Improving Product Search with Session Re-Rank

a Walmart data mining project

Charles Celerier (cceleri), Bill Chickering (bchick), and Jamie Irvine (jirvine)

June 7, 2013

Walmart.com maintains an online catalog of over 2M products. Consequently, enabling users to quickly find products that conform to their specific needs and tastes is especially challenging. Given the difficulty of its task, Walmart.com's product search engine does an impressive job in interpretting the user-provided query and rapidly returning relevant results. Yet, there remains highly significant information that is not fully leveraged. The details of a user's online shopping session are indicative of a user's intent and compliment–indeed, provide context for–the user-provided query. In this report we describe and analyze a ranking scheme we call *Session Re-Rank* that can potentially induce a large increase in both click-through-rates and conversions on the first page of query results.

1 The Technique

Session Re-Rank works by comparing previously clicked items with the top N items returned by the search engine in reponse to a query. Items to be shown that are sufficiently similar to previously clicked items are promoted. The extent (i.e. number of positions) of the promotion for a particular item is a function of its similarity to previously clicked items, its original position, and the promotions of other items.

The similarity between an item to be shown and a previously clicked item is determined within five distinct vector spaces: *click-space*, *cart-space*, *query-space*, *title-space*, *item-space*. The non-unique representation of an item within each of these spaces may be thought of as a binary vector or a set of objects. (MapReduce jobs process historic query data to construct indexes whose keys are itemids and values are lists of the appropriate objects. Great care went into ensuring that index entries can be accessed in $\mathcal{O}(1)$ and that two entries can be merged to compute their intersection or union in linear time.) The similarity $J_s(A, B)$ of two items, A and B, within a particular space s is determined using Jaccard similarity. Similarities within particular spaces are then weighted and summed to determine the composite similarity

$$S(A, B) = \sum_s C_s (J_s(A, B))^{\alpha_s},$$

where C_s and α_s are tuning parameters. The score σ attributed to an item to be shown is then the summation of composite similarities between itself and all previously clicked items plus the click-through-rate (CTR) Γ_i of the item's original position i

$$\sigma = \sum_{B \in P} S(A, B) + \Gamma_i,$$

where P is the set of previously clicked items.

2 Similarity Spaces

The premise behind *click-space* is that two items are similar if they are both clicked within the same online shopping session. The dimensions, or objects, of this space are therefore past user-sessions. The *clicks-index* for the data presented in this report was contructed using approximately half of the Walmart provided data, or about 60M queries (about 120M page views).

Cart-space is based on the notion that two items are similar is they ever appear in a shopping cart together. The objects of this space are therefore shopping carts. The *clicks-index* for the data presented in this report was contructed using approximately half of the Walmart provided data.

Items are also considered similar if they appear in a query together. The objects of *query-space* are therefore queries. We make a distinction, however, between *user-queries* and *unique-queries*. The former are the well-defined entities within the raw Walmart data. The latter is an abstraction based on the notion that multiple *user-queries* can correspond to a single *unique-query*. To derive *unique-queries* from our data, we cluster *user-queries* as follows: two *user-queries* with the same search attributes (e.g. category or price filters) are considered the same *unique-query* if the strings constructed by concatenating the space-seperated, stemmed (we use the Python stemming.porter2 module), forced to lower-case, terms from each of their rawqueries are equal. We point out that while we achieved better results with this policy compared to simply using *user-queries*, we have no reason to believe that this is the ideal way to cluster queries for use within *Session Re-Rank*. Indeed, we believe one way to improve *Session Re-Rank* is to optimize the query clustering policy.

Title-space is straightfoward: each item is associated with a set of terms from its title. We ignore case, but at present do not stem, discard stop words, or weight terms in any way.

Finally, the structure of *item-space* is unique because it involves a level of indirection. The premise here is that if items A and B are clicked in a single user-session and items A and C are clicked in another user-session, that items B and C are similar because they have item A in common. In this way, a large number of relationships between items is created. *Item-space* resembles *click-space* in that if two items are clicked during a single session, they will have nonzero similarity. It differs from *click-space* in two key respects, however. First, items that have historically never been clicked in the same session can have nonzero similarity if they were each clicked with a common third item. Second, if items are clicked together in many session this will increase their Jaccard similarity in *click-space* but not in *item-space*.

