AttriBERT - Session-based Product Attribute Recommendation with BERT

Akshay Jagatap
ajjagata@amazon.com
Amazon
Bangalore, Karnataka, India

Nikki Gupta
gupnikk@amazon.com
Amazon
Bangalore, Karnataka, India

Sachin Farfade
sfarfade@amazon.com
Amazon
Bangalore, Karnataka, India

Prakash Mandayam Comar
prakasc@amazon.com
Amazon
Bangalore, Karnataka, India

ABSTRACT

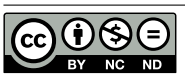
Finding the right product on e-commerce websites with millions of products is a daunting task for a large set of customers. On the search page, product attribute filters a.k.a. "refinements" emerge as a convenient navigational option for customers to narrow down the search results along product attributes of their choice (e.g., Material:Cotton, Color:Black for 'shirt'). However, on mobile devices, refinements are not easily discoverable due to lack of screen space. To improve discoverability, contextually relevant refinements are suggested in-line on search page by refinement recommendation systems. Existing works on refinement recommendations primarily rely on the search context as input, and are trained using aggregated refinement preferences 'explicitly' expressed by customers. These solutions fail to capture 'implicit' preferences expressed during the customer shopping mission through in-session browsing activity. In this paper, we propose a session-based recommendation system (SBRS) which recommends refinements by inferring product attribute preferences of customers based on the sequence of products viewed earlier in the session. For the task of refinement recommendation, we propose a) AttriBERT, a model which extends BERT architecture to learn from the attribute values of products and b) a novel product representation strategy, which represents each product as a dictionary of attribute:value pairs (e.g., RAM Size:64GB). We evaluate our approach on RecSys 2022 Challenge and Amazon e-commerce datasets. Our approach consistently outperforms various state-of-the-art sequence models on the task of session-based refinement recommendation.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Personalization**; **Recommender systems**.

KEYWORDS

BERT, Session-based Recommendations, e-commerce, Search Navigation



SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3594714

ACM Reference Format:
Akshay Jagatap, Nikki Gupta, Sachin Farfade, and Prakash Mandayam Comar. 2023. AttriBERT - Session-based Product Attribute Recommendation with BERT. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3539618.3594714

1 INTRODUCTION

On e-commerce websites, refinements allow customers to narrow down selection by filtering along different product attributes. For example, a customer might initially search for a broad query 'shirt', but then narrow down search results by selecting specific refinements like 'Material:Cotton', 'Color:Black'. We observe that search sessions with refinements applied have higher purchase rate compared to sessions without any refinement. Additionally, applying the right refinement earlier in the shopping mission can help customer find product of their choice quickly and in turn reduce session lengths and session abandonment. While refinements are always visible to customers on desktop, on mobile devices, they are hard to find as they are consolidated behind a drop-down menu due to limited screen space. In such cases, recommendation systems help improve discoverability of refinements [1].

Existing systems formulate refinement recommendation as a search-based ranking task [1, 2], relying primarily on the search query to make recommendations. These systems rely on explicit refinement usage aggregated across customers to recommend relevant refinements, and fail to provide personalized recommendations either based on customer profile or in-session customer behavior. In this work we propose a session-based recommendation system (SBRS) [3] to provide personalized refinement recommendations based on in-session customer behavior [4]. For example, customers searching for "shoes" will be recommended different refinements like Color:Black and Brand:Nike or Color:Blue and Brand:Adidas based on their preferences inferred from recent product views. Existing refinement recommendation systems on the contrary would recommend the same refinements to all the customers.

Current SBRSs like GRU4Rec [5], BERT4Rec [6] are designed to handle the sequence of products as input in order to predict the next product interaction. These systems fail to optimally handle new products added regularly to e-commerce websites. Also, attributes of products viewed by a customer during a shopping mission are likely to be correlated with the attributes of the product viewed next.

Hence, product representation based on product attributes would be more informative for identifying attributes to recommend next. Solutions like KeBERT4Rec [7] propose to handle this problem by using product keywords to improve performance of the recommendation task. Despite leveraging unstructured product information like product keywords for recommendation it is not suitable for handling variable number of structured attributes for product representation. In this paper, we propose a novel approach to represent each product as a dictionary of attribute-value pairs. These attribute-value pairs are comprised of refinement attributes (e.g., 'Material', 'RAM Size') and the corresponding refinement attribute-values (e.g., 'Material:Leather', 'RAM Size:16GB') which are assigned to the product. We also propose the AttriBERT model based on BERT [8] architecture for the task. We represent each product in the sequence as a dictionary of attribute:value pairs (e.g., Material:Cotton) and employ a deep bi-directional self-attention mechanism which captures dependencies along the sequence of products and across the attribute-values. The major contributions of our paper are listed below:

- We propose an SBRS formulation of the refinement recommendation task.
- We implement a novel product representation strategy, which represents each product in a sequence as a dictionary of attribute-value pairs. We adapt BERT to directly learn from such attribute-value representations of products.
- We validate our approach on customer sessions occurring on Amazon and RecSys 2022 Challenge dataset [9].

