SESSION-AWARE PRODUCT FILTER RANKING IN E-COMMERCE SEARCH

Hanqing Lu
luhanqin@amazon.com

Xianfeng Tang
xianft@amazon.com

Chen Luo
cheluo@amazon.com

Limeng Cui
culimeng@amazon.com

Rahul Goutam
rgoutam@amazon.com

Haiyang Zhang
hhaiz@amazon.com

Monica Cheng
chengxc@amazon.com

ABSTRACT

Product filters are commonly used by e-commerce websites to refine search results based on attribute values such as price, brand, size, etc. However, existing filter recommendation approaches typically generate filters independently of the user's search query or browsing history. This can lead to suboptimal recommendations that do not account for what the user has already viewed or selected in their current browsing session. In this paper, we propose a session-aware product filter recommendation framework that leverages user's past actions to provide filter recommendations. An offline evaluation demonstrates that our model achieved significant improvement over non-contextual baseline models.

1 INTRODUCTION

Searching and browsing on e-commerce websites can result in many irrelevant product suggestions being displayed due to vague searches. Search filters help screen out irrelevant results and navigate users to their desired products. An appropriate ranking of product filters is required since it is impossible to show the exhaustive list of product filters. Traditional methods such as maintaining a curated list for each query could not scale to all search traffic, and the result sets change with time and so do the product features Vandic et al. (2017). Another type of method leverage the inherent structure between product attribute and product category Vandic et al. (2013; 2017), and they first map a query to a category and then use the category-attribute map as the ranking for product filters. Recently, Ligaj Pradhan & Sin-garayar (2023) proposes a two-stage framework for ranking product filters, where it adopts search-term-level ranking after obtaining the category-to-attribute mappings. However, those methods don't account for users' actions in the search and are suboptimal. Intuitively, customers may already reveal their focus on the product attributes in their past searches and actions. In this paper, we proposed a general framework to leverage those information and build a product filter ranking model.

2 METHODOLOGY

We formulate the problem of session-aware product filter ranking as follows: given a session consists of queries $q_1, ..., q_n$, user applied filters $f_1, ..., f_n$, user clicked products $p_1, ..., p_n$ and customer current query q, output a list of top-k filters $\hat{f}$=$\hat{f}_1, ..., \hat{f}_k$

Our general framework consists of two components: 1) data retrieval stage: retrieve relevant data from the session data; 2) candidate generation stage: extract filters from relevant data and current query; 3) candidate ranking stage: rank the extracted filters.

2.1 DATA RETRIEVAL

Not all the session data are relevant to the current query, and including irrelevant session data could confuse the model and hurt its performance. The data retrieval stage will select the relevant session data and use them for candidate generation. Note that we could re-organize the session data in a query-centric way, where we attribute each clicked product and applied filter to a search query using their timestamps. For example, we know one customer first searched for query q_i, then applied filters f_i, and then clicked on products p_i. We could group q_i, f_i, p_i together and represent this group using q_i. Then the data retrieval problem becomes finding the relevant group to the current query q. In this paper, we adopted a BERT-based Devlin et al. (2018) query-to-category model to map each query to a category, and we deem those group whose q_i has the same category as q as the relevant session data. A detailed example could be found in Appendix.

2.2 CANDIDATE GENERATION

For all the relevant queries, we adopted a BERT-based Devlin et al. (2018) name-entity-recognition (NER) model as described in Zhang et al. (2021) to extract terms that can be mapped to product filters. Those extracted filters are added to the candidate filter pool. Another candidate filter source is past-clicked products of the relevant data. Those products' attribute values are also added to the candidate filter pool. Additionally, applied filters from relevant data groups also serve as candidate filters. Lastly, we also adopt the non-contextual filter ranking algorithm similar to Vandic et al. (2013; 2017); Ligaj Pradhan & Sin-garayar (2023), where we map the current query q to a category and leverage a curated product filter list for each category as the source for candidates.

