Smoothing Click Counts for Aggregated Vertical Search

Jangwon Seo[1], W. Bruce Croft[1], Kwang Hyun Kim[2], and Joon Ho Lee[2]

[1] CIIR, University of Massachusetts, Amherst, MA, 01003, USA
[2] NHN Corporation, 463-824, Korea
{jangwon, croft}@cs.umass.edu {khkim, joonho}@nhn.com

Abstract. Clickthrough data is a critical feature for improving web search ranking. Recently, many search portals have provided *aggregated search*, which retrieves relevant information from various heterogeneous collections called *verticals*. In addition to the well-known problem of rank bias, clickthrough data recorded in the aggregated search environment suffers from severe sparseness problems due to the limited number of results presented for each vertical. This skew in clickthrough data, which we call *rank cut*, makes optimization of vertical searches more difficult. In this work, we focus on mitigating the negative effect of rank cut for aggregated vertical searches. We introduce a technique for smoothing click counts based on spectral graph analysis. Using real clickthrough data from a vertical recorded in an aggregated search environment, we show empirically that clickthrough data smoothed by this technique is effective for improving the vertical search.

1 Introduction

Clickthrough data is invaluable information that records user actions in response to search results. Recently, there have been a number of efforts using clickthrough data to improve ranking. For example, by analyzing clickthrough data, we can discover the users' preferences for certain search results. These preferences can be used to evaluate search systems [14] or to be incorporated for ranking functions [2]. These approaches have proved to be effective for optimizing web search ranking based on a single web repository.

Many search portals, however, now provide *aggregated* rankings based on various domain-specific collections, e.g., news, blogs, community-based question answering (CQA), images, etc. These domain-specific collections are often called *verticals*. There is a separate index for each vertical and a search engine optimized for the vertical returns its own results. Aggregation logic in the search portal selects relevant verticals and displays results returned from each vertical in a result page. This is referred to as *aggregated search*.

Clickthrough data of each vertical recorded in aggregated search has some different properties than the data recorded in a typical web search. A result page is a limited resource that the verticals share. In many cases, even if a vertical is relevant, the vertical cannot have more than a few results in an aggregated result

page. Consequently, the clickthrough data can suffer from significant distortion. For example, Figure 1 shows the click count distribution of a vertical where only the top 5 results are delivered to aggregated search. We refer to the number of results returned to aggregated search as the *cut-off rank*. Clearly, there is a steep attenuation of click counts after the cut-off rank. We refer to the attenuation caused by the cut-off rank as *rank cut*.

Previous studies on web search have reported that clickthough data suffers from biases such as *rank bias* [14]. Generally, since rank bias is caused by users' scanning behavior, click counts affected by rank bias show a smoothly decreasing curve as the rank decreases. While we can observe rank bias in the top ranks in Figure 1, the most dominant phenomenon is rank cut, which is caused by a system parameter (cut-off rank) rather than user behavior. That is, if rank cut exists, users have limited views of results from verticals. Then, while a click on a document may be considered as a signal that the document is likely to be relevant, more clicks on a document do not imply that the document is more relevant. If we want to optimize vertical search using clickthrough data, this phenomenon can be problematic. The problem of rank cut is not limited to any specific type of aggregated search. That is, whether interleaving or grouping results for aggregated search, as long as only a few results from each of many verticals are selected for display in a result page, rank cut may occur.

Of course, there is a similar cut-off caused by displaying results in pages in general web search. In web search, however, a page delivers more results (usually, 10) and users can easily see results beyond the current page by following the link to the next page. On the other hand, in aggregated search, verticals typically return at most five results and often only one or two results are returned. Furthermore, in an aggregated result page, results from other verticals follow each other or are interleaved with each other. Accordingly, when users want to see more results, they naturally see results from other verticals rather than lower ranked documents from the same vertical. Therefore, the effect of rank cut is more noticeable in aggregated search.

Some search portals provide an option where users can issue queries directly to vertical search engines. If a large volume of queries are input through separate vertical search interfaces, we can use this clickthrough data for vertical search optimization without considering aggregated search. However, not all verticals have their own search interfaces and even if they did, sufficient clickthrough data may not be collected because aggregated search is the default search option in most cases.

