# Gradient Boosted Decision Trees for Personalized E-Commerce Search at CIKM Cup 2016

Martin Wistuba
Institute of Computer Science
Universitätsplatz 1
Hildesheim, Germany
wistuba@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
Institute Computer Science
Universitätsplatz 1
Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

## ABSTRACT

In this work we describe our solution for the CIKM Cup 2016 Track 2: Personalized E-Commerce Search. We describe the features used and generated for solving the task and perform an analysis of which of these features are useful and which are not. Finally, we review the state of the art of (personalized) ranking methodologies and show possible directions on how to improve our approach.

## CCS Concepts

**•Information systems → Personalization; Collaborative search; Learning to rank;**

## Keywords

CIKM Cup 2016; Personalized Ranking

## 1. INTRODUCTION

Recommendation is a very important task for many search applications. Traditional examples are for example the recommendation of movies, videos or music. The most successful systems for these applications are personalized which is not very surprising. The labels for these applications are generated using explicit feedback. This means, the user rates how much he enjoyed a movie and then the machine learning system predicts the rating for all non-watched movies and recommends those with highest predictions.

Then, there are applications where no explicit feedback is given. These are for example web search engines or search engines for online shops. Even though some shops offer the customer the possibilty to rate the provided search results, this option is rarely used. Thus, most of the information observed is the user's browsing behaviour such as clicks on links to web pages or products or other actions such as likes or purchases. The problem of this feedback is that it only positive feedback is observed. If a customer does not like on a link or does not buy a product it does not necessarily mean that she does not like it. Maybe the customer overlooked it, maybe she liked the other product more but maybe it was indeed a bad recommendation.



*CIKM '06 Indianapolis, Indiana USA*



DOI: 10.1145/1235

## 2. PROBLEM DEFINITION

The goal of the CIKM Cup 2016: Personalized E-Commerce Search Challenge has been to rank the products return by an e-commerce search engine for each search according to their relevance levels. A distinction is made between three different relevance levels, ordered from highest to lowest:

1. Products the customer purchased during the session.
2. Products the customer clicked during the session.
3. All other products.

The task of the participants of the challenge was to predict the correct ranking according to these relevance levels. Furthermore, the challenge distinguished two different kinds of searches. Query-full searches are searches where the customer provided a search string query while query-less searches do not contain any search queries. Instead, the customer clicked on a specific product category and then a list of products within this category is returned. The predicted rankings are evaluated using the normalized discounted cumulative gain (NDCG), i.e. the discounted cumulative gain (DCG) divided by the best DCG. The DCG@p for the challenge was defined as

$$\text{DCG@p} := \sum_{i=1}^{p} \frac{2^{\text{rel}_i} - 1}{\log_2 (i+1)} \tag{1}$$

where p was set to the number of products presented as a result of each each search. The final challenge score is the weighted average between the NDCG of query-full and query-less queries with weights 0.2 and 0.8, respectively.

The organizers provided a baseline that gives each item $i$ a score of

$$\hat{y}(i) = 3p_i + 2c_i + v_i \tag{2}$$

where $p_i$, $c_i$ and $v_i$ are the number of purchases, clicks and views of item $i$, respectively.

## 3. DATA

To enable the participants to apply supervised machine learning to automatically rank the products for each search, the organizers provided business data in multiple files. A file describing all products is provided. For each product its price, its anonymized category and its product name is

