Improving Xbox Search Relevance by Click Likelihood Labeling

Jingjing Li, Xugang Ye, and Danfeng Li

Microsoft, Bellevue, WA, USA
{jingjing.li,xugangye,lida}@microsoft.com

Abstract. From the original game console, the Xbox has rapidly evolved into a comprehensive entertainment platform where tens of millions of users could not only play video games but also watch movies and TVs, listen music and enjoy Apps. Therefore, building a cross media ranker to provide relevant and personalized search results for Xbox users has become an interesting and imperative task. In this paper, we present our recent progress on improving Xbox's cross media ranker by mining massive click log data and generating multi-class relevance labels. Our experimental results have shown that incorporating the click likelihoods into the label generation yields better click-performance and meanwhile maintains comparable NDCG values, as compared to solely using the human labels generated by a small number of human judges.

Keywords: Click Likelihood, Click Log, Xbox, Search, Ranking, Relevance Labeling.

1 Introduction

Relevance label, which represents how much a user thinks the returned document is relevant to his/her issued query, is critical to the performance of trained ranker. In a ranking model, the relevance label serves as the target for ranking function to fit. Thus, if the relevance label is prone to error, it is hard for the ranking function to learn the best features and parameters to meet user's relevance expectation.

Usually, the labeling tasks are conducted by a small number of human judges due to high recruiting expenses. Therefore, the labeling process is often time-consuming and laborious-taking [1, 2]. Moreover, because the labeling is conducted by a small number of judges, the labeling judgments may not well represent the large user population [3, 4]. Last but not the least, for search problem that is highly time-sensitive, the extent of relevance between a query and a document could vary significantly overtime. Unless we have large amount of human judges to work diligently enough, it is very hard for the labels to keep up with the relevance changing rate.

An alternative method to generate relevance labels is by mining the relevance signals from click log. By utilizing massive amount of click data, the training data is easy to scale in a much faster fashion. Moreover, the labels are less likely to be biased because the relevance between a query and a document is determined by logging the behaviors of a large amount of real users. Finally, we can easily extract relevance

F.F.-H. Nah (Ed.): HCIB/HCII 2014, LNCS 8527, pp. 735–743, 2014.

labels from click log from time to time so as to accommodate any temporal relevance changes.

In this study, following the design science paradigm proposed by Hevner [5], we propose a method to train a ranking model by mining the click likelihood from large amount of click log. Specifically, we will illustrate how to generate relevance labels by leveraging click likelihoods, how to reduce noises and biases in click likelihoods, and how to embed the click likelihoods into a time-sensitive training framework to accommodate relevance changes. As a proof-of-concept, we demonstrate the applicability of our method by applying it to the Xbox's Cross Media search. Under the same set of features and ranking algorithm, we compare and critically discuss the ranking performance between traditional human labeling models and our proposed models. Finally, we summarize the results and discuss the future research direction.

2 Related Work

2.1 Learning to Rank

Learning to rank aims to automatically construct a ranking function (i.e. ranker) $y = f(x)$ from training data, which is usually a supervised or semi-supervised machine learning problem [6-9]. The training data consists of training example at query-document $\langle q, d \rangle$ level. For each $\langle q, d \rangle$, we extract a feature vector $\vec{x}^{q,d}$ as the input for ranking function, and a relevance label $s(q, d)$. The parameters of ranking function is learned by minimizing the error function of the ranker score and the relevance label: $\text{Error}\big(s(q, d), f(\vec{x}^{q,d})\big)$. Therefore, the relevance label $s(q, d)$ is critical for inferring the correct parameters of ranking function.

2.2 Xbox Cross Media Search

In this section, we will introduce the search problem in Xbox and how it is different from the traditional search.

The search content served at Xbox are media-specific, including movie, TV, music and game. The user interface for Xbox is dramatically different from traditional web search. Figure 1 shows the layout of four contents on the first page on Xbox One. This layout indicates that the relevance of first returned result is more important than that of traditional web search as the area of the first returned doc is much bigger.