The rest of the paper is organized as follows. First, we present the problem statement in section 2. Next, section 3 describes the architecture of our proposed AttriBERT model. Further, in section 4 we discuss different modelling strategies and finally experimental results comparing our approach with various state-of-the-art are presented in section 5.

2 PROBLEM STATEMENT

Let $\mathcal{S} = \{s_1, s_2, \cdots, s_{|\mathcal{S}|}\}$ denote the set of sessions with a coherent customer mission[1] and $\mathcal{A} = \{a_1, a_2, \cdots a_{|\mathcal{A}|}\}$ be the set of unique products available in the Amazon catalogue. We define each session s_i as a chronological ordering of products i.e., $s_i = [a_1^i, a_2^i, \cdots, a_t^i, \cdots a_n^i]$ such that $a_j^i \in \mathcal{A}$; $s_i \in \mathcal{S}$; customer interacts with product a_t^i at time step t and n is the total number of product interactions in the session. Each product a can be represented as a dictionary of attribute-value pairs i.e.,

$$a_i = \{k_1^i : v_1^i, k_2^i : v_2^i, \cdots, k_n^i : v_n^i\}$$

where k_j^i denotes a refinement attribute (e.g., 'Color', 'Material') and v_j^i denotes a refinement picker value (e.g., 'Black', 'Leather').

Based on customer session data we propose two ways to model the session-based refinement recommendation below:

(1) Helping the journey (HTJ): given the partially-complete session s, we predict the dictionary of attribute-values which the customer will explore next i.e., we predict the attribute-values of next product (a_{t+1}): $P(a_{t+1} = a | a_1 \cdots a_t)$.

(2) Helping the decision (HTD): given the partially-complete session s, we directly predict the attribute-values of product the customer purchases (a_p) at time step p such that $p > t$: $P(a_p = a | a_1 \cdots a_t)$.

[1]A customer mission is defined as the objective of a customer to purchase a specific type of product (e.g., white running shoes).

3 ATTRI-BERT MODEL

In this section, we introduce our proposed refinement recommendation model called AttriBERT. The model adapts BERT [8] architecture for the task of refinement recommendation.

3.1 Input Representation and Embedding Layer

AttriBERT model takes two sequences as input; 1) attribute-values of each product in the product-sequence; and 2) position index [10] of each attribute-value. To build the input representation, we first map each product in sequence to the corresponding set of attribute-values. Next, each attribute-value is assigned a position index, such that attribute-values of the i^{th} product are assigned position index of i. The position index provides position information of each attribute-value. We supply these two inputs to the AttriBERT model. Further we define a variant of AttriBERT, AttriBERT+, which also incorporates the corresponding sequence of attributes. Overall, input representation for each attribute-value in a given sequence is computed as sum of attribute-value, position and attribute embeddings (only for AttriBERT+). Figure 1 shows schematic of the inputs to the AttriBERT+ model.

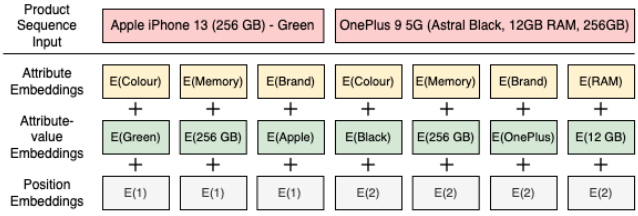


Figure 1: Input to AttriBERT+ is sum of product attribute-value, attribute and position embeddings.

3.2 AttriBERT Architecture

The resulting sequence of embeddings is further processed in encoder layers [11] of the BERT architecture. At each layer, the representation of every element in the sequence is revised iteratively, by exchanging information across all elements at the previous layer in parallel. Similar to [8] after 12 encoder layers, we get the final output $H = \{h_1, h_2, \cdots h_m\}$ for all attribute-values of the input sequence $\{v_1, v_2, \cdots v_m\}$.