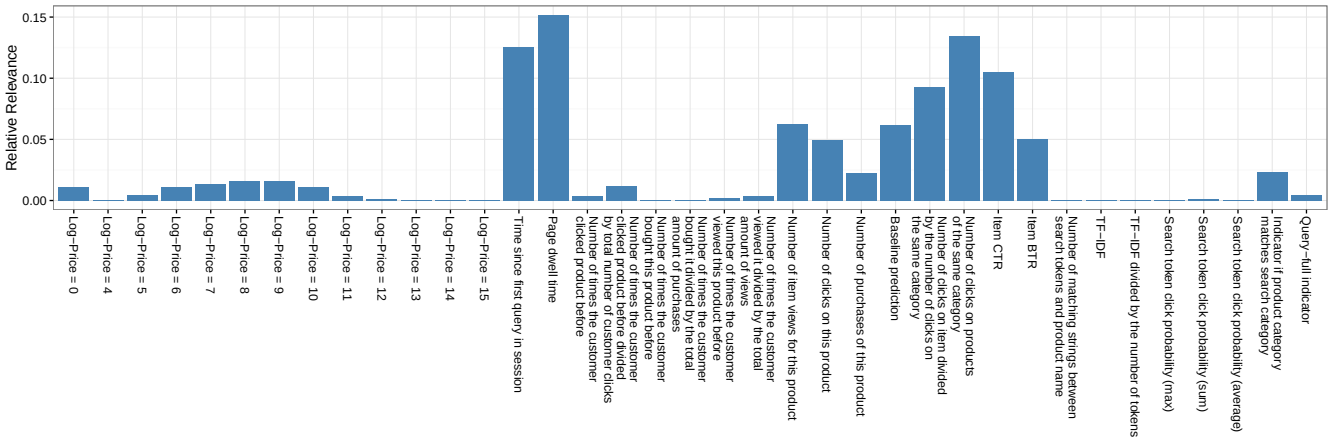


**Figure 1: The relative frequency a feature was chosen as a split criterion for a tree. The number of views, clicks and purchases are one of the most predictive features.**

provided. For the customers only the browsing, clicking and buying logs are available. No additional information is provided. Furthermore, the data provides an additional challenge for latent factor models which are often used for recommender systems. For many searches the customer is anonymous, are new customers or customers with only few observations.

For each search following information is provided

- Customer ID (in many cases this information is not available)
- IDs of the products to be ranked
- Time since the first query in a session
- Page dwell time
- Date of the event
- For query-full searches: query string tokens
- For query-less searches: category ID

Every word in the data has been anonymized by the organizers by mapping each word to a unique number. This affected the query string tokens and the product names.

The training data contained 636,160 searches with 92,271,275 products to be ranked. 184,048 unique products are described. Each search had up to 200 products to be ranked, on average 145. Only about 5% of the searches are query-full.

# 4. PREPROCESSING

For our own preprocessing we made no distinction between purchased and clicked items. Hence, the label for a product was either one, if it has been clicked or purchased and otherwise we set it to zero.

We used following features directly as provided.

- The price category of the product (Hot-Encoding)
- Time since the first query in a session
- Page dwell time

We did not consider which other items have been bought in the same session or the date of the events.

Furthermore, we engineered following additional features.

- If the customer information is given we created following features to enable personalized predictions
  - The number of times the customer clicked the product before
  - The number of times the customer clicked the product before divided by the number of total clicks by this customer on products
  - The number of times the customer bought the product before
  - The number of times the customer bought the product before divided by the number of total purchases by this customer
  - The number of times the customer viewed the product before
  - The number of times the customer viewed the product before divided by the number of total views by this customer
- Number of views of this item
- Number of clicks on this item
- Number of purchases of this item
- Challenge baseline prediction (3 * purchases + 2 * clicks + views)
- Number of clicks on items of the same category
- Number of clicks on item divided by number of clicks on category
- Item click through rate: number of clicks on item divided by the number of times the item was presented
- Item buy through rate: number of purchases of the item divided by the number of times the item was presented

| | Weighted NDCG | NDCG (query-full) | NDCG (query-less) |
|---|---|---|---|
| 1. minerva | 0.4262 | 0.5574 | 0.3935 |
| 2. Dmitrii_Nikitko | 0.4149 | 0.5301 | 0.3861 |
| 3. tjy | 0.4056 | 0.4570 | 0.3928 |
| 4. wistuba | 0.3769 | 0.4495 | 0.3588 |
| 5. joaopalotti | 0.3712 | 0.4860 | 0.3425 |

**Table 1: Final results on the test leaderboard for the five winning teams.**

- If the query is a query-full query
  - Number of matching strings between query tokens and product name
  - TF-IDF of search tokens and product name
  - TF-IDF divided by the number of query tokens
  - Computation of the empirical click probability of each query token for this item. Then we used as features
    - The maximum of these probabilities
    - The sum of these probabilities
    - The average of these probabilities
- Indicator whether item category matches search category
- Indicator whether it is a query-full search

In Figure 1 one can see our investigation on how important which features are by counting how often a feature was used as a split criterion for a decision tree. As already identified by the baseline provider, the number of views, clicks and purchases are very important. Futhermore, information about the price, the session duration and the page dwell time are useful. Finally, user specific features seem to improve the performance by giving personalized recommendations.