Moreover, there are considerable noises in user clicks. We found that the click probability fluctuates dramatically as users continue scrolling to the next page. Specifically, we notice an unusual phenomenon that the click probability at the position greater than 125 on average is greater than 0.8, which means users have more than 80% of the chance to click on a document after the 31st page. One reason behind might be there are some malicious users in the system, who intentionally click on document with poor relevance. Another possible reason is that scrolling and clicking action using Xbox's controller is relatively easy to achieve than traditional web search, thus users may mistakenly scroll too many pages before initiating their clicks. However, when we only consider the clicks in the top 50 position (see Figure 2), the

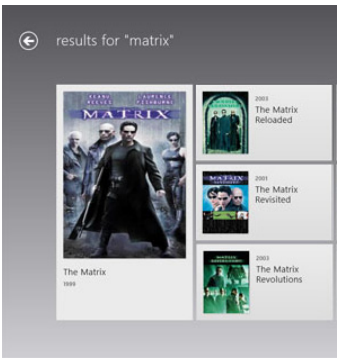

Fig. 1. Content layout on the first page of the Xbox One's search results

click probability is more aligned with the traditional cascade model assumption [10]. Consequently, in order to get more reliable click signals from Xbox, we may need to truncate clicks happened at bottom of the page.

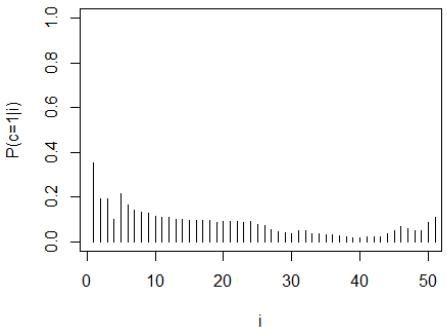

Fig. 2. The click probability at position i

3 Proposed Framework

3.1 Overall Framework

In this section, we will discuss the major steps involved in ranker training with click likelihoods. The first step is to construct training data, which consists of five sub-steps: query sampling, query-document pair selection, click likelihood calculation, smoothing & cleaning and feature extraction. The first two steps focus on generating query-document pair $\langle q, d \rangle$, which involves spamming filtering and tail $\langle q, d \rangle$ pairs removal. Feature extraction aims to find the feature vector $\vec{x}^{q,d}$ for a selected $\langle q, d \rangle$. Click likelihood with its smoothing and cleaning techniques are key components in our article and we will discuss them in 3.2 and 3.3. Finally, we train a ranker with optimal parameters and evaluate its final performance on the test data. The specific machine learning algorithm is called LambdaRank [11].

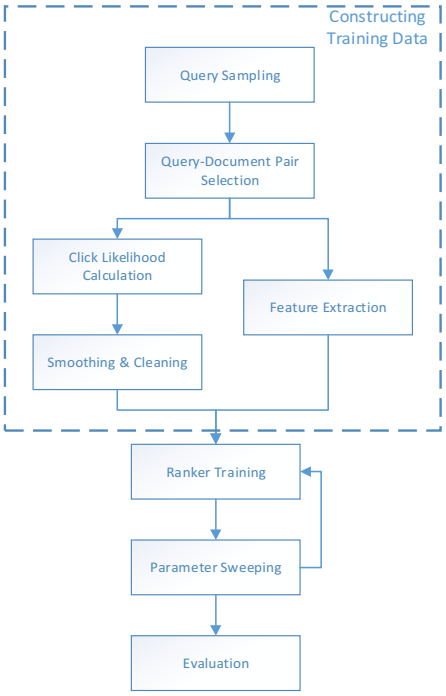

Fig. 3. Overall framework for ranker training with click likelihood

3.2 Click Likelihood and Relevance Label

The ranking function tries to find a list of indexed entities $d_1, \ldots, d_l$ for a query q such that the $s(q, d_i) \geq s(q, d_j)$ if $i < j$, where s is the relevance measure. Our goal therefore is to find a metric that could best approximate $s(q, d_i)$.

The click likelihood for a query-document pair $(q, d)$is denoted as:

$$Click\ Likelihood\ (CL) = p\left(c_{qd} = 1 \mid s_{qd} = 1\right)$$
$$= \frac{\#\ of\ times\ d\ clicked\ under\ q}{\#\ of\ times(q,d)\ shown\ to\ the\ users} = \frac{n_{c_{qd}}}{n_{s_{qd}}} \quad (1)$$

Assuming that a user would only click on the relevant documents, the click likelihood thus could be used as the relevance measures.

