Project Report
Ranking Web Search Results Using Personal Preferences

Shantanu Joshi
Stanford University
joshi4@cs.stanford.edu

Pararth Shah
Stanford University
pararth@cs.stanford.edu

December 14, 2013

Abstract

The diversity of content on the web has exploded exponentially, while our vocabulary to parse it remains at a relative plateau. It is increasingly difficult for search engines to understand what the user really wants by considering only the query terms [1]. For instance, a wildlife photographer searching for "jaguar" is probably looking for information on the animal, while an automobile enthusiast is probably looking for the sports car. In this paper, we present a study of algorithms for personalizing search results based on the user's search history.

1 Introduction

One of the challenges faced by search engines is to identify when personalization is useful and when it is not. Personalisation works best on queries that have a large entropy; continuing our example from above, personalizing search rankings for queries like "Jaguar" is a good idea but is not required for queries like "Google". Another important factor is the difference in a user's long term and short term search history, where short term implies the current query session, and long term is the aggregate of all the query and click history of that user. They are useful at varying times and modeling this correctly can significantly increase the relevance of the search results [2]. For example, users often issue the same query with slight variations in a search session, as they refine their search for a particular resource [3].

The authors of [2] do not prescribe a silver bullet for the search personalization problem and instead suggest and evaluate different approaches which are useful in different contexts. For example, they show that click based methods work best for repeated queries but modeling a user's profile along with a group's preference works better in tackling queries that are not present in that user's search history.

In Section 2, we lay down some prelimnary definitions and formalize the search personalization problem which we address in this paper. Section 3 presents a study of recent work in this domain. In Section 4 we explain the crux of our approach, followed by Section 5 where we describe our experiments on the dataset.The next section provides a summary and evaluation of our results and we conclude with a discussion on possible extensions to our work.

2 Problem Formulation

Although web search personalization is a well-known problem, there are multiple variations depending upon what is included in the scope of personalization. For example, having demographic information about the user would help in personalizing based upon the gender, age-group, country, etc of the user. Similarly, having access to the text of the indexed webpages enables clustering the URLs based on keyword similarity and giving higher ranks to URLs that are closely related to the URLs that were previously clicked by the user.

However, in this paper we focus only on re-ranking based on the click history and dwell time of the user, without assuming that we have access to any demographic information of the user or the text of the result pages. We define dwell time of a particular URL as the time difference between the click on that URL and the next click. For URLs that form the last click of a search session, we assume that the user found the URL of their interest, hence the dwell time should be infinity. (In our experiments we cap this to the maximum dwell time of the data set.) We describe our problem formulation in the rest of this section.

2.1 Formal Problem Statement

Suppose a user U posts a query $\{q_1, q_2, ...q_k\}$ to the search engine, where the $q_i's$ are terms of the query,

and the resulting set of URLs returned by the search engine is $\{d_1, d_2, ...d_n\}$, with the URLs ordered according to decreasing relevance to the query. Our goal is to re-order the URLs to $\{d'_1, d'_2, ...d'_n\}$ in order of their relevance to the user. We define the relevance of a URL to a user for a given query as follows:

- Relevance = 2(*Highly Relevant*), if the user clicked on the URL with the last click in the session, or spent more than 400 time units[1] perusing the URL
- Relevance = 1(*Relevant*) corresponds to links with click and dwell time between 50 and 399 time units (incluscive)
- Relevance = 0 (*Irrelevant*) Corresponds to documents with no clicks or those with clicks and dwell time strictly within 50 time units.

3 Relevant Work

The authors of [4] state that all queries to a search engine can be broken down into three categories: transactional, informational and navigational. The authors focus on the last category and present a new algorithm that correctly predicts which links a user will click on for $\frac{1}{6}$ of all queries.

First, the authors cite a large body of work which supports their claim that navigational queries make up a large amount of the traffic handled by search engines. The solution to the navigational problem is not a one size fits all kind of solution and thus lends its self perfectly to re-ranking of results served by the search engine (The authors of [4] use Bing!). The authors propose a very simple yet effective algorithm which predicts the links that will be clicked on by a user for a given query with a decent level of accuracy. The authors provide a method of computing the entropy of a query and label queries which have a very low click entropy as navigational queries.

