Empirical Exploitation of Click Data for Task Specific Ranking

Anlei Dong Yi Chang Shihao Ji Ciya Liao Xin Li Zhaohui Zheng
Yahoo! Labs
701 First Avenue
Sunnyvale, CA 94089
{anlei,yichang,shihao,ciyaliao,xinli,zhaohui}@yahoo-inc.com

Abstract

There have been increasing needs for task specific rankings in web search such as rankings for specific query segments like long queries, time-sensitive queries, navigational queries, etc; or rankings for specific domains/contents like answers, blogs, news, etc. In the spirit of "divide-and-conquer", task specific ranking may have potential advantages over generic ranking since different tasks have task-specific features, data distributions, as well as feature-grade correlations. A critical problem for the task-specific ranking is training data insufficiency, which may be solved by using the data extracted from click log. This paper empirically studies how to appropriately exploit click data to improve rank function learning in task-specific ranking. The main contributions are 1) the exploration on the utilities of two promising approaches for click pair extraction; 2) the analysis of the role played by the noise information which inevitably appears in click data extraction; 3) the appropriate strategy for combining training data and click data; 4) the comparison of click data which are consistent and inconsistent with baseline function.

1 Introduction

Learning-to-rank approaches (Liu, 2008) have been widely applied in commercial search engines, in which ranking models are learned using labeled documents. Significant efforts have been made in attempt to learn a generic ranking model which can appropriately rank documents for all queries . However, web users' query intentions are extremely heterogeneous, which makes it difficult for a generic ranking model to achieve best ranking results for all queries. For this reason, there have been increasing needs for task specific rankings in web search such as rankings for specific query segments like long queries, time-sensitive queries, navigational queries, etc; or rankings for specific domains/contents like answers, blogs, news, etc. Therefore, a specific ranking task usually correspond to a category of queries; when the search engine determines that a query is belonging to this category, it will call the ranking function dedicated to this ranking task. The motivation of this divide-and-conquer strategy is that, task specific ranking may have potential advantages over generic ranking since different tasks have task-specific features, data distributions, as well as feature-grade correlations.

Such a dedicated ranking model can be trained using the labeled data belonging to this query category (which is called dedicated training data). However, the amount of training data dedicated to a specific ranking task is usually insufficient because human labeling is expensive and time-consuming, not to mention there are multiple ranking tasks that need to be taken care of. To deal with the training data insufficiency problem for task-specific ranking, we propose to extract click-through data and incorporate it with dedicated training data to learn a dedicated model.

In order to incorporate click data to improve the ranking for a dedicate query category, it is critical to fully exploit click information. We empirically explore the related approaches for the appropriate click data exploitation in task-specific rank function learning. Figure 1 illustrates the procedures and critical components to be studied.

1) Click data mining: the purpose is to extract informative and reliable users' preference information from click log. We employ two promising approaches: one is heuristic rule approach, the other is sequential supervised learning approach.

2) Sample selection and combination: with labeled training data and unlabeled click data, how

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1086–1095,
Singapore, 6-7 August 2009.

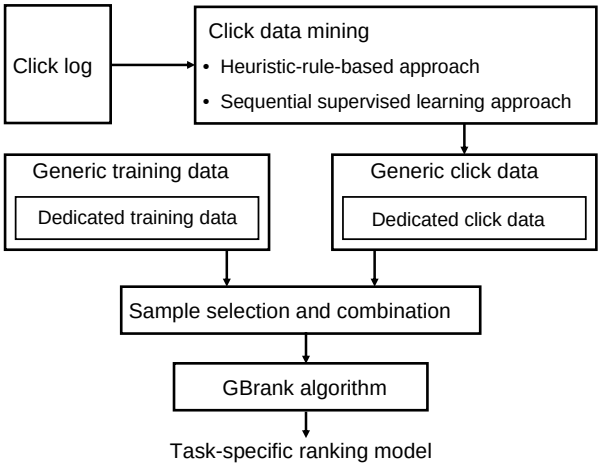

Figure 1: Framework of incorporating click-through data with training data to improve dedicated model for task-specific ranking.

to select and combine them so that the samples have the best utility for learning? As the data distribution for a specific ranking task is different from the generic data distribution, it is natural to select those labeled training samples and unlabeled click preference pairs which belong to this query category, so that the data distributions of training set and testing set are consistent for this category. On the other hand, we should keep in mind that: a) non-dedicated data, i.e, the data that does not belong the specific category, might also have similar distribution as the dedicated data. Such distribution similarity makes non-dedicated data also useful for task-specific rank function learning, especially for the scenario that dedicated training samples is insufficient. b) The quality of dedicated click data may be not as reliable as human labeled training data. In other words, there are some extracted click preference pairs that are inconsistent with human labeling while we regard human labeling as correct labeling.

