Learning to Rank for Personalized E-Commerce Search at CIKM Cup 2016

Joao Palotti
Vienna University of Technology
Austria
palotti@ifs.tuwien.ac.at

ABSTRACT

This paper reports on my experiments for the CIKM Cup 2016 Track 2: Personalized E-Commerce Search Challenge. E-Commerce search is relatively not so well studied as other search tasks, such as Web search, and therefore this challenge provides an excellent opportunity for both academia and industry researchers to test their ideas and approaches in this domain. Here I report on my Learning to Rank approach, along with my highlights and mistakes.

Keywords

CIKM CUP; Learning to Rank; Personalized E-Commerce Search; Product Search; XBoost

1. INTRODUCTION

Shopping online has become a popular activity. It can be shown by the number of giant e-commerce companies such as Amazon, Alibaba, Etsy and Ebay; or by recent polls like the one conducted by the investment bank Piper Jaffray and reported at Pew Research blog stating that more than three-quarters of teens frequently shop online [3].

Unlikely offline stores, online stores can potentially keep huge catalogs of all their products and show every single one to their users. However, there is no single brave online user who would be patient enough to visualize the whole catalog of such online stores. Users are already drowned by choices in any corn flacks [pratileira] in a supermarket. In many specialized stores, like jewelries or cloth stores, consumers can rely on clerks for tips and suggestions. Similarly, online stores rely on specialized systems, such as recommender systems, to provide the users with suggestions that they might like.

Much of this new technology is concomitantly developed in the academia and in the industry. However, the datasets freely available for academic users are often limited or outdated. To bridge these two worlds, competitions such as the CIKM CUP play a key role. In this year edition two tracks were proposed and I present in this report the experiments conducted by me during the CIKM CUP 2016 Track 2: Personalized E-Commerce Search Challenge. This competition was hosted at Codalab and lasted two months from August 5 to October 5 2016.

The proposed challenge of this competition was to predict the search relevance of products given the personal shopping, search and browsing behavior of a large number of users. Rich anonymized search and browsing logs were provided along with anonymized product data and transactions made.

CIKMCUP '16 Indianapolis, Indiana USA

2. DATA

The company Diginetica[1] and their partners provided the participants with a total of 923,127 queries, from which 636,160 (69%) were used as training data. This data was divided into two groups:

1. *query-less interactions* containing data on browsing history. There are a total of 1,005 possible categories available in the dataset. Users could browse on one or more category;

2. *query-full interactions* containing conventional query search. There are 26,138 unique anonymized search queries made from January 1st to June 1st of 2016. Queries have on average 2.66 terms and 23% of the queries have only one term, similarly to what is found on regular Web search [7, 4].

Product data was also provided: 184,047 products with their log2 transformed price and anonymized product name were made available. On average, 5.11 terms were used in product's names, and the average price found was 5.11 monetary units (as the product price was logarithmic transformed, the real average product price was around 34.5 monetary units).

Finally, the organizers provided data on 18,025 product purchases, 1,127,764 product clicks and 1,235,380 product views. More information and the description of the dataset can be found online at https://competitions.codalab.org/competitions/11161#learn_the_details-data2.

3. EVALUATION

The evaluation measure used in this competition was the Normalized Discounted Cumulative Gain (NDCG) measure, calculated using the ranking of products provided by participants for each query, and then averaged over all test queries. There were three grades for relevance: 0 – irrelevant: grade

[1] http://diginetica.com/

```
QueryId, ItemId, ItemView, ItemClick....Label
      1,   1000,       12,         5....... 2
      1,   1001,        1,         0....... 0
      1,   2222,        6,         0....... 0
      1,   2929,        5,         1....... 1
.........................................
............many other examples.............
.........................................
   200,    2929,        5,         1....... 0
   200,    9003,       10,         0....... 1
```

Figure 1: Partial examples of the training set.

corresponds to the products with no clicks; 1 – somewhat relevant: corresponds to the products, which were clicked by the user. 2 – relevant: grade corresponds to the products that were clicked and purchased by the user. If a product was purchased several times (e.g. three items of the same kind), it is still used 2 as a relevance grade.