# 5. ALGORITHM SELECTION AND HYPERPARAMETER OPTIMIZATION

We choose gradient boosted decision trees [4] as a strong ensemble model and optimized it for a pairwise loss. In our case, learning a classifier and using the probabilistic scores to rank the items had almost no negative impact on the prediction performance and is also a valid option. Due to the lack of time we did not create a train/validation split to optimize our hyperparameters. Instead, we tried few hyperparameter configurations that have been good on other data sets and chose the best performing on the validation leaderboard to choose the hyperparameter configurations for our final submission. Hence, we followed the recommendations of meta-learning to save time.

The final results are provided in Table 1.

# 6. RELATED WORK

The modern learning to rank methods started with the pairwise approaches. The ranking support vector machine [5] was the first notable pairwise ranking approach. It reformulated the classical support vector machine for classification to the problem of ranking. In the period from 2005 to 2010 learning to rank for the classical problem of unpersonalized document ranking with respect to a query became a hot topic for researchers. Among the many publications we want to mention the most important. Burges et al. [2] proposed RankNet, a learning to rank model based on neural networks that was optimizing the cross-entropy loss between the difference of the predictions for a pair of documents using stochastic gradient descent. Shortly after, first approaches tried to directly optimize for the listwise losses used in learning to rank as evaluation measures such as NDCG. One of the first works has been done by Burges et al. by deriving a gradient that was supposed to optimize a listwise loss. Hence, LambdaRank [1] can be considered to be the first listwise methods. Only a year later, Cao et al. [3] were the first to propose a true listwise loss and named his model ListNet. Since ensemble models usually outperform single models, first ensemble techniques such as AdaRank [12] based on AdaBoost were proposed.

In 2008 the Netflix Challenge kickstarted a new hot topic: latent factor models [6] for recommender systems. The first research worked close to the problem as described by the challenge. Minimizing the mean squared error based on explicit feedback i.e. user ratings. Soon after, research started to consider the problem of item recommendation as a ranking problem using implicit feedback e.g. clicks. Since 2009 researchers slowly started to apply the pairwise and listwise approaches to the latent factor models for the problem of item recommendation. Rendle et al. [10] proposed BPR that used the idea of RankNet. Further work followed such as LambdaMF [7] and ListRank-MF [11]. Also the boosting methods have been proposed [8]. The generalization of latent factor models to factorization machines [9] led then also to corresponding ranking models [13].

There are two main differences between the research on item recommendation and the classical research on learning to rank. Firstly, classical learning to rank does not consider personalized rankings and secondly, item recommendations are provided without using a search query. The use of both aspect will be advantageous for this challenge.

# 7. CONCLUSIONS

We started working too late on the challenge and hence must admit that a week of work is just not enough time. During this time we were able to provide a stronger baseline but many questions remain unanswered. For future work we would like to distinguish between click and purchase event. Furthermore, we would like to try using latent factor models. The problem with them might be that we have in many cases no customer, a new customer or a customer with only few activities and then these models are not applicable. But maybe a subset of customers exists where these models can be applied and the overall performance can be further improved.

## 8. REFERENCES


[1] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 193–200, 2006.

[2] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, pages 89–96, 2005.

[3] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 129–136, 2007.

[4] 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, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794, 2016.

[5] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

[6] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 426–434, 2008.

[7] G. Lee and S. Lin. LambdaMF: learning nonsmooth ranking functions in matrix factorization using lambda. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 823–828, 2015.

[8] Y. Liu, P. Zhao, A. Sun, and C. Miao. A boosting algorithm for item recommendation with implicit feedback. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1792–1798, 2015.

[9] S. Rendle. Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000, 2010.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461, 2009.

[11] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 269–272, 2010.

[12] J. Xu and H. Li. AdaRank: a boosting algorithm for information retrieval. In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 391–398, 2007.

[13] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang. LambdaFM: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24 - 28, 2016*, pages 1451–1460, 2016.