3) Rank function learning algorithm: we use GBrank (Zheng et al., 2007) algorithm for rank function learning, which has proved to be one of the most effective up-to-date learning-to-rank algorithms; furthermore, GBrank algorithm also takes preference pairs as inputs, which will be illustrated with more details in the paper.

2 Related work

Learning to rank has been a promising research area which continuously improves web search relevance (Burges et al., 2005) (Zha et al., 2006) (Cao et al., 2007) (Freund et al., 1998) (Friedman, 2001) (Joachims, 2002) (Wang and Zhai, 2007) (Zheng et al., 2007). The ranking problem is usually formulated as learning a ranking function from preference data. The basic idea is to minimize the number of contradicted pairs in the training data, and different algorithm cast the preference learning problem from different point of view, for example, RankSVM (Joachims, 2002) uses support vector machines; RankBoost (Freund et al., 1998) applies the idea of boosting from weak learners; GBrank (Zheng et al., 2007) uses gradient boosting with decision tree; RankNet (Burges et al., 2005) uses gradient boosting with neural net-work. In (Zha et al., 2006), query difference is taken into consideration for learning effective retrieval function, which leads to a multi-task learning problem using risk minimization framework.

There are a few related works to apply multiple ranking models for different query categories. However, none of them takes click-through information into consideration. In (Kang and Kim, 2003), queries are categorized into 3 types, informational, navigational and transactional, and different models are applied on each query category.

a KNN method is proposed to employ different ranking models to handle different types of queries (Geng et al., 2008). The KNN method is unsupervised, and it targets to improve the overall ranking instead of the rank-ing for a certain query category. In addition, the KNN method requires all feature vector to be the same.

Quite a few research papers explore how to obtain useful information from click-through data, which could benefit search relevance (Carterette et al., 2008) (Fox et al., 2005) (Radlinski and Joachims, 2007) (Wang and Zhai, 2007). The information can be expressed as pair-wise preferences (Chapelle and Zhang, 2009) (Ji et al., 2009) (Radlinski et al., 2008), or represented as rank features (Agichtein et al., 2006). Task-specific ranking relies on the accuracy of query classification. Query classification or query intention identification has been extensively studied in (Beitzel et al., 2007) (Lee et al., 2005) (Li et al., 2008) (Rose and Levinson, 2004). How to combine editorial data and click data is well discussed in (Chen et al., 2008) (Zheng et al., 2007). In addition, how to use click data to improve ranking are also exploited in personalized or preference-based search (Coyle

Table 1: Statistics of click occurrences for heuristic rule approach.

imp	impression, number of occurrence of the tuple
cc	number of occurrence of the tuple where two documents both get clicked
ncc	number of occurrence of the tuple where url_1 is not clicked but url_2 is clicked
cnc	number of occurrence of the tuple where url_1 is clicked but url_2 is not clicked
$ncnc$	number of occurrence of the tuple where url_1 and url_2 are not clicked

and Smyth, 2007) (Glance, 2001) (R. Jin, 2008).

3 Technical approach

This section presents the related approaches in Figure 1. In Section 4, we will make deeper analysis based on experimental results.

3.1 Click data mining

We use two approaches for click data mining, whose outputs are preference pairs. A *preference pair* is defined as a tuple $\{< x_q, y_q > | x_q \succ y_q\}$, which means for the query q, the document x_q is more relevant than y_q. We need to extract informative and reliable preference pairs which can be used to improve rank function learning.

3.1.1 Heuristic rule approach

We use heuristic rules to extract skip-above pairs and skip-next pairs, which are similar to Strategy 1 (click $>$ skip above) and Strategy 5 (click $>$ no-click next) proposed in (Joachims et al., 2005). To reduce the misleading effect of an individual click behavior, click information from different query sessions is aggregated before applying heuristic rules. For a tuple $(q, \text{url}_1, \text{url}_2, \text{pos}_1, \text{pos}_2)$ where q is query, url_1 and url_2 are urls representing two documents, pos_1 and pos_2 are ranking positions for the two documents with $\text{pos}_1 \prec \text{pos}_2$ meaning url_1 has higher rank than url_2, the statistics for this tuple are listed in Table 1.

Skip-above pair extraction: if ncc is much larger than cnc, and $\frac{cc}{imp}$, $\frac{ncnc}{imp}$ is much smaller than 1, that means, when url_1 is ranked higher than url_2 in query q, most users click url_2 but not click url_1. In this case, we extract a skip-above pair, i.e., url_2 is more relevant than url_1. In order to have highly accurate skip-above pairs, a set of thresholds are applied to only extract the pairs that have high impression and ncc is larger enough than cnc.

Skip-next pair extraction: if $\text{pos}_1 = \text{pos}_2 - 1$, cnc is much larger than ncc, and $\frac{cc}{imp}$, $\frac{ncnc}{imp}$ is much smaller than 1, that means, in most of cases when url_2 is ranked just below url_1 in query q, most users click url_1 but not click url_2. In this case, we regard this tuple as a skip-next pair.