Also, weights were attributed to each group of queries. The weight for the *query-less* group was 0.8, while the weight for the *query-full* group was 0.2. These arbitrary weights followed previous data analysis done by the data provider.

4. MY APPROACH

As I am familiarized with Learning to Rank (L2R), I decided to participate in this competition with a L2R approach. In L2R, a machine learned model is used to re-rank a given list of documents. As in any supervised learning method, the set of training examples is crucial. A training example can be defined as a vector of integer values containing its representation (features), and a label stating how important an example is. Figure 1 partially shows how the training examples look like. In Figure 1, 2 queries are shown. The first one has id 1 and has at least 4 items, while the other, with query id 200, has at least 2 items. Two features are represented, ItemView and ItemClick, which are values extracted from the training query logs representing the number of times a given item was viewed and clicked, respectively. The last column of each training example is its label. The larger the value the more important it is for a query.

A variety of machine learning models were tested, as well as a number of representations for the training examples. Section 4.1 expands the above example showing the full range of features used in this work. Then, Section 4.2 describes the frameworks and machine learning methods tried, and, finally, Section 4.3 describes the methodology used and problems found during the competition.

4.1 Representation

In a learning to rank approach, a model is learned based on a large group of training examples. Over the two months of this competition, a number of features were inclemently created. Table 1 shows all features used in the final model created. The features were divided into 4 different types: item-dependent, query-dependent, session-dependent and item-query-dependent, according to the kind of information used. Item-dependent features, for example, are features directly linked to an itemId, as itemView and ItemClick for itemId 2929 in Figure 1.

Roughly following the labels used for evaluation, we defined 3 possible labels for each example. Label 0 is used for irrelevant examples, label 1 was used for examples that were either *viewed* or *clicked* for a given query, and label 2 was used for items that were *purchased* for a given query. Slightly different from the grade labels used by the organizers (see Section 3), we considered an item that is viewed with the same label than an item that is clicked.

4.2 Frameworks

There are a large number of open-source frameworks available for learning to rank. SVMRank[2][5], RankLib[3], sofia-ml[4][9] and XGboost[5][2] are some of them that deserve a highlight as they were all tested during my participation in this challenge.

Initially I experimented with Ranklib because of its great range of algorithms containing both pairwise approaches (e.g. RankNet and RankBoost) and listwise approaches (e.g. AdaRank and ListNet). However, as more and more features were added to my models, the training time became a problem for me. As I was not willing to wait more than 24 hours to test a model, nor willing to use less data to train my models, I decided to move to other frameworks. My experience with sofia-ml was much more satisfactory in terms of training time, but the effectiveness of my methods were slightly worse (using the partial test set from the leaderboard). Finally, I tried XGboost and its boosted trees just were amazing, both in terms of efficiency and effectiveness. My best submission was done with XGboost.

4.3 Methodology

In this section I briefly describe my approach for validation and the separation of query-less and query-full sets, which were my worst and best decisions.

4.3.1 Validation Set

Usually a validation set is used in order to understand the impact of new features, check the effectiveness of a L2R model or simply to tune hyperparameters. However, a validation set for temporal datasets such as the one of this competition is very challenging to create. During the first month of competition, I struggled to have a validation dataset which I could trust, but I fail to build it. Therefore, the only way to evaluate my different approaches was to explore a trial-and-error method exploring the public scores from the leaderboard, and hope that no big surprises were going to happen when the official test set was released.

4.3.2 Separating Query-less and Query-full sets

Separating the training and test dataset into *query-less* and *query-full* sets significantly increased the effectiveness for query-full queries. Because of that, I decided to optimize the *query-full* set first and then work on the *query-less* set. Many features were then created exclusively for the *query-full* set: query length, jaccard coefficient between query text and item text, the past NDCG and MRR scores, and so on. For the past NDCG and MRR scores, for example, the average score of each query was calculated based on the query

[2]http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[3]https://sourceforge.net/p/lemur/wiki/RankLib/