In order to avoid outliers and false positives they stipulate that each navigational query must have occurred atleast 10000 times in their logs and must be accompanied by atleast 10000 clicks on the $SERP$[2]. This constraint effectively prunes against queries which have a low entropy because they occur very infrequently (*e.g. Health coverage*) or because the query is answered on the SERP page itself. (*e.g. define schadenfreude*). Having pruned the data they look at the query submitted by the user, try and find two successive queries by the same users that match the current query and if such queries are found where in both the previous cases the user clicked on the same link then the current query is deemed to be a personal navigational query whose prediction is that the user will click the same url as the previous two occasions.

In [7] the author proposes a modification to the widely used Lloyds k-means algorithm which addresses the short comings of k-means with regards to latency, scalability and sparsity commonly found in todays user-facing web applications. The sailent modifcation which we wish to highlight here is that the proposed algorithm takes small randomly chosen batches of the dataset for each iteration, instead of iterating over the entire dataset. From, here the algorithm proceeds as usual by assigning a cluster to each point in the dataset, which are refined with each new batch/iteration through the dataset.Thus, the centroids get updated much faster than the regular K-means update. Like k-means, the new mini-batch k-means algorithm suffers from the problem of getting caught at a local optimum.

4 Core Algorithm

As mentioned before, our approach is to develop a piecewise algorithm that targets different use cases. In the rest of the section we describe two important parts of our algorithm: handling navigational queries and modeling user preferences.

4.1 Handling navigational queries

In our literature survey we observed that the problem of efficiently dealing with repeated query terms to find the same resources had a significant benefit and thus we can get significant gains for personal navigational queries by taking into account only the search history of the current user. The algorithm was outlined in section 3.

4.2 Modeling user preferences

The second part of our algorithm focuses on a more general form of personalization which is more broadly applicable to other types of queries as well. In this algorithm we generate a user profile vector for each user. The user profile vector spans over the space of all the domains encountered in our training set. In this setting, each domain name is associated with a score depending upon features like click rate and dwell time. Having assigned a score to each domain for every user we end up with a sparse matrix $X^{|Users|X|Domains|}$ where $|Users|$ and $|Domains|$ represent the number of users and domains respectively. As a user can only

[1]Kaggle and Yandex will not release the length of 1 time unit
[2]Search Engine Results Page

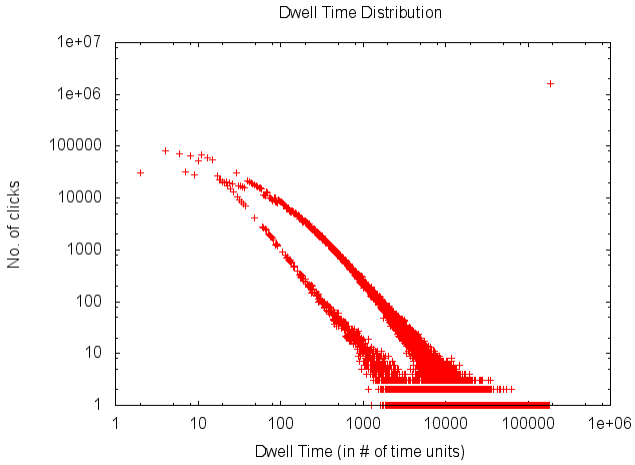

Figure 1: *Frequency distribution of the dwell time of each click.*

visit a small number of domains X is a sparse matrix. The next step is to run k-means clustering on X in order to find similar user profiles. The optimal value of the number of clusters will have to be determined emprically. Once, in a cluster, the scores associated with each domain will be an average of the scores associated with each domain for each user profile vector belonging to the cluster. With this method we will be able to incorporate the preferences of a large number of similar users and leverage this to make better recommendations. Finally, in order to arrive at the best re-ordering of the search results we will identify the cluster to which the user belongs and re-rank the links in order of the most helpful domains as represented by the associated averaged user profile vector.

5 Experiments

5.1 Dataset

The dataset used in our experiments is provided by the search engine Yandex, as part of the Personalised Web Search Challenge on Kaggle [5]. The dataset includes user sessions extracted from Yandex logs, with user ids, queries, query terms, URLs, their domains, URL rankings and clicks. The user data is fully anonymized and includes only meaningless numeric IDs of users, queries, query terms, sessions, URLs and their domains[3].