Nonetheless, we need to consider the fact that the click logs are often very noisy. For example, spamming users could generate abnormal click statistics. We implemented several techniques to reduce the noise. We will discuss some representative methods in the following section:

(1) **Spam detection:** in the query sampling stage, we allow a user to only contribute one search for a given query in one day. By using this technique, we eliminate the affect posed by those malicious users or robots who issue the same searches numerous times.

(2) **Query document filtering:** the estimate of click likelihood for query-document that only show a limited amount of times is more likely to be biased. For example, if a query-document pair only shows two times in the log, and has been clicked once, the click likelihood is 0.5—much higher than the average. However, with only two views of this query-document pair, this particular click is very likely to happen by chance. To reduce this bias in our dataset, we discard any query-document pairs which show less than 50 times given a specific time frame.

Moreover, searching a query could return dozens or hundreds of documents, but it is unlikely that all of them are examined by users. According to the definition of click likelihood, we need to only count the documents that are seen by the users as the denominator. Therefore, we applied two filtering techniques:

a. $n_{s_{qd}} = \#\ of\ documents\ whose\ position >= last\ click\ position$

b. $n_{s_{qd}} = \#\ of\ documents\ whose\ position >= a\ fixed\ position\ (usually\ less\ than\ 50)$

According to our observation in Figure 2, we expect the ranker performance using method b. to be better than that using method a.

(3) **Click likelihood smoothing:** we also applied a smoothing technique to penalize the query-document pair that has low number of view count. The smooth function is denoted as:

$$CL_s = \frac{n_{c_{qd}} + sm * prior(CL)}{n_{s_{qd}} + sm} \tag{2}$$

Where sm is a smoothing factor and $prior(CL)$ is the average click likelihood of the entire data set.

(4) **Outlier removal:** We transformed the final click likelihood into log-odds ratios to identify outliers. Since the distribution of log-odds ratios generally follow normal distribution, we can manual check the distribution graph to inspect possible outliers and remove long tails. The log-odds ratio function is denoted as:

$$logodds(CL_s) = \frac{\ln(CL_s)}{\ln(1 - CL_s)} \tag{3}$$

Finally, we need to convert the cleaned click likelihood into discretized relevance label gain. Since the maximum label gain for human labeling setting is 15, and the final click likelihood ranges in [0,1], we simply multiple of the click likelihood by 15 and rounded to integers and generate a 15-level relevance labels.

3.3 Constructing Training Data for Time-Sensitive Features and Relevance Labels

Because many features in the ranking model is highly time-sensitive, we need to consider time effect when constructing the training data. Suppose the original data is collected in a time horizon $[t_0, t_1]$. We cut this time zone into the so-called "feature zone" and "target zone", which are respectively $[t_0, \tau)$ and $[\tau, t_1]$ with $t_1 - \tau < \tau - t_0$. For each (q, d)-pair that is both available in $[t_0, \tau)$ and $[\tau, t_1]$, the feature vector $\vec{x}^{q,d}$ is generated within the time frame $[t_0, \tau)$, while the relevance label $click\ likelihood$ is estimated using the information in the time frame $[\tau, t_1]$.

A more advanced method to prepare training data that could accommodate relevance changes overtime is to use a sliding window. We generate multiple training dataset corresponding to $[t_0, \tau_i)$ and $[\tau_i, t_1]$ where $\tau_i \in (t_0, t_1)$ and combine them together. Therefore, for a single query-document pair, we can investigate how the feature vector and relevance label changes over time.

4 Demonstration and Evaluation

4.1 Evaluation Metrics

We use two types of metrics to evaluate the ranker performance: NDCG (normalized discounted cumulative gain) **Error! Reference source not found.** and future click-performance. NDCG is a traditional metric to evaluate ranking performance. It allows each document to have a graded relevance (e.g., bad, fair, good, excellent, perfect) while some other traditional measures (precision, recall, …, etc.) only allows binary relevance. It also assigns higher weight to the document at the top of the result list. We use the NDCG of the top i positions for $i = 1,2,4$ as our NDCG measurements. Specifically, the NDCG of the top i positions is calculated as

$$\mathrm{NDCG}_i = \frac{1}{N}\sum_q \left(\sum_{j=1}^{i} \frac{rel_j^q}{\log_2(1+j)}\right) / \left(\sum_{j=1}^{i} \frac{\overline{rel}_j^q}{\log_2(1+j)}\right) \tag{4}$$

where $\overline{rel}_1^q \geq \cdots \geq \overline{rel}_i^q$ represent the descending order of $rel_1^q, \ldots, rel_i^q$, which respectively are the relevance gains of the first i documents under the query q that does not have all zero-gain results, and N is the total number of such queries.