Therefore, in this paper, we focus on improving vertical search using clickthrough data by mitigating a negative aspect of clickthrough data recorded in aggregated search, i.e. rank cut. To address rank cut, we introduce a technique based on spectral graph analysis (*click count regularization*). Using a real vertical collection and its click log data recorded in aggregated search, we study the effectiveness of this click count smoothing technique for vertical search optimization.

2 Related Work

To the best to our knowledge, there is no work that explicitly addresses skews in click distributions of verticals in aggregated search. On the other hand, there is some prior work on exploiting clickthrough data in vertical search. Li et al. [16] use click graphs to classify query intent for vertical search. Their work is similar to our work in that both use semi-supervised learning techniques based on clickthrough data. However, while they use click graphs to expand a volume of labeled queries for the purpose of training, we regularize click counts on document graphs.

There have been many efforts to optimize general Web search using clickthrough data. Joachims [12] introduces the use of clickthrough data to learn ranking functions. Joachims et al. [14] investigate various biases in clickthrough data by user behavior analysis and propose extracting implicit relevance feedback from the data. Dupret and Piwowarski [9] and Chapelle and Zhang [5] analyze browsing behaviors of users in response to search results using click models that take bias into account.

Recent work by Gao et al. [10] focus on the sparseness problem of clickthrough data. They introduce a random walk algorithm based on click graphs and a discounting technique to expand clickthrough data. Since the rank cut problem can be considered as a kind of sparseness problem, their work is the closest to our work. However, their work differs from ours in that they use non-content-based algorithms and their domain is limited to web search where the sparseness problem is often less serious than aggregated vertical search. In other related work, Radlinkski et al. [19] propose an active exploration method to tackle the sparseness problem caused by rank bias.

Click count regularization can be considered as a label propagation algorithm [21] [24] [4]. Label propagation is a semi-supervised learning algorithm which leverages labeled data to classify unlabeled data. In particular, it assumes that if two instances in a high-density region are close, their corresponding values are similar. In addition, Diaz [8] leverages a graph Laplacian-based regularization technique for re-ranking tasks in Information Retrieval. Similar to our work, he builds document-content-based graphs. However, the objective to be regularized is a document score vector. Moreover, a different cost and an optimization technique are used.

3 Data Set

In this paper, as a data set for experiments, we use a snapshot of the community-based question answering (cQA) vertical[3] of Naver, which is a major commercial search portal in Korea. The snapshot which contains about 30 million documents was directly downloaded from the service databases. Since the CQA service is the most popular service in Naver and the most clicks occur on search results

[3] http://kin.naver.com/

from the CQA vertical in aggregated search, we choose this data set. However, since we want to look at the collection as a vertical in aggregated search, we do not address unique features associated with CQA collections.

Naver provides an aggregated search service. In this service, there are various verticals, e.g., news, CQA, blog, forum, image, video, book, etc. Each vertical search engine retrieves its own relevant items for a user query. The results are grouped according to the corresponding verticals and each group is ranked by a vertical ranking algorithm. Note that there are various ways to aggregate search results from verticals. The results can be interleaved with each other or grouped by each corresponding vertical. Aggregated search provided in Naver uses the latter. Currently, the CQA vertical returns 5 search results to the aggregated search interface if the CQA vertical is determined to be a relevant vertical by a vertical selection algorithm. In addition, although Naver provides a separate search interface for each vertical, aggregated search is the default and most users use aggregated search even when their queries are more suitable for a specific vertical [17].

To establish a test collection for the CQA vertical, we chose 972 of the most frequent queries input to the Naver aggregated search interface, considering the following: i) the queries should be able to address as many different topics as possible, ii) the length distribution of the queries should be close to that of real user queries. For each query, the top 50 documents retrieved by the current Naver CQA vertical search engine are considered as a document set for relevance judgments. We asked 20 editors to judge them on a four point scale: non-relevant, partially relevant, relevant and highly relevant. In total, we made 46,270 relevance judgments. Note that there are few overlaps between the judgment sets of different topics. We use 500 queries as a training set and the remaining 472 queries as a test set.