Table 2: Skip-above pairs count vs. human judgements (e.g., the element in the third row and second column means we have 40 skip-above pairs with "excellent" url_1 and "perfect" url_2). P: perfect; E: excellent; G: good; F: fair; B: bad.

	P	E	G	F	B
P	13	13	12	4	0
E	40	44	16	2	2
G	27	53	103	29	8
F	10	15	43	27	5
B	4	4	11	20	14

Table 3: Skip-next pairs vs. human judgements (e.g., the element in the third row and second column means we have 10 skip-next pairs with "excellent" url_1 and "perfect" url_2). P: perfect; E: excellent; G: good; F: fair; B: bad.

	P	E	G	F	B
P	126	343	225	100	35
E	10	71	84	37	12
G	6	9	116	56	21
F	1	5	17	29	14
B	1	1	1	2	5

To test the accuracy of preference pairs, we ask editors to judge some randomly selected pairs from skip-above pairs and skip-next pairs. Editors label each query-url pair using five grades according to relevance: perfect, excellent, good, fair, bad. Table 2 shows skip-above pair distribution. The diagonal elements have high values, which are for tied pairs labeled by editors but determined as skip-above pairs from heuristic rules. Higher values appear in the left-bottom triangle than in the right-top triangle, because there are more skip-above preferences agreed with editors than disagreed with editors. Summing up the tied pairs, agreed and disagreed pairs, 44% skip-above preference judgments agree with editors, 18% skip-above preference judgments disagree with editors,

and there are 38% skip-above pairs judged as tie pairs by editors.

Table 3 shows skip-next pair distribution. Summing up the tied pairs, agreed and disagreed pairs, 70% skip-next preference judgments agree with editors, 4% skip-next preference judgments disagree with editors, and 26% skip-next pairs judged as tie pairs by editors.

Therefore, skip-next pairs have much higher accuracy than skip-above. That is because in a search engine that already has a good ranking function, it is much easier to find a correct skip-next pairs which are consistent with the search engine than to find a correct skip-above pairs which are contradictory to the search engine. Skip-above and skip-next preferences provide us two kinds of users's feedbacks which are complementary: skip-above preferences provide us the feedback that the user's vote is contradictory to the current ranking, which implies the current relative ranking should be reversed; skip-next preferences shows that the user's vote is consistent with the current ranking, which implies the current relative ranking should be maintained with high confidence provided by users' vote.

3.1.2 Sequential supervised learning

The click modeling by sequential supervised learning (SSL) was proposed in (Ji et al., 2009), in which user's sequential click information is exploited to extract relevance information from click-logs. This approach is reliable because 1) the sequential click information embedded in an aggregation of user clicks provides substantial relevance information of the documents displayed in the search results, and 2) the SSL is supervised learning (i.e., human judgments are provided with relevance labels for the training).

The SSL is formulated in the framework of global ranking (Qin et al., 2008). Let $\mathbf{x}^{(q)} = \{x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}\}$ represent the documents retrieved with a query q, and $\mathbf{y}^{(q)} = \{y_1^{(q)}, y_2^{(q)}, \ldots, y_n^{(q)}\}$ represent the relevance labels assigned to the documents. Here n is the number of documents retrieved with q. Without loss of generality, we assume that n is fixed and invariant with respect to different queries. The SSL determines to find a function F in the form of $\mathbf{y}^{(q)} = F(\mathbf{x}^{(q)})$ that takes all the documents as its inputs, exploiting both local and global information among the documents, and predict the relevance labels of all the document *jointly*. This is distinct to most of learning to rank methods that optimize a ranking model defined on a single document, i.e., in the form of $y_i^{(q)} = f(x_i^{(q)})$, $\forall\, i = 1, 2, \ldots, n$. This formulation of the SSL is important in extracting relevance information from user click data since users' click decisions among different documents displayed in a search session tend to rely not only on the relevance judgment of a single document, but also on the *relative* relevance comparison among the documents displayed; and the global ranking framework is well-formulated to exploit both local and global information from an aggregation of user clicks.

The SSL aggregates all the user sessions for the same query into a tuple <query, n-document list, and an aggregation of user clicks>. Figure 2 illustrates the process of feature extraction from an aggregated session, where $\mathbf{x}^{(q)} = \{x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}\}$ denotes a sequence of feature vectors extracted from the aggregated session, with $x_i^{(q)}$ representing the feature vector extracted for document i. Specifically, to form feature vector $x_i^{(q)}$, first a feature vector $x_{i,j}^{(q)}$ is extracted from each user j's click information, and $j \in \{1, 2, \ldots\}$, then $x_i^{(q)}$ is formed by averaging over $x_{i,j}^{(q)}$, $\forall j \in \{1, 2, \ldots\}$, i.e., $x_i^{(q)}$ is actually an aggregated feature vector for document i. Table 4 lists all the features used in the SSL modeling. Note that some features are statistics independent of temporal information of the clicks, such as "Position" and "Frequency", while other features reply on their surrounding documents and the click sequences. We use 90,000 query-url pairs to train the SSL model, and 10,000 query-url pairs for best model selection.