5.2 Key Statistics

The dataset contains about 21 million unique queries, 700 million unique URLs, 6 million unique users and 35 million search sessions.

We parsed the dataset to compute the distribution of dwell times of each click. In Figure 1, we have plotted this distribution for a subset of 2 million queries. Note that the last click of every session is given a dwell time as the maximum dwell time in the dataset. The lone data point in the top right area denotes that majority of search queries resulted in only a single click, which was assigned the maximum dwell time.

5.3 Personal Navigation

This algorithm is a simple way of providing the search engine with memory to remember a users favorite link(s) for repeated queries. If for example, a user U enters a query q and chooses the document d which the search engine has ranked as the 7th most relevant for him. If the user repeatedly exhibits identical behavior then from the third time q is issued our algorithm will have learnt his preference for d and it will be shown as the top result.

In Figure 2 we have plotted the number and count of the navigational queries for the entire dataset of 21 million queries. The graph shows that a large number of queries occur only a few times (< 10) but we exploit the long tail that any power law distribution has and thus the graph in Figure 2 makes concrete the impact of adding personalization excluscively focused on navigational queries.

5.4 K-Means

After constructing the matrix X (explained in Section 4) we ran a naive implementation of k-means on the matrix hoping to extract the user-clusters. However, since X has slightly more than 5.9 million rows and 5.2 million columns the algorithm was painfully slow and did not scale to such a large dataset. This was when we decided to adopt the mini-batch k-means mentioned in [7]

5.5 Mini-Batch K-Means

Mini-Batch K-means is a highly scalable algorithm that can provide very good results for unsupervised clustering over very large and sparse datasets,like the one we have. The algorithm takes as input the number of centroids k, the batch size b and optionally a parameter that specifies the maximum number of iterations over the dataset after which the algorithm stops irrespective of other conditions. The formal specification of Mini-Batch K-means as illustrated in [7] is shown in Algorithm 1

[3]Yandex has permitted the use of this dataset for academic research.

Algorithm 1 Mini-Batch K-Means

$k \leftarrow NumberOfCentroids$
$b \leftarrow BatchSize$
$X \leftarrow GivenSparseMatrix$
$limit \leftarrow NumberofIterations$
$k \leftarrow$ Randomly choose k rows from X
$v \leftarrow []$
for i = 1 to limit **do**
    $M \leftarrow b$ rows randomly chosen from X
    for all $x \in M$ **do**
        distance[x] $\leftarrow$ DistanceToNearestCenter(x)
    end for
    for all $x \in M$ **do**
        c $\leftarrow$ d[x] // get the nearest center from x
        $v[c] \leftarrow v[c] + 1$ //Count of x per center
        $\eta \leftarrow \frac{1}{v[c]}$//Learning rate per center
        $c \leftarrow (1-\eta)c + \eta x$ //gradient descent update
    end for
end for

BatchSize	K	Fraction in Largest Cluster
20000	65	0.40
25000	15	0.67
25000	55	0.379
25000	75	0.34
25000	155	0.372
50000	65	0.30
200000	65	0.396
250000	10	0.6
250000	35	0.88
250000	75	0.73
250000	100	0.35

Table 1: Clustering Results for different values of the parameters

The quality of results obtained by this algorithm is largely dependent upon the values for b (Batch Size) and k (Number of Centroids) provided the number of iterations has been set to a relatively high value (300 in this case). We ran many simulations for different values of the parameters in order to find a good clustering. Our aim was to choose values of these parameters which avoid overly large (and as a consequence extremely small) clusters. Table 1 shows the results of our experiments

An analysis of the results in Table 1 reveals that increasing or decreasing one parameter alone has no verifyable trend and that both the batch-size and number of centres need to be varied to find the sweet spot which gives rise to the best clustering.

