Personalized Web Search Ranking

CS 229 : Project Report

Ankit Kumar
ankitkr@stanford.edu

Abstract

In this project, we investigate new approaches to personalize web search result pages for users using anonymized search logs with scant data per user. We investigate modelling users into groups and personalize the user's SERP based on the response of the users' 'peers' towards the documents in the SERP. For evaluation, we compare our personalization strategy against the popular ranking strategy of sorting of documents in the SERP based on overall number of clicks. We quickly realize that personalizing every SERP makes our algorithm perform worse. When then try to learn when to personalize a SERP using a SVM on our chosen features. Although we do not achieve high tranining/test accuracy in learning when to personalize, the accuracy we achieve is sufficient to beat the click sorted ranking scheme.

1. Introduction

This project aims at personalization of web search results according to user preferences. By personalization, we mean the process of showing the same set of (user-independent) documents relevant to the user query to different users in a different order in which he is most likely to consume the documents. We try to develop an algorithm that models preference/interests of a user and accordingly re-ranks the search engine result page (SERP). To model a user, we use the short-term (seesion) user context as well as the long term history of the user on the search engine. One way to formalize this problem is to come up with a "personalization" score for every document which can then be mixed with the query relevance score to give a personalized search experience. Another way of formalization is to classify web pages as interesting/non-interesting to the user based on his click history.

Search personalization is a very important problem as it enables us to provide the user the documents which he is most likely to consume higher in the page. Furthermore, the problem becomes very interesting when the search query is ambiguous. In such cases, user interests can help us greatly in narrowing down the pages the user might be interested in. For example, for the query '"mouse", a biologist might be interested in the animal, while a tech-engineer is most likely looking for computer hardware. Given that majority of queries on web search are short [1], personalization provides great insight in query understanding/disambiguation.

2. Related Work

There is rich literature [2] [3] available on web search personalization strategies and user modelling. Some of the approaches are click-based i.e. try to boost documents which the user has clicked sometime before while others are based on user profiling where we try to represent a user using a set of topics [4] [5] [6] he is interested in (learned from his search/click history) and then boost documents that align with these topics.

There are aso approaches to use the web-hyperlink structure in personalization. For example, proposed a "topic-sensitive" page-rank system [7], a modification of the original page-rank model to this effect.

There are also works on identification of potential queries for optionalization [8]. It is observed that not all queries are suitable for personalization and for only a subset of queries we should personalize to get better gains.

3. Experimental Setup

3.1 Data Set

The dataset [9] used for this project is provided by a popular search engine 'Yandex' that includes anonymized user sessions extracted from Yandex logs of one geographic location during one month of search activity comprising of user ids, queries, query terms, URLs, their domains, URL rankings and clicks. We have about 16GB of uncompressed logs - with 27 days of search logs for training and 3 days for evaluation.

The dataset has 21,073,569 unique queries, 703,484,26 unique urls, 5,736,333 unique users, 34,573,630 training sessions, 64,693,054 click records and 167,413,039 total records.

Given the gigantic size of the dataset, we have decided to sample a small portion of this dataset for faster experimentation and iterations of the learning algorithms. For training we randomly sample 25,000 users and train the model on their search sessions (over 27 days) only. For evaluation, we randomly sample 4000 test users from the aforementioned 25,000 users and take one query per test user (in the 3 day test period).

We also selected 9500 users (separate from the test set users) and take one query per user (which was not used in training set) as the validation set. This set will be used to train a SVM later.

We do not include users with no click history in the training period into our test set or validation set.

3.2 Evaluation Metrics

For evaluation of the search result pages (SERP) reranked by our models, we use the Rank Scoring [10] [11] metric as well as the average rank [12] metrics.

3.2.1 Rank Scoring

The expected utility of a ranked list of web pages is dened as

$$R_s = \sum_j \frac{\delta(s,j)}{2^{(j-1)/(\alpha-1)}}$$

where j is the rank of a page in the list, $\delta(s,j)$ is 1 if page j is clicked in the test query s and 0 otherwise, and α is set to 5 as the original authors did. The final rank scoring reflects the utilities of all test queries:

$$R = 100\frac{\sum_s R_s}{\sum_s R_s^{max}}$$

R_s^{max} is the maximum utility achievable when all documents clicked are on the top of the page. Large value of rank scoring indicates better performance of personalized search.

3.2.2 Average Rank

The average rank of a query s is dened as :

$$AvgRank_s = \frac{1}{|P_s|}\sum_{p \in P_s} R(p)$$

Here P_s denotes the set of clicked web pages on test query s, $R(p)$ denotes the rank of page p. The final average rank on test query set S is computed as:

$$AvgRank = \frac{1}{|S|}\sum_{s \in S} AvgRanks_s$$

Smaller average rank value indicates better placements of relevant result, or better result quality.