With the sequential click modeling discussed above, several sequential supervised algorithms, including the conditional random fields (CRF) (Lafferty et al., 2001), the sliding window method and the recurrent sliding window method (Dietterich, 2002), are explored to find a global ranking function F. We omit the details but refer one to (Ji et al., 2009). The emphasis here is on the importance to adapt these algorithms to the ranking problem.

After training, the SSL model can be used to predict the relevance labels of all the documents in a new aggregated session, and thus pair-wise preference data can be extracted, with the score difference representing the confidence of preference

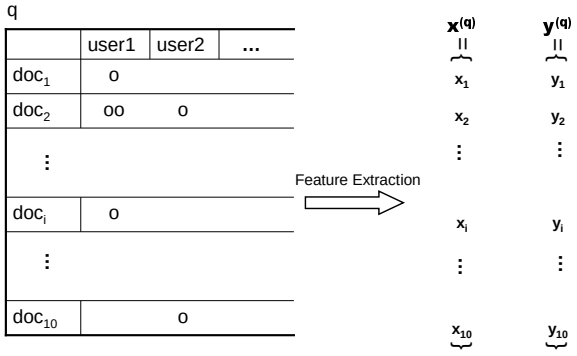

Figure 2: An illustration of feature extraction for an aggregated session for SSL approach. $\mathbf{x}^{(q)}$ denotes an extracted sequence of feature vectors, and $\mathbf{y}^{(q)}$ denotes the corresponding label sequence that is assigned by human judges for training.

Table 4: Click features used in SSL model.

Position	Position of the document in the result list
ClickRank	Rank of 1st click of doc. in click seq.
Frequency	Average number of clicks for this doc.
FrequencyRank	Rank in the list sorted by num. of clicks
IsNextClicked	1 if next position is clicked, 0 otherwise
IsPreClicked	1 if previous position is clicked, 0 otherwise
IsAboveClicked	1 if there is a click above, 0 otherwise
IsBelowClicked	1 if there is a click below, 0 otherwise
ClickDuration	Time spent on the document

prediction. For the reason of convenience, we also call the preference pairs contradicting with production ranking as skip-above pairs and those consistent with production ranking as skip-next pairs, so that we can analyze these two types of preference pairs respectively.

3.2 Modeling algorithm

The basic idea of GBrank (Zheng et al., 2007) is that if the ordering of a preference pair by the ranking function is contradictory to this preference, we need to modify the ranking function along the direction by swapping this prefence pair. Preferences pairs could be generated from labeled data, or could be extracted from click data. For each preference pair $< x, y >$ in the available preference set $S = \{< x_i, y_i > | x_i \succ y_i, i = 1, 2, ..., N\}$, x should be ranked higher than y. In GBrank algorithm, the problem of learning ranking functions is to compute a ranking function h , so that h matches the set of preference, i.e, $h(x_i) \geq h(y_i)$, if $x \succ y$, $i = 1, 2, ..., N$ as many as possible. The following loss function is used to measure the risk of a given ranking function h.

$$R(h) = \frac{1}{2}\sum_{i=1}^{N}(\max\{0, h(y_i) - h(x_i) + \tau\})^2, \quad (1)$$

where τ is the margin between the two documents in the pair. To minimize the loss function, $h(x)$ has to be larger than $h(y)$ with the margin τ, which can be chosen as constant value, or as dynamic values varying with pairs. When pair-wise judgments are extracted from editors' labels with different grades, pair-wise judgments can include grade difference, which can further be used as margin τ. The GBrank algorithm is illustrated in Algorithm 1, and two parameters need to be determined: the shrinkage factor η and the number of iteration.

Algorithm 1 GBrank algorithm.

Start with an initial guess h_0, for $m = 1, 2, ...$

1. Construct a training set: for each $< x_i, y_i > \in S$, derive $(x_i, \max\{0, h_{m-1}(y_i) - h_{m_1}(x_i) + \tau\})$, and
$(y_i, -\max\{0, h_{m-1}(y_i) - h_{m_1}(x_i) + \tau\})$.
2. Fit h_m by using a base regressor with the above training set.
3. $h_m = h_{m-1} + \eta s_m h_m(x)$, where s_m is found by line search to minimize the object function, η is shrinkage factor.

3.3 Sample selection and combination

We use a straightforward approach to learn ranking model from the combined data, which is illustrated in Algorithm 2.