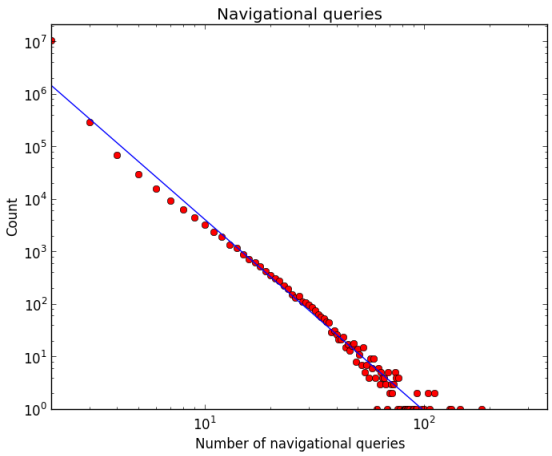

Figure 2: *Plot above shows that navigational queries obey the power law with a slope of 3.52. A point on this graph tells us the number of navigational queries x that occur y times*

6 Evaluation

The results of our alogrithms were evaluated using NDCG (Normalized Discounted Cumulative Gain [6]) measure, which will be calculated using the ranking of URLs for each query in the test set, and then averaged over all queries.We used the Kaggle interface to submit our predictions on the test set.

6.1 Evaluating Mini-Batch K-Means

While intuitively, it feels wrong that a good clustering of the users would involve a single very large cluster we ran mini-batch k-means on those cases as well as the ideal ones to verify our intuition with numbers. The results are summarized in Table 2.

To put our scores in context, the non-personalized baseline i:e the score for the current output of the Yandex search engine is 0.79056. Looking at these scores, we thought a possible reason for k-means doing worse than the base line was the large number of test-queries it modified (over 95%). Thus, it was our hypothesis that mini batch k-means was suffering from overfitting. The personalizations it got wrong were hurting our score more than the personalizations it got right. Hence, we felt that adopting a more selective approach to applying the recommendations of mini-batch k-means might counter the overfitting.

The idea that we came up with was to only change the top 3 links and leave the rest as they are. This would minimize the magnitude of the changes the algroithm made while ensuring we could still benefit significantly from the personalization recommended by

BatchSize	K	Largest Cluster Fraction	Score
250000	35	0.88	0.62579
250000	30	0.66	0.60830
200000	65	0.396	0.59264
50000	65	0.41	0.68189

Table 2: Results for mini-batch k-means used to personalize links over the test set

our algorithm. The last row in Table 2 shows the result of applying this change.

Our hypothesis was proved right and there was a significant jump in our score from 0.59 to 0.68.

6.2 Personalizing Navigational Queries

This algorithm provides deterministic results and in contrast to k-means changes only a small fraction of the queries (16%). However, we can be confident that any change advocated by this algorithm will be a positive one. As expected this algorithm performed much better than the baseline and scored 0.79397, propelling us to our highest ranking of 15^{th} out of the 111 teams.

6.3 Combining k-means and Personal Navigation

The final step in our investigation of mini-batch k-means and personalizing navigational queries was to combine the two algorithms. We did this by feeding the results of k-means as input to the personalization algorithm. Our resulting score for this algorithm was 0.68286 which is slightly better than the result of running a selective version of k-means but still worse than the navigational algorithm alone. On closer analysis this result is not suprising as personalizing navigational queries on the output of mini-batch k-means can only improve the result obtained by k-means but provides no guarantees on improving the result of applying just the alogrithm used to personalize navigational queries.

7 Conclusion and Future Work

From our results above it is clear that k-means and personalizing navigational queries are most useful in personalizing search engine results when they are used as two distinct algorithms instead of being merged into one. The former algorithm tackles the case when a user issues a query that he has never used before while the latter leverages an users search history to provide better results when an user inevitably repeats a query.

The clustering algorithm to deal with unseen queries can be expanded further to provide even better performance. In this paper we have investigated the impact of k-means and mini-batch k-means on search personalization, applications of other clustering algorithms like Means Shift, Spectral clustering and Hierarchical clustering could be explored in the future.

References

[1] 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).

[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 (WWW '07)

[3] Milad Shokouhi, Ryen W. White, Paul Bennett, and Filip Radlinski. Fighting search engine amnesia: reranking repeated results. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '13)

[4] Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. Understanding and predicting personal navigation. In Proceedings of the fourth ACM international conference on Web search and data mining (WSDM '11)

[5] Yandex Personalized Web Search Challenge On Kaggle: Data https://www.kaggle.com/c/yandex-personalized-web-search-challenge/data

[6] Kalervo Jarvelin, Jaana Kekalainen: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)

[7] Sculley D.Web-Scale K-Means Clustering. In Proceedings of the 19th international conference on World Wide Web (WWW '10)