2.3 CANDIDATE RANKING

We trained a simple Gradient-Boosting Decision Tree (GBDT) model Friedman (2001); Ke et al. (2017) to rank the candidate filters. We designed the following features: 1) time_gap: the timestamp difference between each candidate filter and the current query 2) is_from_filters: whether the candidate is from past applied filters 3) is_from_queries: whether the candidate is extracted from past queries using NER model 4) is_from_products: whether the candidate is extracted from past clicked products 5) non_contextual_score: if the candidate is provided by the non-contextual model, then we use the curated score as the feature value, otherwise the feature value is 0. We adopted a pair-wise ranking loss Cao et al. (2007) that maximizes the margin between the model score of the filter applied right after the current query and the model score of a random candidate filter.

3 EXPERIMENT RESULTS

We evaluated the proposed framework using an e-commerce search dataset. We used recall@10 as the evaluation metric, and we reported the relevant improvement over the non-contextual model in Table 1. We could observe that both models have improvements over the non-contextual model, and the model with relevant data retrieval has more improvement. It's interesting to observe that even the model without relevant data retrieval still outperforms the non-contextual model. We have two possible reasons: 1) ranking model already takes the timestamp of candidate into account, if one candidate is from an irrelevant data group, it's likely to happen a long time ago and the model would demote this candidate in the ranking; 2) ranking model knows the non_contextual_score, and if one candidate is from an irrelevant data group, it's likely to have 0 non_contextual_score and the model would demote it in the ranking stage.

Table 1: Experiment Results

Model	**Relevant Improvement**
Our Model w/o Data Retrieval	+1.82%
Our Model	+2.41%

URM Statement

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

References

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 129–136, New York, NY, USA, 2007. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Bingxin Li Venkata Simhadri Ligaj Pradhan, Le Yu and Jeyaprakash Sin-garayar. Dynamic filter discovery and ranking framework for search and browse experiences in e-commerce. *Proceedings of SIGIR e-commerce workshop*, 2023.

Damir Vandic, Flavius Frasincar, and Uzay Kaymak. Facet selection algorithms for web product search. CIKM '13, pp. 2327–2332, New York, NY, USA, 2013. Association for Computing Machinery.

Damir Vandic, Steven Aanen, Flavius Frasincar, and Uzay Kaymak. Dynamic facet ordering for faceted product search engines. *IEEE Trans. on Knowl. and Data Eng.*, 29(5):1004–1016, may 2017.

Danqing Zhang, Zheng Li, Tianyu Cao, Chen Luo, Tony Wu, Hanqing Lu, Yiwei Song, Bing Yin, Tuo Zhao, and Qiang Yang. Queaco: Borrowing treasures from weakly-labeled behavior data for query attribute value extraction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pp. 4362–4372, New York, NY, USA, 2021.

A Future Works

Our work provides a framework of ranking product filters leveraging the session data and achieved improvement over non-contextual models. In the future, we plan to explore end-to-end method that can directly take the whole session data as input, and output the filters ranking. Compared to the current three-stage methods, we hope the end-to-end method would simplify the system and minimizing the cascading errors.

B Data Retrieval Example

For example, if one customer has the following session data {search query: *nike shoes*, applied filters: *size 10*; search query: *white nike shoes*, applied filters: *price under $50*; search query: *blue t-shirt*", applied filters: *size XL*}, and the same customer searches for *shoes for men*. We would apply the query-to-category model to map current query to category *shoes*. In the session data, *nike shoes* and *white nike shoes* could also be mapped to *shoes* category, but *blue t-shirt* will be mapped to *shirts* category. Therefore, the retrieved relevant session data are {search query: *nike shoes*, applied filters: *size 10*; search query: *white nike shoes*, applied filters: *price under $50*}

C Discussion

The current three-stage method has the following limitations: models in different stages (category mapping model, attribute extracted model, ranking model) are trained independently without global optimization, which leads to errors compounding across stages. For example, a hard query-to-category mapping may not accurately capture multi-category products, or ambiguous queries such as *harry potter* could be both *movie* and *books*, which impacts downstream models. Dividing the problem into separate stages also limits the ability to learn joint representations of queries, categories, and attributes. On the other hand, an end-to-end method can be trained with full data to directly optimize the final ranking goal. The joint model can learn soft category and attribute representations instead of discrete mappings, and it has the capacity to find complex interactions between inputs (query, category, attributes) that may improve filter relevance prediction. End-to-end training also allows error and signal propagation from the final ranking stage back to earlier encoding stages, enabling more globally optimal feature learning.