Algorithm 2 Learn ranking model by combining editorial data and click preference pairs.

Input:
- Editorial absolute judgement data.
- Preference pairs from click data.

1. Extract preference pairs from labeled data with absolute judgement.
2. Select and combine preference pairs from click data and labeled data.
3. Learn GBrank model from the combined preference pairs.

Absolute judgement on labeled data contains (query, url) pairs with absolute grade values labeled by human. In Step 1, for each query with

n_q query-url pairs with corresponding grades, $\{<$ query, $\text{url}_i, \text{grade}_i > | i = 1, 2, \ldots, n_q\}$, its preference pairs are extracted as

$\{<$ query, $\text{url}_i, \text{url}_j, \text{grade}_i - \text{grade}_j > | i, j = 1, 2, \ldots, n_q, i \neq j\}$.

When combining human-labeled pairs and click preference pairs, we can give use different relative weights for these two data sources. The loss function becomes

$$R(h) = \frac{w}{N_l} \sum_{i \in \textbf{Labeled}} (\max\{0, h(y_i) - h(x_i) + \tau\})^2 \quad \frac{1-w}{N_c} \sum_{i \in \textbf{Click}} (\max\{0, h(y_i) - h(x_i) + \tau\})^2, \quad (2)$$

where w is used to control the relative weights between labeled training data and click data, N_l is the number of training data pairs, and N_c is the number of click pairs. The margin τ can be determined as grade difference for editor pairs, and be a constant parameter for click pairs.

Step 2 is critical for the efficacy of the approach. A few factors need to be considered:

1) data distribution: for the application of task-specific ranking, our purpose is to improve ranking for the queries belonging to this category. An important observation is that the relevance patterns for the ranking within a specific category may have some unique characteristics, which are different from generic relevance ranking. Thus, it is reasonable to consider only using dedicated labeled training data and dedicated click preference data for training. The reality is that dedicated training data is usually insufficient, while it is possible that non-dedicated data can also help the learning.

2) click pair quality: it is inevitable there exist some incorrect pairs in the click preference pairs. Such incorrect pairs may mislead the learning. So overall, can the click preference pairs still help the learning for task-specific ranking? By our study, skip-above pairs usually contain more incorrect pairs compared with skip-above pairs. Does this mean skip-next pairs are always more helpful in improving learning than skip-above pairs?

3) click pair utility: use labeled training data as baseline, how much complimentary information can click pairs bring? This is determined by the methodology of click data mining approach.

While it is possible to achieve some learning improvement for task-specific ranking by using click pairs by a plausible method, we attempt to empirically explore the above interweaving factors for deeper understanding, in order to apply the most appropriate strategy to exploit click data on real-world applications of task-specific ranking.

4 Experiments

4.1 Data set

Query category: in the experiments, we use long query ranking as an example of task-specific ranking, because it is commonly known that long query ranking has some unique relevance patterns compared with generic ranking. We define the long queries as the queries containing at least three tokens. The techniques and analysis proposed in this paper can be applied to other ranking tasks, such as rankings for specific query segments like time-sensitive queries, navigational queries, or rankings for specific domains/contents like answers, blogs, news, as long as the tasks have their own characteristics of data distributions and discriminant rank features.

Labeled training data: we do experiments based on a data set for a commercial search engine, for which there are 16,797 query-url pairs (with 1,123 different queries) that have been labeled by editors. The proportion of long queries is about 35% of all queries. The data distribution of such long queries may be different from general data distribution, as it will be validated in the experiments below.

The human labeled data is randomly split into two sets: training set (8,831 query-url pairs, 589 queries), and testing set (7,966 query-url pairs, 534 queries). The training set will be combined with click preference pairs for rank function learning, and the testing set will be used to evaluate the efficacy of the ranking function. In the training set, there are 3,842 long query-url pairs (229 queries). At testing stage, the learned rank functions are applied only to the long queries in the testing data, as our concern in this paper is how to improve task-specific ranking, i.e., long query ranking in the experiment. In the testing data, there are 3,210 query-url pairs (193 queries) are long query data, which will be used to test rank functions.

Click preference pairs: using the two approaches of heuristic rule approach and sequential supervised approach, we extract click prefence pairs from the click log of the search engine. Each approach yields both skip-next and skip-above pairs, which are sorted by confidence descending order respectively.

