a wide variety of settings, and, hence, there are various directions for future work.

Given a set of query-document pairs equipped with editorial labels, we can use the method proposed in our study to learn confidence levels of these labels, taking into account query, document and assessor features. Besides it, the weights we learn potentially provide a valuable signal for automatic evaluation of search engine performance. Namely, similarly to classical click-based metrics like MRR and MAP, it is possible to use measures of the form (1) that would incorporate the tuned weights w_i into the gain function $\mathcal{G}$. At last, as we have briefly discussed in Section 6.4, the weight values and coefficients β_i of the trained weight functions $\mathcal{W}_\beta$ encode important information about user's behavior patterns and provide useful insights for its modelling and analysis.

Besides diverse possible applications of our framework, there is a large room for its development and improvement. Probably the most essential direction for future research is to extend our algorithm, which is primarily intended for linear regression models, to more complicated machine learning models, e.g., neural networks and decision trees. The gap between improvements demonstrated in the linear and decision tree model cases, see Figure 1 and Table 6, suggests that the solution to this *extension problem* will likely substantially improve the overall effectiveness of our framework.

8. CONCLUSION

In this paper, following the most studies on personalization of web search, we deal with a training of a personalized ranker on the basis of user clicks. The state-of-the-art approaches to personalization employ very simple and noisy implicit personal relevance labels, missing vast amount of useful information. We have presented an optimization framework for training the confidence levels of these labels. Given a click-through dataset and a fixed evaluation methodology, our algorithm extracts weights of relevance labels from various features characterising user interaction with the result page. These weights could be utilized in the majority of machine learning algorithms.

Our experiments with the large-scale publicly-available dataset demonstrate that rankers trained via both simple linear models and the state-of-the-art decision trees machine learning algorithms significantly benefit from the employment of the trained weight function.

Acknowledgements

We are grateful to the team `pampampampam` and its members, who have shared their learning dataset with us. Also we would like to thank Igor Kuralenok for valuable discussions and advice on this research.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 19–26, New York, NY, USA, 2006. ACM.

[2] A. Aitken. On least squares and linear combination of observations. volume 55 of *Proc. R. Soc. Edinb*, pages 42–48, 1934.

[3] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 135–144, New York, NY, USA, 2011. ACM.

[4] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 185–194, New York, NY, USA, 2012. ACM.

[5] K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 403–412, New York, NY, USA, 2011. ACM.

[6] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search

strategies. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 581–590, New York, NY, USA, 2007. ACM.

[7] J. H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, Feb. 2002.

[8] D. Jiang, K. W.-T. Leung, and W. Ng. Context-aware search personalization with concept preference. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 563–572, New York, NY, USA, 2011. ACM.

[9] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[10] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 379–386, New York, NY, USA, 2010. ACM.

[11] N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 25–34, New York, NY, USA, 2011. ACM.

[12] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 43–52, New York, NY, USA, 2008. ACM.

[13] P. Serdyukov, G. Dupret, and N. Craswell. Log-based personalization: The 4th web search click data (wscd) workshop. WSDM '14, pages 685–686, New York, NY, USA, 2014. ACM.

[14] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 43–50, New York, NY, USA, 2005. ACM.

[15] M. Shokouhi, R. W. White, P. Bennett, and F. Radlinski. Fighting search engine amnesia: Reranking repeated results. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 273–282, New York, NY, USA, 2013. ACM.

[16] D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. WSDM '12, pages 433–442, New York, NY, USA, 2012. ACM.

[17] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 718–723, New York, NY, USA, 2006. ACM.

[18] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: Optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 77–86, New York, NY, USA, 2008. ACM.

[19] Y. Ustinovskiy and P. Serdyukov. Personalization of web-search using short-term browsing context. In *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '13, pages 1979–1988, New York, NY, USA, 2013. ACM.

[20] M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1089–1096, New York, NY, USA, 2009. ACM.

[21] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1009–1018, New York, NY, USA, 2010. ACM.

[22] R. W. White, W. Chu, A. Hassan, X. He, Y. Song, and H. Wang. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1411–1420, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[23] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 451–458, New York, NY, USA, 2010. ACM.

[24] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 107–114, New York, NY, USA, 2008. ACM.

[25] L. Xu, M. White, and D. Schuurmans. Optimal reverse prediction: A unified perspective on supervised, unsupervised and semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1137–1144, New York, NY, USA, 2009. ACM.

[26] Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 507–514, New York, NY, USA, 2010. ACM.