Given the output of the final encoder layer, assuming we masked v_t we predict the masked product based on h_t. We apply a feed-forward network with GELU activation to produce an output distribution over target attribute-values:

$$P(v) = \text{softmax}(\text{GELU}(h_t W + b)E^T + b') \quad (1)$$

where W is the projection matrix, b, and b' are bias terms, $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the embedding matrix for the attribute-value set $\mathcal{V}$ and embedding size d.

3.3 Training & Inference Configuration

We train AttriBERT as a Masked Language Model (MLM) [8, 12] for refinement recommendation. Given the partially-complete session of product interactions, the model predicts attribute-values of the masked product. For this purpose, we implement a "whole product attribute" (WPA) masking strategy, similar to whole-word masking [13]. For each product, with probability p we randomly mask (i.e., replace with special mask token "[M]") all the corresponding attributes and attribute-values. With this masked input, we train the model to predict the original attribute-values corresponding to the masked tokens based on its left and right context. In the following example, we show the impact of whole-product masking for the session s:

$$s = [a_1, a_2, a_3]$$

$$a_1 = \{k_1^1 : v_1^1, k_2^1 : v_2^1\}, a_2 = \{k_1^2 : v_1^2, k_2^2 : v_2^2\}, a_3 = \{k_1^3 : v_1^3\}$$

$$[v_1^1, v_2^1, v_1^2, v_2^2, v_1^3] \xrightarrow{\text{train transform}} [v_1^1, v_2^1, [\text{M}], [\text{M}], v_1^3]$$
$$[k_1^1, k_2^1, k_1^2, k_2^2, k_1^3] \xrightarrow{\text{train transform}} [k_1^1, k_2^1, [\text{M}], [\text{M}], k_1^3]$$
$$[1, 1, 2, 2, 3] \xrightarrow{\text{train transform}} [1, 1, 2, 2, 3]$$

Next, we define the loss for each masked input as the negative log-likelihood of the masked targets:

$$\mathcal{L} = \frac{1}{|V_m|} \sum_{v_m \in s_m} -logP(v_m = v_m^* | s') \tag{2}$$

where s' is the masked session, V_m is the set of masked attribute-values, v_m is the masked attribute-value and v_m^* is the original attribute-value which was masked. The objective of training task is to predict the masked attribute-values. Though, while testing, we need to predict future attribute-values which customer will explore. Similar to BERT4Rec [6], to address this mismatch while testing, we append the special token "[M]" to the end of sequence, and then predict the attribute-values based on the final hidden representation of the "[M]". We show the masking strategy while testing on the session s:

$$[v_1^1, v_2^1, v_1^2, v_2^2, v_1^3] \xrightarrow{\text{test transform}} [v_1^1, v_2^1, v_1^2, v_2^2, v_1^3, [M]]$$
$$[k_1^1, k_2^1, k_1^2, k_2^2, k_1^3] \xrightarrow{\text{test transform}} [k_1^1, k_2^1, k_1^2, k_2^2, k_1^3, [M]]$$
$$[1, 1, 2, 2, 3] \xrightarrow{\text{test transform}} [1, 1, 2, 2, 3, 4]$$

The final hidden vector corresponding to mask "[M]" is fed into an output softmax over the set of attribute-values, and we recommend the top-K scoring attribute-values.

4 DISCUSSION

In this section, we describe two modelling strategies for session-based refinement recommendations.

Product ID (PI) - based modelling: One simple approach to recommend session-personalized refinements is to process the product sequence using any of the sequential recommendation techniques like GRU4Rec [5], BERT4Rec [6], etc. Here, the models learns embeddings for individual products such that co-viewed products get similar embedding vectors. Based on these product embeddings, the models predict the refinement attributes of target product (i.e., next product in case of HTJ-task and purchased product in case of HTD-task). A key challenge here is that the model might not generalise well during inference time on the new, unseen products which were not present during the training.

Product Attribute (PA)-based modelling: **PI** models fail to incorporate additional product information. In case of **PA** models, like KeBERT4Rec [7] and AttriBERT, product attribute information is leveraged to improve performance on the recommendation task. Unlike the **PI** models, this approach generalises to previously unseen products as long as the model has learnt embedding representations for attributes of the product.

5 EXPERIMENTS AND RESULTS

5.1 Datasets

In this section, we show the effectiveness of the proposed models via empirical evaluation on Amazon and Dressips [9] datasets.

Amazon datasets consists of customer search sessions, that resulted in a purchase. We restrict the data to purchase sessions, in order to learn refinement recommendations which result in a purchase. Training on non-purchase sessions could result in refinement recommendations which result in session abandonment (non-conversion). For a session $s = \{a_1, a_2, \ldots a_p\}$ with sequence of p products such that a_p is the purchased product, we truncate the browsed product sequence into $(a_1, a_2, ..a_t)$ with $t < p$ and train two separate models, one each for **Amazon-HTJ** and **Amazon-HTD**. For the latter we train the model to predict the refinements of product a_p as output and for the former we predict the refinement corresponding to a_{t+1} as output.