Table 5: Use click data by heuristic rule approach (Data Selection: "N": not use; "D": use dedicated data; "G": use generic data. Data Source: "T": training data; "C": click data)

(a) skip-next pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.7822	0.7906 (1.2%)	0.7997(2.4%)
GC	0.7834	0.7908 (1.2%)	0.7950 (1.7%)

(b) skip-above pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.6649	0.7676 (-1.6%)	0.7748 (-0.8%)
GC	0.6792	0.7656 (-2.0%)	0.7989 (2.2%)

Table 6: Use click data by SSL approach (Data Selection: "N": not use; "D": use dedicated data; "G": use generic data. Data Source: "T": training data; "C": click data)

(a) skip-next pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.7752	0.7933 (1.5%)	0.7936 (1.5%)
GC	0.7624	0.7844 (0.4%)	0.7914 (1.2%)

(b) skip-above pairs

	NT	DT	GT
NC	n/a	0.7736	0.7813
DC	0.6756	0.7636 (-2.2%)	0.7784 (-0.3%)
GC	0.6860	0.7717 (-1.2%)	0.7774 (-0.5%)

4.2 Setup and measurements

We try different sample selection and combination strategies to train rank functions using GBrank algorithm. For the labeled training data, we either use generic data or dedicated data. For the click preference pairs, we also try these two options. Furthermore, as more click preference pairs may bring more useful information to help the learning while on the other hand, the more incorrect pairs may be given so that they mislead the learning, we try different amounts of these prefence pairs: 5,000, 10,000, 30,000, 50,000, 70,000 and 100,000 pairs.

We use NDCG to evaluate ranking model, which is defined as

$$\text{NDCG}_n = Z_n \sum_{i=1}^{n} \frac{2^{r(i)}-1}{\log(i+1)}$$

where i is the position in the document list, $r(i)$ is the score of Document i, and Z_n is a normalization factor, which is used to make the NDCG of ideal list be 1.

4.3 Results

Table 5 and 6 show the NDCG_5 results by using heuristic rule approach and SSL approach respectively. We do not present NDCG_1 results due to space limitation, but NDCG_1 results have the similar trends as NDCG_5.

Baseline by training data: there are two baseline functions by using training data sets 1) use dedicated training data (DT), NDCG_5 on the testing set by the rank function is 0.7736; 2) use generic training data (GT), NDCG_5 is 0.7813. It is reasonable that using generic training data is better than only using dedicated training data, because the distributions of non-dedicated data and dedicated data share some similarity. As the dedicated training data is insufficient, the adoption of the extra non-dedicated data helps the learning. We compare learning results with Baseline 2) (use generic training data, the slot of NC + GT in the tables), which is the higher baseline.

Baseline by click data: we then study the utilities of click preference pairs by using them alone for training without using labeled training data. In Table 5 and 6, each of the NDCG_5 results using click preference pairs is the highest NDCG_5 value over the cases of using different amounts of pairs (5000, 10,000, 30,000, 50,000, 70,000 and 100,000 pairs). The results regarding the pairs amounts are illustrated in Figure 3, which will help us to analyze the results more deeply.

If we only use click preference pairs for training (the two table slots DC+NT and GC+NT, corresponding to using dedicated click preference pairs and generic click pairs respectively), the best case is using skip-next pairs extracted by heuristic rule approach (Table 5 (a)). It is not surprising that skip-next pairs outperform skip-above pairs because there are significantly lower percentage of incorrect pairs in skip-next pairs compared with skip-above pairs. It is a little bit surprising that the case of DC+NT has no dominant advantage over GC+NT as we expected. For example, in Table 5 (a), the NDCG_5 values (0.7822 and 0.7834) are very close to each other. However, in Figure 3, we find that with the same amount of pairs, when we use 30,000 or fewer pairs, using dedi-

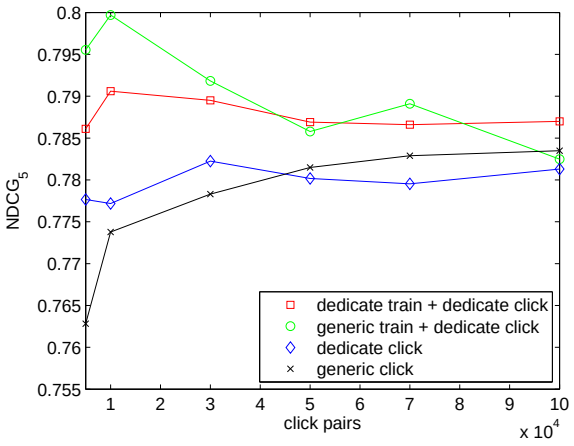

Figure 3: Incorporate different amounts of skip-next pairs by heuristic rule approach with generic training data.

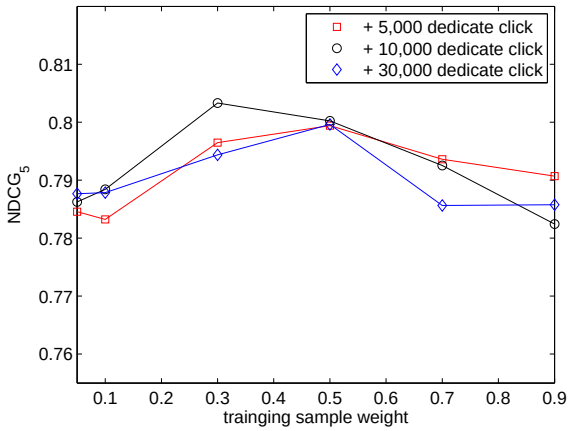

Figure 4: The effects of using different combining weights. Skip-next pairs by heuristic rule approach are combined with generic training data.

cated click pairs alone is always better than using generic click pairs alone. With more click pairs being used ($> 30,000$), the noise rates become higher in the pairs, which makes the distribution factor less important.