We use query logs and click logs which were collected for one week - from Aug 26, 2008 to Sep 01, 2008 - through both the aggregated search interface and the CQA vertical search interface in Naver.

An entry of the query logs contains the following information:

$$< q, v_1[(d_{11}, r_{11}), \cdots, (d_{1n}, r_{1n})], v_2[(d_{21}, r_{21}) \cdots], \cdots >$$

where q is a user query, v_i is a vertical ID, d_{ij} is a document ID in v_i, and r_{ij} is a rank of d_{ij}. Therefore, from the query logs, we can know which documents are returned to users.

An entry of the click logs contains the following information:

$$< q, v, d, r >$$

where q is a user query, and v and r are a vertical and a rank of clicked document d, respectively. Therefore, from the click logs, we can know which documents are clicked for a query.

We filtered out the click log entries which do not include any result from the CQA vertical. Then we took click logs whose queries are contained in the query set of the test collection. To compare queries in the logs with queries in

the test collection, we stemmed the queries using a Korean stemmer used for the current Naver search engine. Consequently, we obtained about 3.3 million query log entries corresponding to the test collection. We applied the same process to the click logs and obtained about 1 million click log entries.

Note that only 178 test queries appear in click data collected from the separated CQA vertical search while all test queries appear in click data from the aggregated search. This is because most users input queries to the aggregated search interface. A similar situation can happen in case that a separated vertical search interface is not popular or cannot be provided to users. Then, using only the vertical click data seems inappropriate although the data is relatively free from some artifacts such as rank cut. Accordingly, in this work, we merged click data from the aggregated search interface and the separated CQA vertical search interface.

To observe skewness in click log data, for each rank, we summed all click counts over all queries in the test collection. Figure 1 presents the distribution of click counts according to ranks. As we see, the only top 5 results dominate significantly the entire click counts. This shows that the effect of rank cut can be serious.

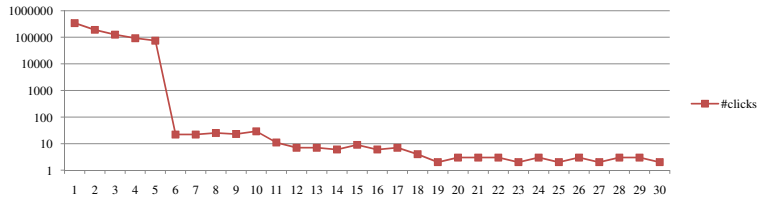

Fig. 1. Distribution of click counts of a vertical recorded via aggregated search and separated vertical search according to ranks. The horizontal axis is the rank and the vertical axis is the click count. The aggregated search interface delivers the top 5 results from the vertical to users.

4 Click Count Regularization

One problem that can be attributed to the rank cut is that beyond a cut-off rank, there can be relevant documents which could have been clicked if they had been delivered to users. We solve this problem by predicting the click counts of such documents.

Our motivation is that if two documents are similar with respect to any measure and one of them is clicked for a query, then the other is likely to be clicked for the same query. This can be seen as an interpretation in terms of click counts of the cluster hypothesis by van Rijsbergen, that states "closely associated documents tend to be relevant to the same requests" [22].

Let us consider a click count itself as a label.[4] A document which has been returned to users can be considered as a labeled instance whereas a document

[4] We do not claim that a click is considered as a relevance label. A label here is a metaphor for explaining a semi-supervised learning setting.

beyond the cut-off rank can be considered as an unlabeled instance. Then, we would like to predict labels (click counts) of the unlabeled instances. This task is similar to semi-supervised learning tasks such as label propagation [21] [24] [4]. In this section, for ease of explanation, we refer to documents which have been returned to users as labeled documents and to the other documents as unlabeled documents. To determine if a document is labeled, we count query log entries which contain the document. If the document has been returned to users more than E times, we consider the document labeled. Note that some labeled documents may not receive any clicks. In the regularization process, these documents presumably contribute to non-relevant documents having small click counts. We set $E = 100$ in this paper because most documents ranked above the cut-off of rank cut have been presented to users more than 100 times in our data set.