3 The Data

Walmart.com has generously supplied us with a large dataset consisting of about 250M pageviews comprising about 120M query results which occurred over about 30 days. The data includes the user-provided rawqueries together with search attributes, visitorids and sessionids, shown items, clicked items, which items were placed in a shopping cart, and which items were ultimately purchased. In addition, they have provided detailed item information including item title, description, category, and other details. We leverage this data in three ways. First, we re-structure the data into indexes that allow us to identify relationships between items in realtime. Second, we isolate subsets of the data in order to conduct experiments and perform measurements, which enable us to refine our technique and optimize, or tune, the associated parameters. And third, once developed and optimized, we conduct experiments on previously withheld data and analyze the results in an effort to determine the efficacy, or lack thereof, of our technique.

4 Our Technique vs Our Experiments

An important distinction should be made between the *Session Re-Rank* technique and the experiments described in this report. Both the technique and the experiments leverage the provided data–however, the experiments are simulations and limited ones at that. A key limitation is that the provided query data is confined to what the user was actually shown. That is, the search engine may have identified several pages worth of results in response to a *user-query*, but the dataset consists only of those pages rendered to the user. Meanwhile, the concept behind the *Session Re-Rank* technique calls for a search engine to deliver to the algorithm the top N items in response to a *user-query independent of the number of items ultimately shown to the user*. As a consequence, it is difficult to simulate our technique using shown query results that are truncated because a user only viewed one or two pages. More generally, the use of historic data to demonstrate the consequences of a online ranking algorithm is intrinsically limited since one cannot be certain how users would have behaved if presented with different results. Nonetheless, we have done our best to conduct the most fair and informative experiments and analyses.

5 Experiment 1

The goal for experiment 1 is to simulate *Session Re-Rank* using the provided historical query data, which is limited to what users were actually shown. Since an online implementation of our technique would receive the top N items from the search engine for re-ranking prior to showing any results to a user, the final ranking would be independent of the total number of items actually shown (e.g. the number of page views requested

by a user). We therefore limit the test set of experiment 1 to queries where either at least $N = 100$ were shown or the number shown is not divisible by 16. The reason for this is that Walmart.com provides two options for the number of items shown per page: 16 or 32. By performing the experiment on this subset of the data we preclude queries where the top N items are not available to our algorithm. The choice of $N = 100$, meanwhile, is somewhat arbitrary and was made by balancing our desire for a large test set with our desire to use a value comparable to what would be appropriate for an online implementation. It is therefore quite possible that a larger value of N (e.g. 1000) would achieve better results in the actual online scenario.

Confining the experiment to this subset of queries allows our algorithm to re-rank the same number of

6 Results

Dataset	000050_0.test_data.filtered
Index	48chunk
Parameters	
coeff_carts	free or 0.00
coeff_clicks	free or 0.00
coeff_ctr	1.000
coeff_item_title	free or 0.00
coeff_items	free or 0.00
coeff_queries	free or 0.00
coeff_rank	0.000
ctr_by_position	16chunk.CTRs.json
exp_carts	0.400
exp_clicks	0.500
exp_item_title	0.800
exp_items	0.800
exp_queries	0.500
exp_rank	1.300
insert_position	2
k	100
n	100