Combine training data and click data: we compare the four table slots, DC+DT, GC+DT, DC+GT, GC+GT, in Table 5 and 6, and there are quite a few interesting observations:

1) *Skip-next vs. skip-above*: overall, incorporating skip-next pairs with training data is better than incorporating skip-above pairs, due to the reason that there are more incorrect pairs in skip-above pairs, which may mislead the learning. The only exception is the slot GC+GT in Table 5 (b), whose NDCG_5 improvement is as high as 2.2%. We further track this result, and find that this is the case by using only 5,000 generic skip-above pairs. The noise rate of these 5,000 pairs is low because they have the highest pair extraction confidence values. At the same time, these 5,000 pairs may provide good complementary signals to the generic training data, so that the learning result is good. However, in general, skip-next pairs have better utilities than skip-above pairs.

2) *Dedicated training data vs. generic training data*: using generic training data is generally better than only using dedicated training data. If training data is insufficient, the extra non-dedicated data provides useful information for relevance pattern learning, and the distribution dissimilarity between dedicated data and non-dedicated data is not the most important factor.

3) *Dedicated click data vs. generic click data*: using dedicated click data is more effective than using generic click data. From Figure 3, we observe that when 30,000 or fewer pairs are incorporated into training data, using dedicate click pairs is always better than using generic click pairs.

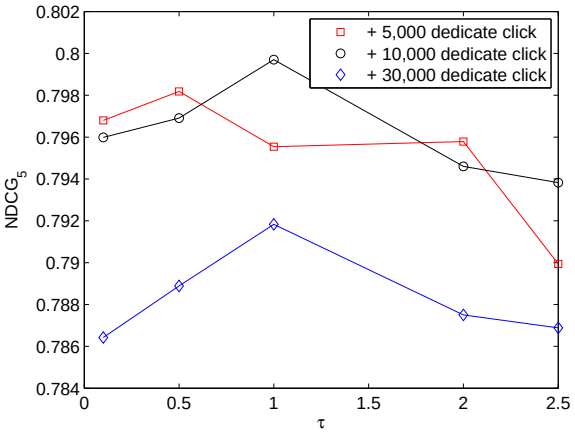

Figure 5: The effects of using different margin values for click preference pairs. Skip-next pairs by heuristic rule approach are incorporated with generic training data.

4) *Heuristic rule approach vs. SSL approach*: the preference pairs extracted by heuristic rule approach have better utilities than those extracted by SSL approach.

5) *GBrank parameters for combining training data and click pairs*: the relative weight w for combining training data and click pairs in (2) may also affect rank function learning. Figure 4 shows the effects of using different combining weights,

for which skip-next pairs by heuristic rule approach are combined with generic training data. We observe that neither over-weighting training data or over-weighting click pairs yields good results while the two data sources are best exploited at certain weight values when there is good balance between them. Another concern is the appropriate margin value τ for the click pairs in (2). Figure 5 shows that $\tau = 1$ consistently yields good learning results, which suggests us that click pair provides good information at $\tau = 1$.

4.4 Discussions

we have defactorized the related approaches for exploiting click data to improve task-specific rank learning. The utility of click preference pairs depends on the following factors:

1) Data distribution: if click pairs have good quality, we should use dedicated click pairs instead of generic click pairs, so that the samples for training have similar distribution to the task of task-specific ranking.

2) The amount of dedicated training data: the more dedicated training data, the more reliable the task-specific rank function is; thus, the less room for learning improvement using click data. For the case in the experiment that dedicated training is insufficient, the non-dedicated training data can also help the learning as non-dedicated training data share relevance pattern similarity with the dedicated data distribution.

3) The quality of click pairs: if we can extract large amount of high-quality click pairs, the learning improvement will be significant. For example, as shown in Figure 3, at the early stage with fewer click pairs (5,000 and 10,000 pairs) being combined with training data, the learning improvement is best. With more click pairs are used, the noise rate in the click pairs becomes higher so that the learning misleading factor is more important than information complementary factor. Thus, it is important to improve the reliability of the click pairs.

4) The utility of click pairs: by our study, the quality of click pairs extracted by SSL approach is comparable to those extracted by heuristic rule approach. The possible reason that heuristic-rule-based click pairs can bring more benefit is that these pairs provide more complementary information compared with SSL approach. As the methodologies of these two click data extraction approaches are totally different, in future we will explore the concrete reason that causes such utility difference.