We also evaluate our models on **Dressipi-HTD** [9] dataset, similar to Amazon-HTD dataset.

5.2 Metrics & Baselines

In this section, we describe different baseline approaches. Given the ranked list of attribute-value predictions, we employ three types of metrics for evaluation.

Customer behavior metrics like Precision [14], Recall [14] and Mean Reciprocal Rank (MRR) [15] quantify how well a model captures the attribute-value preferences of customers.

Diversity metrics like Coverage [15] quantify the percentage of unique attribute-values appearing in model predictions; and

Popularity Bias (Pop) [15] quantifies the tendency of model to recommend popular (high frequency) attribute-values.

We compare AttriBERT against the following baselines to demonstrate the effectiveness of our approach.

- **FastText-PI (FT-PI)**: FastText [16] is modelled using the PI-based strategy. The model is trained with the sequence of product IDs as input and attribute-values of the target product as labels.
- **GRU4Rec-PI (G4R-PI)**: GRU4Rec [5] is adapted for the PI-based strategy to recommend attribute-values of the target product given the sequence of product IDs as input.
- **BERT4Rec-PI (B4R-PI)**: BERT4Rec [6] pre-trained on the sequence of product IDs in a session as proposed in the original paper. We fine-tuned the resultant model with the multi-label objective of predicting attribute-values of the target product.

Table 1: Results on a) Amazon datasets denote absolute percentage points differences relative to the average of metric across all models; b) Dressipi dataset denote absolute performance. Bold: Top-3

Metric	Dataset	FT-PI	G4R-PI	B4R-PI	KB4R-PA	FRQ-PA	FT-PA	AttriBERT	AttriBERT+
P@5 (↑)	Ama-HTJ	-2.88%	-0.10%	1.18%	**1.98%**	-1.77%	-2.76%	**3.09%**	**3.26%**
	Ama-HTD	-2.02%	-0.06%	**1.59%**	1.33%	-4.70%	-0.91%	**2.93%**	**3.19%**
	Dre-HTD	75.95%	83.54%	85.66%	**86.34%**	56.60%	78.10%	**89.90%**	**90.35%**
R@5 (↑)	Ama-HTJ	-5.80%	0.12%	3.03%	**4.60%**	-1.38%	-5.12%	**4.51%**	**4.67%**
	Ama-HTD	-3.92%	-0.15%	**3.47%**	3.07%	-4.91%	-2.30%	**3.54%**	**4.28%**
	Dre-HTD	18.31%	19.94%	20.53%	**20.96%**	13.60%	18.60%	**21.72%**	**21.83%**
MRR@5 (↑)	Ama-HTJ	-3.28%	0.81%	**1.50%**	1.47%	-0.70%	-3.08%	**2.30%**	**2.44%**
	Ama-HTD	-1.13%	0.24%	**0.65%**	0.63%	-3.35%	0.05%	**1.46%**	**2.09%**
	Dre-HTD	9.20%	9.72%	9.88%	**9.90%**	6.00%	9.26%	**10.19%**	**10.24%**
Pop@5 (↓)	Ama-HTJ	**7.81%**	**5.83%**	**1.82%**	1.06%	-4.76%	-2.24%	-4.20%	-4.26%
	Ama-HTD	**10.11%**	**8.31%**	**3.76%**	-6.14%	-2.52%	-5.47%	-7.06%	-7.16%
	Dre-HTD	**78.22%**	**73.66%**	68.99%	66.47%	27.60%	**80.50%**	59.66%	59.96%
Cov@5 (↑)	Ama-HTJ	-14.77%	-13.94%	-6.14%	1.59%	**31.54%**	-7.71%	**5.57%**	**5.43%**
	Ama-HTD	-15.10%	-7.22%	-3.21%	-0.31%	**30.65%**	-10.37%	**2.80%**	**2.44%**
	Dre-HTD	0.73%	6.34%	16.79%	24.38%	**42.30%**	0.73%	**37.68%**	**38.51%**

- **Frequency-PA (FRQ-PA)**: This is a simple baseline which recommends the most popular K attribute-values in the window of N-most recently viewed products.
- **FastText-PA (FT-PA)**: FastText is modelled using the PA-based strategy. The model is trained with the input as a "flattened" sequence of attribute-values of products in a session and the attribute-values of the target product as labels.
- **KeBERT4Rec-PA (KB4R-PA)**: Here KeBERT4Rec [7] is adapted for PA-based strategy to learn E2E attribute-value embedding. To represent each product in the sequence, embeddings of attribute-values corresponding to the product are summed up. It is then leveraged by the model to learn patterns of co-occurrence in the training sessions using an MLM training strategy much like BERT4Rec-PI. Similar to BERT4Rec-PI, this model is fine-tuned for predicting the attribute-values of the target product.