Future click-performance represents the user engagement to the search service after a ranker is deployed to production. We use the click-happening-rate (CHR) and the last-click-rate (LCR) of the top i positions for $i = 1,2,4$ as our click-metrics. The CHR of the top i positions is calculated as

$$\mathrm{CHR}_i = \frac{\text{Number of sessions having} \geq 1 \text{ click in first } i \text{ positions}}{\text{Number of sessions}}. \tag{5}$$

The LCR of the top i positions is calculated as

$$\mathrm{LCR}_i = \frac{\text{Number of sessions having the last click in first } i \text{ positions}}{\text{Number of sessions}}. \tag{6}$$

4.2 Data Set

We collected the query-document pairs that satisfying the selection criteria from the click log in Feb 1st ~ June 1st, 2013. Notice that only 17% of the selected query-document pairs have human labels. Thus the training data set for human label ranker is from these 17% of query-document pairs. The training data set for click-based ranker include all the query-document pairs. The click-logs in June 2013 are used to test the future click-performance. The human-judged query-document pairs generated in movie and TV domains from June 2013 to October 2013 were used to test the future NDCG-performance.

4.3 Experimental Results

In this section, we present some experimental results of comparing the relevance labels generated by our method with several other types of relevance labels. For each relevance label setting, we kept the same feature set and used the same ranker-learning algorithm LambdaRank. Following are the description of each experiment:

i. HJ: Human judgment. The labels have three unique values of gain: 15 for Excellent, 7 for Good, and 0 for Bad. The query-document pairs in the training data contain only those with HJ labels.
ii. lCTR: CTR under the assumption that in a session, the returned results before the last click are viewed and those after the last click are not viewed. The label gain is calculated as $\mathrm{lCTR} \cdot 15$ and rounded. The query-document pairs in the training data contain only those with lCTR labels.
iii. tCTR: CTR under the assumption that in a session, the returned results before a pre-defined truncated position are viewed and those after the truncated position are not viewed. The label gain is calculated as $\mathrm{tCTR} \cdot 15$ and rounded. The query-document pairs in the training data contain only those with tCTR labels.
iv. tCTR∪HJ: Combination of tCTR-labels and HJ-labels. The query-document pairs in the training data are the union of those with tCTR-labels and those with HJ-labels. For each with both the tCTR- label and the HJ-label, the final label gain is calculated as the rounded average.

Once the relevance labels were determined in the target zone, we performed the feature extraction for each query-document pair in the feature zone. Four rankers were trained from these training data, and were used to score the query-document relevance on the same test data. The test results are summarized in Table 1 and 2.

Table 1. The test click-performance

Label	CHR_1	CHR_2	CHR_4	LCR_1	LCR_2	LCR_4
HJ	55.77%	70.65%	85.40%	49.00%	64.47%	81.20%
lCTR	53.11%	69.46%	84.73%	46.34%	63.18%	80.65%
tCTR	**57.39%**	**71.39%**	85.69%	**50.18%**	65.22%	81.68%
tCTR∪HJ	57.13%	71.27%	**85.78%**	50.17%	**65.25%**	**81.71%**

Table 2. The NDCG values from one human judgment data

Label	Movie			TV		
	$NDCG_1$	$NDCG_2$	$NDCG_4$	$NDCG_1$	$NDCG_2$	$NDCG_4$
HJ	**91.71%**	**90.22%**	**89.93%**	**94.35%**	**93.63%**	**93.95%**
lCTR	83.92%	81.77%	81.83%	91.81%	90.44%	91.17%
tCTR	86.46%	83.70%	84.30%	90.80%	91.03%	92.26%
tCTR∪HJ	90.56%	88.09%	88.41%	92.78%	92.73%	93.25%

From Table 1, we found that lCTR has the worst performance, indicating the existence of noises in click logs. However, after only utilizing the click likelihood in the top 50 documents per query, the performance is dramatically increased. This means the clickes on the first few pages are more trustworthy. After combined with the human label, tCTR∪HJ model also has satisfactory performance on future click metrics.