In order to determine relationships between unlabeled documents and labeled documents, we specify how to measure similarity between documents. For example, the heat kernel has good properties as a similarity measure [15]:

$$K_{heat}(x_1, x_2) = (4\pi t)^{-n/2} \exp\left(-(4t)^{-1} distance(x_1, x_2)^2\right) \quad (1)$$

where t is a parameter. Since we will use the graph Laplacian which is closely related to the heat equation [3] in this work, the Gaussian form of the heat kernel is a favored choice as a similarity measure. Considering the heat kernel, we define two similarity measures based on different distances: topic similarity and quality similarity.

For topic similarity, we assume that a document D can be represented by a multinomial distribution $\theta = (tf_{w_1}/|D|, tf_{w_2}/|D|, \cdots, tf_{w_n}/|D|)$, where tf_{w_k} is a term frequency of term w_k and n is the size of vocabulary. Then, we can define the geodesic distance on the n-simplex by $2\arccos(\sqrt{\theta_1} \cdot \sqrt{\theta_2})$ on a manifold of multinomial models with the Fisher information metric [15]. Using this distance, the heat kernel becomes the multinomial diffusion kernel that has been proved to be effective in many tasks [15].

For quality similarity, we define a quality distance by $1 - \min(q_1, q_2)$, where q_k is a quality score which has a range [0,1]. The quality score can be any quality estimate such as PageRank [18]. We here use a quality estimate for a CQA vertical which is similar to what has been done by Jeon et al. [11]. The distance implies that if the qualities of both documents are good, the documents are close. That is, click diffusion is assumed to occur only between good quality documents.

We now build an affinity matrix W based on each similarity measure. One of the most straightforward approaches to build an affinity matrix is to compute the similarity between all documents in a collection. However, this is infeasible in case of a large collection. Instead, since we are interested in documents whose ranks are not high enough to be above cut-off rank R, we compute the similarity between the top N documents, where $N >> R$. An N by N affinity matrix is constructed as follows:

$$W_{ij} = \begin{cases} K(x_i, x_j) & i \neq j \ \& \ K(x_i, x_j) > T \\ 0 & \text{otherwise} \end{cases}$$

where T is a threshold to make the affinity matrix sparse. Without the threshold, click counts can be diffused to unrelated documents because similarity values are not often 0, even when documents are not similar.

We may want to combine multiple similarity measures to make an affinity matrix. For example, two documents which are similar in terms of both their topics and quality may reveal a stronger connection than other two documents which are similar in terms of only quality. We combine M affinity matrices computing a geometric mean to preserve sparsity, that is, the combined entry is 0 if any entry in matrices to be combined is 0.

We now define a cost to be minimized by regularization, following the quadratic cost criterion by Joachims [13] and Bengio et al. [4].

$$C(\hat{f}) = ||I_{[l]}(\hat{f} - f)||^2 + \pi \hat{f}^T L \hat{f} + \epsilon ||\hat{f}||^2$$

where $I_{[l]}$ is a diagonal matrix where $I_{[l]_i} = 1$ only if ith column (or row) corresponds to a labeled document and all other entries are 0, f is an original click count vector, $\hat{f}$ is a regularized click vector, L is the un-normalized graph Laplacian ($L = D - W$, where D is the diagonal degree matrix given by $D_i = \sum_j W_{ij}$), and π and ϵ are parameters which are greater than 0.

The first term of the cost is a constraint to minimize the difference between the regularized click counts and the original click counts of the labeled documents. The second term is the Dirichlet sum, which expresses smoothness on the underlying manifold [6]. The smaller the Dirichlet sum is, the smoother f is. This term is crucial because the notion of smoothness implies that two similar documents have similar click counts. The weight of this term can be controlled by the parameter π. The last term controls a size of $\hat{f}$ and gives numerical stability. The weight of this term can be controlled by the parameter ϵ.

Following Bengio et al. [4], we use the Jacobi method [1] to solve this linear system. Then, the approximated solution can be written as follows:

$$\hat{f}_i^{(t+1)} = \frac{1}{I_{[l]_i} + \pi \sum_j W_{ij} + \epsilon} \left(I_{[l]_i} f_i + \pi \sum_{j \neq i} W_{ij} f_j^{(t)} \right)$$

We can guarantee convergence of the Jacobi method as a strictly diagonally dominant matrix is used. In our experiments, most runs quickly converged with fewer than 10 steps.