4. Our Approach

Due to the fact that the data logs are completely anonymized for query terms and documents as well, many/most of the profiling based apporached described earlier are not applicable as we do not have access to the actual document/queries (rather a bunch of unique ids) in order to classify them into topics of interest. Hence, we investigate a different approach to profile a user based on the set of documents that the user has clicked on in past. This idea explores the hypothesis that the set of documents clicked by a user over a period of time are a good indicator of his interests.

Now, thae dataset we have does not have a lot of data per user (on average 10-15 SERPs per user), hence the user history click-based methods decribed earlier do not work as well. In this approach, instead of trying to predict what a user thinks of a document by looking at his search history for that document, we try to predict what the "peers" of the user think about the document.

The important steps in our approach are as follows. They will be explained in detail in the coming sections.

1. Represent user profile by their click history and cluster users with similar clicks
2. Develop a reranking strategy :
   - Given a SERP L, let $R = randomize(L)$
   - $CSL = ClickSorted(L)$
   - $UCSL = UserClusterClickSorted(CSL)$
   - L is your Yandex ranking, $Borda(L, R)$ is a random strategy, $Borda(L, CSL)$ is our baseline, $Borda(L, UCSL)$ is our approach
3. Train a clasiffier on a traing set to predict when to personalize
4. Use the predictions and evaluate the strategies on a test set

4.1 Model groups of Users

We first try to find groups of users with similar interests based on the profiling we described above. Having clustered users into groups we can rerank the test SERPS based on the response of the 'peers' of the user to each document.

For earch user , we create a 'click vector', where $click\_vector[i]$ = 1 if the user has clicked on document i at some point of time in past. Note that this representation of users will be very sparse. Now, after defining the click vectors for all users, we try to find groups of users who tend to click on same set of documents. That, we believe is a reasonable way of identifying users with similar interests. To achieve this, we use the k-means clustering algorithm on the click-vectors of our training set users. However, given the very high dimensionality of these sparse vectors, we decide to include documents which have been clicked by atleast 0.01% users at some point of time. This reduces the dimesionality of the click vectors immensely and speeds up execution. We run the k-means clustering algorithm for $max\_num\_iterations(70)$ iterations to cluster users into groups.

4.2 Reranking Scheme

Given a SERP, we first ignore the query-relevance of the documents in SERP and just consider them as a list of documents that are to be ranked by some personalization strategy. The personalized ranked list is then combined with the original ranking of the SERP using Borda's rank aggregation method to get the final re-ranked SERP. We now describe the personalization strategy we use to rank an unordered list of documents.

For getting the personalized ranked list, we first randomize the SERP and then sort the documents by the popular strategy of sorting based on overall clicks a document has got historically (in this case, number of users who have clicked on a document). We then again re-sort the SERP obtained by the number of users in the tester's cluster who have clicked on this document. Note that the re-sorting will be a stable-sort, i.e. the documents with same number of clicks from tester's cluster have the same ordering after re-sorting as what they had by the globally click-sorted ranking scheme.

We evaluate the quality of the re-ranked list obtained using the Rank Scoring as well as the average rank metrics described in the previous section. As baseline for evaluation, we compare these metrics for our personalization strategy vs the click sorted ranking scheme.

4.3 Learning when to Personalize

We quickly realized that when personalizing every query, we were taking more losses than gains against the click-sorted scheme and hence, the overall gain by our scheme was negative. Hence, we tried to learn when to personalize a query and when not to. However, as we have not modeled queries in our algorithm, we tried to learn which SERP's should be personalized and which should not. We used a C-SVM [13] with a radial basis kernel function on the training data which represents SERPS using the following features.

1. total number of clicks on any document of the SERP : an indicator of how popular a 'query' is
2. fraction of results in the SERP with any clicks, indicator of navigational queries
3. mean, standard deviation of overall clicks of the results in SERP with clicks, indicator of click distribution
4. average click position on the SERP defined by $\sum_i i \times \frac{overall\_clicks[result[i]]}{\sum_j overall\_clicks[result[j]]}$
5. total number of clicks by the tester's cluster on any document of the SERP : an indicator of how popular a 'query' is in the tester's circle
6. ratio of features 1 and 5, an indicator of how authoritative the tester's cluster is on this kind of SERP
7. fraction of results in the SERP with any clicks by tester's cluster