5 Conclusions

By empirically exploring the related factors in utilizing click-through data to improve dedicated model learning for task-specific ranking, we have better understood the principles of using click preference pairs appropriately, which is important for the real-world applications in commercial search engines as using click data can significantly save human labeling costs and makes rank function learning more efficient. In the case that dedicated training data is limited, while non-dedicated training data is helpful, using dedicated skip-next pairs is the most effective way to further improve the learning. Heuristic rule approach provides more useful click pairs compared with sequential supervised learning approach. The quality of click pairs is critical for the efficacy of the approach. Therefore, an interesting topic is how to further reduce the inconsistency between skip-above pairs and human labeling so that such data may also be useful for task-specific ranking.

References

E. Agichtein, E. Brill, and S. Dumais. 2006. Improving web search ranking by incorporating user behavior information. *Proc. of ACM SIGIR Conference*.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, and O. Frieder. 2007. Varying approaches to topical web query classification. *Proceedings of ACM SIGIR conference*.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. Learning to rank using gradient descent. *Proc. of Intl. Conf. on Machine Learning*.

Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. 2007. Learning to rank: From pairwise approach to listwise. *Proceedings of ICML conference*.

B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. 2008. Here or there: preference judgments for relevance. *Proc. of ECIR*.

O. Chapelle and Y. Zhang. 2009. A dynamic bayesian network click model for web search ranking. *Proceedings of the 18th International World Wide Web Conference*.

K. Chen, Y. Zhang, Z. Zheng, H. Zha, and G. Sun. 2008. Adapting ranking functions to user preference. *ICDE Workshops*, pages 580–587.

M. Coyle and B. Smyth. 2007. Supporting intelligent web search. *ACM Transaction Internet Tech.*, 7(4).

T. G. Dietterich. 2002. Machine learning for sequential data: a review. *Lecture Notes in Computer Science*, (2396):15–30.

S. Fox, K. Karnawat, M. Mydland, S. Dumias, and T. White. 2005. Evaluating implicit measures to improve web search. *ACM Trans. on Information Systems*, 23(2):147–168.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. *Proceedings of International Conference on Machine Learning*.

J. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Ann. Statist.*, 29:1189–1232.

X. Geng, T. Liu, T. Qin, A. Arnold, H. Li, and H. Shum. 2008. Query dependent ranking with k nearest neighbor. *Proceedings of ACM SIGIR Conference*.

N. S. Glance. 2001. Community search assistant. *Intelligent User Interfaces*, pages 91–96.

S. Ji, K. Zhou, C. Liao, Z. Zheng, G. Xue, O. Chapelle, G. Sun, and H. Zha. 2009. Global ranking by exploiting user clicks. In *SIGIR'09*, Boston, USA, July 19-23.

T. Joachims, L. Granka, B. Pan, and G Gay. 2005. Accurately interpreting clickthough data as implicit feedback. *Proc. of ACM SIGIR Conference*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

I. Kang and G. Kim. 2003. Query type classification for web document retrieval. *Proceedings of ACM SIGIR Conference*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

U. Lee, Z. Liu, and J. Cho. 2005. Automatic identification of user goals in web search. *Proceedings of International Conference on World Wide Web*.

X. Li, Y.Y Wang, and A. Acero. 2008. Learning query intent from regularized click graphs. *Proceedings of ACM SIGIR Conference*.

T. Y Liu. 2008. Learning to rank for information retrieval. *SIGIR tutorial*.

T. Qin, T. Liu, X. Zhang, D. Wang, and H. Li. 2008. Global ranking using continuous conditional random fields. In *NIPS*.

H. Li R. Jin, H. Valizadegan. 2008. Ranking refinement and its application to information retrieval. *Proceedings of International Conference on World Wide Web*.

F. Radlinski and T. Joachims. 2007. Active exploration for learning rankings from clickthrough data. *Proc. of ACM SIGKDD Conference*.

F. Radlinski, M. Kurup, and T. Joachims. 2008. How does clickthrough data reflect retrieval quality? *Proceedings of ACM CIKM Conference*.

D. E. Rose and D. Levinson. 2004. Understanding user goals in web search. *Proceedings of International Conference on World Wide Web*.

X. Wang and C. Zhai. 2007. Learn from web search logs to organize search results. In *Proceedings of the 30th ACM SIGIR*.

H. Zha, Z. Zheng, H. Fu, and G. Sun. 2006. Incorporating query difference for learning retrieval functions in world wide web search. *Proceedings of the 15th ACM Conference on Information and Knowledge Management*.

Z. Zheng, H. Zhang, T. Zhang, O. Chapelle, K. Chen, and G. Sun. 2007. A general boosting method and its application to learning ranking functions for web search. *NIPS*.