We refer to this process as "click count regularization". We call $\hat{f}$ after convergence the regularized click count vector. Each value in $\hat{f}$ is referred to as a regularized click count of the corresponding document. Figure 2 presents the effect of click count regularization. As we see, before regularization, the click count curve sharply decreases and only a few documents have clicks. On the other hand, after regularization, we can observe slow attenuation and clicks on more documents.

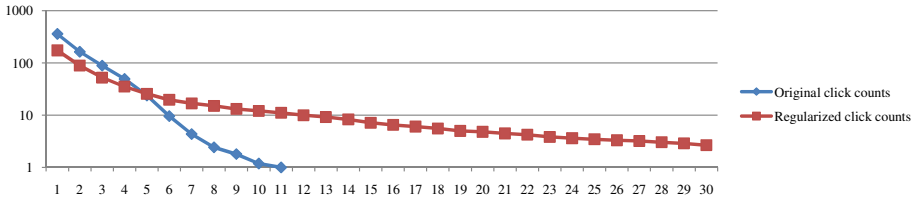

Fig. 2. Change of click counts after click count regularization. Clicks on documents for each query in the testing set are sorted in decreasing order and averaged over all testing queries. The horizontal axis is the sorted order and the vertical axis is the average click counts.

5 Experiments and Discussion

We carried out the following experiments to evaluate effectiveness of the click count regularization technique in vertical search.

5.1 Features and Learning

Two types of features are used for ranking experiments: text-based features and click-based features. Text-based features are derived from a bag-of words language modeling approach for Information Retrieval [7]. Specifically, we use the Dirichlet-smoothed unigram language model [23]. For implementation of the language model, we used the Indri search engine[20]. All documents are stemmed by a Korean stemmer used for the current Naver search engine. We make four language model-based features by applying different normalization factors or taking the logarithm of the query-likelihood score.

Click-based features are derived from the ratio of a click count on a document to the total click count for a given query. Similar to text-based features, we have five different features via manipulation by different normalization factors and the logarithm function.

For learning to rank with these features, we use the rank SVM algorithm [12].

5.2 Baselines and Evaluation

We consider a weak baseline and two strong baselines for evaluation and comparison. The weak baseline is a model which does not use any click count-based feature. That is, this is not different from the unigram language model. The first strong baseline (strong baseline 1) is a model which uses features based on the original click counts. That is, this baseline produces rankings in case that any click count smoothing technique is not employed. The second strong baseline (strong baseline 2) uses features based on click counts smoothed by a state-of-the-art smoothing technique [10] instead of the original click counts. This technique leverages a random walk algorithm on click graphs and a discounting technique inspired by the Good-Turing estimator. Gao et al. [10] have

demonstrated that this technique is effective for addressing sparseness of clickthrough data for Web search. Note that in contrast to our proposed technique, this technique uses click graphs extracted from click logs without considering document contents.

We evaluate the proposed click count regularization technique using the test collection described in Section 3. For each query, we initially rank 50 documents using the unigram language model on the test collection and re-rank the documents by a learned ranking model incorporating click count-based features. To build affinity matrices, we consider 'topic + quality' combination as well as topic similarity. We use four evaluation metrics: normalized discounted cumulative gain at 5 and at 10 (NDCG@5 and NDCG@10), and mean average precision (MAP). This scores are computed using relevance judgments on the CQA vertical described in Section 3.

In order to tune models on the training queries, we find parameter values which maximize NDCG at 5. A Dirichlet smoothing parameter μ was set to 2000 since the value has showed the best performance for the weak baseline. For click count regularization, parameters to be tuned are a similarity threshold T ($\in [0.1, 0.9]$), matrix combination parameters α's ($\in [0.1, 0.9]$), and two regularization parameters, π and ϵ ($\in [0.1, 1]$). To test the statistical significance of an improvement, we perform a paired randomization test with p-value < 0.05.