Per Table 2, we found that ranker trained purely from human labels has the best results. This is expected because the test data for NDCG evaluation uses the human label as the relevance labels. Similar to the pattern in click metrics, the lCTR model has the worst performance, and the gap to the optimal performance is even larger. However, after removing the query-document pairs happened after the 50^{th} position, tCTR has gained considerable performance. The best click-based model is the one that combines the human label and click likelihood, which is almost comparable to the pure human-label ranker (p value greater than 0.05).

The initial experimental results has shown that the ranker trained on click labels yielded comparable performance to the ranker trained on human judgment labels. This result is encouraging because we can quickly increase the training data size with limited manual efforts. We also found that, compared to the model with human label, the model with click labels performs especially better on head queries. Hence in the future we can adopt a hybrid model which utilizes click to generate labels for head queries and employs human judges to label tail queries. Last but not the least, we found the click- based rankers are better in predicting the future customer engagement.

However, we also find there are considerable noises contained in click likelihood. For example, eye-tracking experiments have found that the probability of a document being clicked under a query is not only determined by relevance but also the document position and presentation format. Therefore, we are currently working on a generative model that could accommodate position bias.

5 Conclusion

In this study, we propose an alternative method to generate relevance labels for Xbox's cross media ranker training. We find that the traditional human labeling method is time-consuming, not representative and not responsive to market dynamics. Therefore, we propose a method to use click likelihood as relevance labels. Since click logs often contain noises and is biased towards position and presentation, we discussed several techniques to reduce these noises. Finally, using the same set of features and ranking algorithm, we compared our ranker with traditional human-labeled ranker. We found that the click-based ranker is more suitable to predict future user engagement and human-label-based ranker performs better in traditional NDCG metric. The overall best

performing model is the hybrid model which combines human label and click likelihood. Therefore, in the future we could leverage click log to get training data at low cost and in a more timely fashion, and let human judges could concentrate on the relevance labeling for tail query-document pairs that are more likely to contain click noises.

One limitation of our proposed method is that we did not differentiate the click likelihood between queries. From the Xbox click log, we found that some queries are title-specific (such as "skyfall"), while some are people-specific (such as "jennifer lawrence"). These two types of queries in general give different click patterns: the title-specific queries has more concentrate click distribution while the people-specific queries has more spread distribution. Therefore, one future research direction is to create query-specific click likelihood.

Another limitation is that we reduce the position bias by relatively simple heuristics—only disregarding document clicks after a fixed position (less than 50). A more sophisticated method is to look into the relationship between click sequence, viewing position and relevance levels at the same time. Therefore, we are currently working on a generate model which could account for the aforementioned factors in a comprehensive way.

References

1. Voorhees, E.M.: Variations in relevance judgments and the measurement of retrieval effectiveness. Information Processing & Management 36, 697–716 (2000)
2. Bailey, P., Craswell, N., Soboroff, I., Thomas, P., de Vries, A.P., Yilmaz, E.: Relevance assessment: are judges exchangeable and does it matter. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 667–674. ACM (2008)
3. Agrawal, R., Halverson, A., Kenthapadi, K., Mishra, N., Tsaparas, P.: Generating labels from clicks. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, pp. 172–181. ACM (2009)
4. Xu, J., Chen, C., Xu, G., Li, H., Abib, E.R.T.: Improving quality of training data for ||learning to rank using click-through data. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 171–180. ACM (2010)
5. von Alan, R.H., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly 28, 75–105 (2004)
6. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 89–96. ACM (2005)
7. Burges, C.J., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: NIPS, pp. 193-200 (2006)
8. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning, pp. 129–136. ACM, New York (2007)
9. Chapelle, O., Chang, Y.: Yahoo! Learning to Rank Challenge Overview. Journal of Machine Learning Research-Proceedings Track 14, 1–24 (2011)
10. Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: Proceedings of the 2008 International Conference on Web Search and Data Mining, pp. 87–94. ACM (2008)
11. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning 11, 23–581 (2010)