8. mean, standard deviation of clicks from tester's cluster of the results in SERP with clicks
9. average click position by tester's cluster on the SERP defined by $\sum_i i \times \frac{clicks\_by\_testers'\_circle[result[i]]}{\sum_j clicks\_by\_testers'\_circle[result[j]]}$
10. a measure of how tight assignment of the tester's cluster is : $\frac{max\_dist - dist(tester, cluster\_centroid\_of\_tester's\_cluster)}{\sum max\_dist - dist(tester, cluster\_centroid\_i)}$

We generated training data on the validation set described earlier under our reranking schemes and evaluation metrics and trained the SVM. We selected the model parameters C and γ using a grid search on the validation set. Results showed us that we were not attaining very good accuracy rates. The learning curve shows both higher training and test errors which are flat with increase in training set size indicating an underfitting and the need of better features.

5. Results

In this project we provide comparison between our approach, the Yandex ranking, popular strategy of ranking based on click-sorting and a 'randomization' personalization strategy. It should be noted that there is an inherent bias of the user to click on the results at top of the SERP and we will evaluate assuming the user scanned through all search results before clicking on the document for test queries.

Unless explicitly stated, we report the results for $m = 25,000$ users, $max\_iterations = 70$, number of clusters $k = 50$, number of queries for training SVM = 9500, number of test queries = 3800.

	Rank Scoring	Avg. Rank
Yandex	88.374	2.307
Our approach	81.965	2.894
Click-sorting	81.938	2.896
Random	74.901	3.58

Varying the number of clusters used for clustering k, we get the following end-to-end numbers for our user-cluster based approach:

k	Rank Scoring	Avg. Rank
10	81.941	2.898
20	81.962	2.896
30	81.942	2.895
40	81.920	2.896
50	81.965	2.894
75	81.941	2.894
125	82.002	2.892

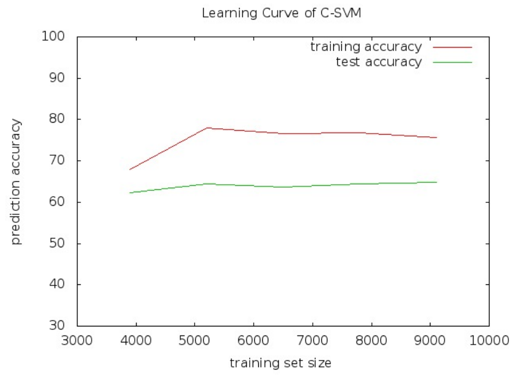

Figure 1: Learning Curve

The learning curve of the C-SVM is shown is fig. 1

6. Conclusion

We are successful in achieving results better than the click-sorted ranking scheme. However, we are still much behind the Yandex scheme of ranking.

We have a couple of ideas that may be used next to improve on the system :

1. In this model, we represented user by the set of document he has clicked (as a bit vector), we would like to also incorporate the number of clicks made by the user on a particular document into our model as a document which is visited by user repeatedly has more to say about the user than other documents.
2. Our model is a click-based model that models users by their click histories and hence trying to optimize clicks. However, if we look at the results obtained above, it looks like the average rank for the baseline (Yandex) ranking is 2.3. While evaluation we did not take into account the fact that the clicks made by the user can be very biased by the ranking itself i.e. user may click on top results even if they do not find it useful. So in a model based on click events, this plays to our disadvantage. We would like our model to incorporate dwell time of the user on a clicked result to identify the actual document that satisfied the user and evaluate based on dwell time and not on click position.
3. As evident by the learning curve of the SVM, more work needs to be done in designing better features for learning when to personalize.

References

[1] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, September 1999.

[2] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web*, pages 581–590, Banff, Alberta, Canada, 2007. ACM.

[3] Uichin Lee, Zhenyu Liu, and Junghoo Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 391–400, Chiba, Japan, 2005. ACM.

[4] A. Pretschner and S. Gauch. Ontology based personalized search. In *Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on*, pages 391–398, 1999.

[5] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 824–831, New York, NY, USA, 2005. ACM.

[6] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 178–185, New York, NY, USA, 2005. ACM.

[7] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM.

[8] Jaime Teevan, Susan T. Dumais, and Daniel J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 163–170, New York, NY, USA, 2008. ACM.

[9] https://www.kaggle.com/c/yandex-personalized-web-search-challenge/data.

[10] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: A novel approach to personalized web search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 382–390, New York, NY, USA, 2005. ACM.

[11] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[12] Feng Qiu and Junghoo Cho. Automatic identification of user interest for personalized search. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 727–736, New York, NY, USA, 2006. ACM.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.