5.3 Results

Table 1 shows the experimental results. Not surprisingly, even without using the click count smoothing technique, the model incorporating click count-based features (strong baseline 1) outperforms the model incorporating only unigram language model-based features (weak baseline). This is in part because the Naver CQA vertical search engine used for collecting clickthrough data outperforms the weak baseline. Since click count-based features are extracted from clicks on documents highly ranked by the vertical search engine, the performance of the strong baseline can be assumed to be similar to or better than that of the real vertical search engine. Consequently, all models incorporating click count-based features significantly outperform the weak baseline.

Interestingly, the model using click counts smoothed by the random walk and discounting technique (strong baseline 2) fails to show any significant difference from the model using the original click counts (string baseline 1). The random walk/discounting technique has been proved to be effective for sparseness problems in Web search clickth logs, but click logs for verticals in aggregated search are often much sparser than click logs from Web search because of the effect of rank cut. Therefore, relying only on sparse log information without considering contents of documents may not be enough for addressing highly sparse click logs.

Click count regularization shows consistent improvements over both strong baselines. Moreover, all improvements are statistically significant. When using only the topic similarity-based affinity matrix, regularization shows statistically significant improvements on the strong baseline for all evaluation metrics. When using both of topic similarity and quality similarity, the results show the best

performance for most evaluation metrics although the improvements on the results only by topic similarity are somewhat marginal. This shows that given that two documents are topically similar, the fact that their quality is equally high may add a hint about the closeness of documents in terms of click-likelihood.

	NDCG@5	NDCG@10	MAP
Text [weak baseline]	0.4944	0.5204	0.4476
Text + Original Click Counts [strong baseline 1]	0.6261	0.6160	0.5127
Text + RandomWalk/Discounting [strong baseline 2]	0.6177	0.6159	0.5110
Text + Regularized Click Counts			
topic	0.6327	0.6254	0.5252
topic+quality	0.6327	0.6260	0.5259

Table 1. Experimental results for evaluating effectiveness of regularized click counts. All results by employing click count regularization are statistically significantly better than the weak baseline and two strong baselines.

	NDCG@5	NDCG@10	MAP
Text [weak baseline]	0.4944	0.4108	0.4476
Text + Original Click Counts [strong baseline 1]	0.5508	0.4225	0.4749
Text + RandomWalk/Discounting [strong baseline 2]	0.5592	0.4271	0.4769
Text + Regularized Click Counts			
topic	0.5739	0.4360	0.4915
topic+quality	0.5747	0.4373	0.4921

Table 2. Experimental results for evaluating effectiveness of regularized click counts when only clicks on a document for a query are allowed. All results by employing click count regularization are statistically significantly better than the weak baseline and two strong baselines.

5.4 Robustness

The CQA vertical that we used returns 5 results in an aggregated search result page. Indeed, in many real aggregated search applications, five results are quite a lot, and it could be argued that this number of results should be sufficient to collect user behavior information without click count regularization. However, the number of verticals tends to increase every year and the page length that users are willing to scan is not long. Therefore, there is a good chance that search portals will reduce the number of results from each vertical in the aggregated view. Indeed, we already find that only one or two results from a vertical are displayed in aggregated search services of major search portals. Furthermore, in the mobile search environment, the resources for displaying search results are significantly more limited.

We want to see whether our smoothing technique works in such extreme but still likely cases. To simulate this situation, we get rid of most records from the click logs so that click logs for only one document with the largest number

of clicks for each query remains. Although we keep only one document with the largest click count to simulate a situation where the cut-off rank is 1, this setting should be somewhat relaxed in a real situation because dynamic features of ranking functions may cause other documents to be ranked at 1 over a time span. However, for now, we just assume a simple situation, that is, only one document for a query in click logs.

Using this modified click log, we repeated the same experiments. The same parameters trained with the original click logs are used. Table 2 shows the results. As we see, click regularization works well even when there are clicks on only one document for a query. Click count regularization leads to statistically significant improvements on the strong baselines in all metrics without regard to types of affinity matrices. Furthermore, in this extremely sparse setting, performance gain of our regularization techniques over the strong baselines becomes noticeable. In sum, these results show the robustness of click count regularization in that the technique can address even clickthrough data collected when a more strict resource constraint for aggregated search results is imposed on a vertical search engine.