[4]https://code.google.com/p/sofia-ml/

[5]https://xgboost.readthedocs.io/

Type	Category	Feature Name	Description
Item-Dependent	Popularity Signs	Item View	Counts the number of times an item was viewed in the training data
		Item Click	Counts the number of times an item was clicked in the training data
		Item Purchase	Counts the number of times an item was purchased in the training data
		Unique Users Viewing	Counts the number of unique users view this item
		Unique User Purchase	Counts the number of unique users purchasing this item
	Ranking Signs	Item Mean Rank	Mean rank position of an item across all SERP with that item
		Item Median Rank	Median rank position of an item across all SERP with that item
		Item Max Rank	Best rank position of an item across all SERP with that item
		Item Min Rank	Worst rank position of an item across all SERP with that item
	Textual Signs	Item Name Length	Number of terms in an item name
	Price Signs	PriceLog2	Raw price value for an item
Query-Dependent	Ranking Signs	Rank Size	Number of items for this query
	Textual Signs	Query Length	Number of terms used for this query
	Price Signs	Max Price	Highest price of any item for this query
		Min Price	Lowest price of any item for this query
		Mean Price	Mean price of items for this query
		Median Price	Median price of items for this query
		Price Range	Difference between the highest and the lowest item price for this query
	IR Measure Signs	NDCG	The previously calculated NDCG for this query
		MRR	The previously calculated Mean Recipoical Ranking for this query
	Length Sign	Average Duration	Average duration for this query
Session-Dependent	IR Measure Sign	NDCG Per Session	Average value for NDCG for this session ID
	Length Signs	Queries Per Session	Average number of queries for this session ID
		Average Session Duration	Average duration for this session ID
Item-Query	Popularity Sign	Count Clicks	Count the number of times this item was clicked when this query was issued
	Ranking Sign	Original Rank	The original position of this item in this query
	Price Sign	How Expensive	The normalized price of this item compared to other items for this query
	Textual Sign	Jaccard Coefficient	Size of the intersection of terms between item description and query terms
	Groups Sign	Category Percentage	The percentage of items of the same category of this item among the items for this query

Table 1: All features used in my best model divided into groups

terms. For example, the query terms "*528941,529116*" were together issued 12 times, 8 in the training set and 4 in the test set. The average NDCG for this 8 times in the training set was 0.3690, and this value was used for the feature *NDCG* for all 12 examples in the dataset.

Unfortunately, there was not enough time for me to properly explore the *query-less* set, and that was a big mistake, as the evaluation weights of this competition were highly biased towards the *query-less* set.

5. LESSONS LEARNED AND CONCLUSION

I really enjoyed a lot working on this competition. Unfortunately, I had limited time to do so, given the high load of work from my last year of PhD. My biggest mistakes were failing to have a proper validation set, and not spearing enough time to work on the *query-less* set of the challenge. The biggest highlights of my approach were the creation of a large amount of features, given the time frame of this competition, the division of *query-less* and *query-full* sets, and the exploration of a number of frameworks for learning to rank. As future work, among the many things to do, I would explore the use of fusion methods to combine the output of different L2R method (for example, using Condorcet fusion [6] or Borda Count [1] to combine rankings created by different L2R methods), and explore the use of recommender systems complementary to the L2R approach used.

My code is available at https://github.com/joaopalotti/mycikmcup2016 along with a short documentation on the parameters that I used.

6. REFERENCES

[1] J. A. Aslam and M. Montague. Models for metasearch. In *Proc. of SIGIR*, 2001.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[3] D. Desilver. Shop online? Many teens do it, but more prefer the store. Technical report, The Pew Internet & American Life Project, June 2013.

[4] B. J. Jansen and A. Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing & Management*, 42(1):248 – 263, 2006. Formal Methods for Information Retrieval.

[5] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[6] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *Proc. of CIKM*, 2002.

[7] J. Palotti, A. Hanbury, H. Müller, and C. E. Kahn. How users search and what they search for in the medical domain. *Information Retrieval Journal*, 19(1):189–224, 2016.

[8] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke. Lerot: an online learning to rank framework. In *Living Labs for Information Retrieval Evaluation workshop at CIKM.*, 2013.

[9] D. Sculley. Large scale learning to rank. In *NIPS Workshop on Advances in Ranking*, pages 58–63, 2009.