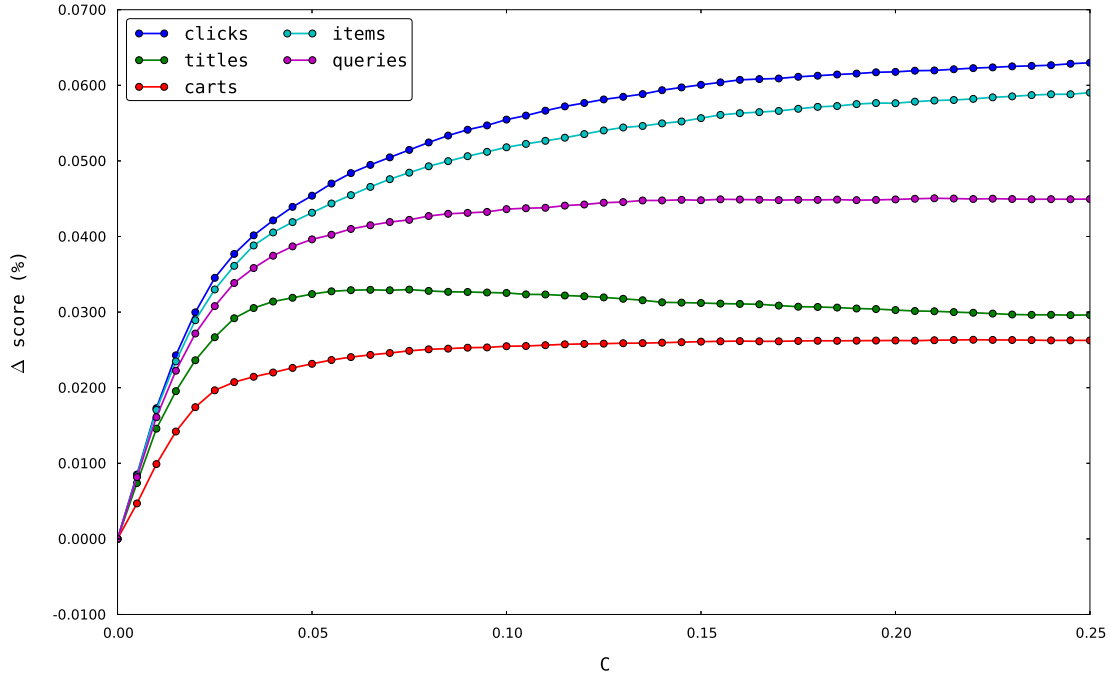

Figure 1: % increase of position score

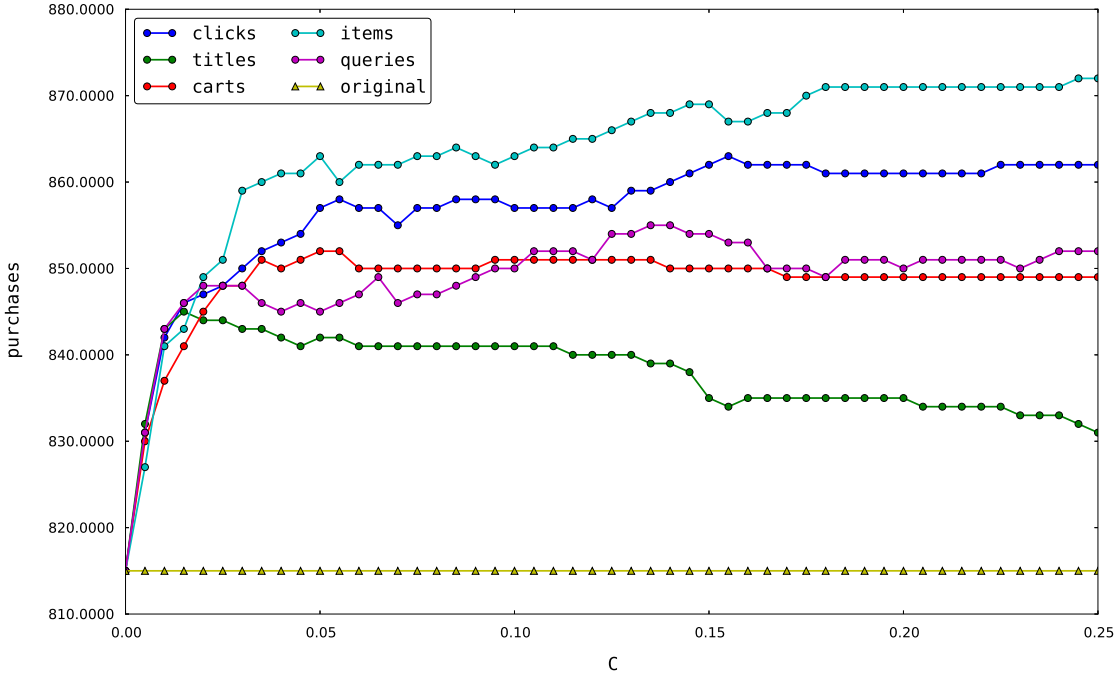

Figure 2: total purchases on the front page