6 Conclusion

In this paper, we described skews that exist in click log data of a vertical recorded in an aggregated search interface, i.e. rank cut, in addition to the well-known rank bias. In particular, rank cut can cause a serious sparseness problem for clickthrough data. To address these issues, we proposed click count regularization as a click count smoothing technique. This technique addresses rank cut using spectral graph analysis. Through experiments, we demonstrated that click count regularization can yield significant improvements compared to a strong baseline. Furthermore, the robustness of click count regularization was empirically shown by experiments in a simulated situation with only a single retrieved document.

For future work, we will consider various types of queries and verticals. In this work, we focused on a general framework to address skews in vertical clickthrough data and somewhat ignored the various properties of queries and verticals. For example, while some queries in verticals may require diversity of results, others do not. Therefore, we will consider different approaches depending on types of queries and verticals. Furthermore, more unique features of each vertical could be considered as part of defining new affinity relationships.

7 Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NHN Corp. and in part by NSF grant #IIS-0711348. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

1. Acton, F.S.: Numerical Methods that Work. The Mathematical Association of America, second edn. (1997)
2. Agichtein, E., Brill, E., Dumais, S.: Improving web search ranking by incorporating user behavior information. In: SIGIR '06. pp. 19–26 (2006)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: NIPS 14. pp. 585–591 (2001)
4. Bengio, Y., Delalleau, O., Le Roux, N.: Label propagation and quadratic criterion. In: Chapelle, O., Schölkopf, B., Zien, A. (eds.) Semi-Supervised Learning, pp. 193–216. MIT Press, Cambridge, MA (2006)
5. Chapelle, O., Zhang, Y.: A dynamic bayesian network click model for web search ranking. In: WWW '09. pp. 1–10 (2009)
6. Chung, F.R.K.: Spectral Graph Theory. American Mathematical Society (1997)
7. Croft, W.B., Lafferty, J.: Language Modeling for Information Retrieval. Kluwer Academic Publishers, Norwell, MA, USA (2003)
8. Diaz, F.: Regularizing ad hoc retrieval scores. In: CIKM '05. pp. 672–679 (2005)
9. Dupret, G.E., Piwowarski, B.: A user browsing model to predict search engine click data from past observations. In: SIGIR '08. pp. 331–338 (2008)
10. Gao, J., Yuan, W., Li, X., Deng, K., Nie, J.Y.: Smoothing clickthrough data for web search ranking. In: SIGIR '09. pp. 355–362 (2009)
11. Jeon, J., Croft, W.B., Lee, J.H., Park, S.: A framework to predict the quality of answers with non-textual features. In: SIGIR '06. pp. 228–235 (2006)
12. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD '02. pp. 133–142 (2002)
13. Joachims, T.: Transductive learning via spectral graph partitioning. In: ICML 2003. pp. 290–297 (2003)
14. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: SIGIR '05. pp. 154–161 (2005)
15. Lafferty, J., Lebanon, G.: Diffusion kernels on statistical manifolds. The Journal of Machine Learning Research 6, 129–163 (2005)
16. Li, X., Wang, Y.Y., Acero, A.: Learning query intent from regularized click graphs. In: SIGIR '08. pp. 339–346 (2008)
17. Murdock, V., Lalmas, M.: Workshop on aggregated search. SIGIR Forum 42(2), 80–83 (2008)
18. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the web. Tech. Rep. 1999-66, Stanford InfoLab (1999)
19. Radlinski, F., Joachims, T.: Active exploration for learning rankings from clickthrough data. In: KDD '07. pp. 570–579 (2007)
20. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: A language model-based search engine for complex queries. In: Proceedings of the International Conference on Intelligence Analysis (2005)
21. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: et al., T.D. (ed.) Advances in Neural Information Processing Systems. vol. 14. MIT Press (2001)
22. Van Rijsbergen, C.J.: Information Retrieval, 2nd edition. Dept. of Computer Science, University of Glasgow (1979)
23. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval. In: SIGIR '01. pp. 334–342 (2001)
24. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Tech. Rep. CMU-CALD-02-107, Carnegie Mellon University (2002)