5.3 Performance Comparison

In this section, we compare performance of models across the two tasks, summarized in Table 1.

Helping the Journey (HTJ). Firstly, we evaluate both PI- and PA-based models on the Amazon-HTJ dataset.

The Amazon-HTJ dataset has ~1MM unique products, while the same dataset when represented in terms of attribute-values has ~30K unique attribute-values. Hence, the solutions relying on product ID representations learn sparser patterns of co-occurrence as compared to solutions relying directly on product attribute representations. Due to this, PA-based models learn richer representations of attribute-values, hence outperforming the corresponding PI-based counter-parts on the HTJ task. FT-PA and AttriBERT improve over FT-PI and BERT4Rec-PI by 12bps and 208bps in P@5; 68bps and 164bps in R@5; 20bps and 94bps in MRR@5 respectively on the HTJ task. We also note that, the three PI-based approaches have highest Pop@5 and among the lowest Cov@5 indicating the tendency to recommend a small set of popular attribute-values.

Further, among the PA-models, we compare AttriBERT against KeBERT4Rec. In our implementation of KeBERT4Rec, we learn attribute-value embeddings which are summed-up to represent each product. While, in case of AttriBERT, each attribute-value is represented separately in the session (Figure 1). We observe a performance improvement in customer behavior metrics 111bps and 83bps in P@5 and MRR@5 while maintaining a higher Cov@5 (+398bps) and lower Pop@5 (-526bps). Validating the effectiveness of our attribute-value representation strategy in AttriBERT.

Finally, AttriBERT+ is the best performing model across all the customer behavior metrics for HTJ task. We observe an improvements of 17bps in P@5, 16bps in R@5, 14bps in MRR@5 over AttriBERT. This improvement can be ascribed to the inclusion of attribute embeddings in addition to the attribute-value embeddings.

Helping the Decision (HTD). For HTD task, we evaluated each model on the 2 datasets, namely Amazon-HTD and Dressipi-HTD (Table 1). HTD evaluates the ability of model to predict the attribute-values of the purchased product occurring at the end of session. We have observations similar to the HTJ task, AttriBERT consistently out-performs both BERT4Rec-PI and KeBERT4Rec-PA in customer behavior and diversity metrics across the 2 datasets. While, AttriBERT+ further improves over AttriBERT and consistently outperforms all models across customer behavior metrics.

6 CONCLUSION

In this paper, we introduce AttriBERT for refinement recommendation. AttriBERT learns to recommend refinements directly from the product attributes as opposed to learning from the product IDs. We compare our model with sequence models of different complexities and demonstrate effectiveness of the dictionary representation combined with bi-directional architecture on our problem. On reviewing the recommendations, we determine that HTJ approach is more suitable in the beginning of the session, while HTD approach is suitable after a few session interactions. Our improved AttriBERT+ model out-performs all models across both tasks. As next step, we plan to explore strategies to combine models trained for HTJ and HTD tasks in order to optimally guide the customers in completing their shopping mission.

7 AUTHOR BIO

Akshay Jagatap is an Applied Scientist at Amazon, he works on NLP and personalization-related problems focused on improving search. He develops machine learning solutions to improve the customers' experience on Amazon. He received his BE CSE and MSc Physics from BITS-Pilani.

Nikki Gupta is an Applied Scientist at Amazon. She received her BE CSE from BITS-Pilani. She works on building machine learning systems to improve customer user experience on Amazon. She develops NLP and sequence modelling solutions in the personalization and recommendation domain.

Sachin Farfade is working as a Principal Applied Scientist in Amazon, India. He works on developing NLP and CV solutions to improve and build new customer experiences on Amazon.in. Before joining Amazon, Sachin worked at Yahoo Labs, where he developed computer vision solutions for face detection, face recognition and large-scale image classification using deep learning. He received his ME degree from Indian Institute of Science, Bangalore in 2005.

Prakash Mandayam Comar is a Principal Applied Scientist at Amazon. He received his PhD from Michigan State University. He works on implementing machine learning algorithms for ranking, relevance, autocomplete-related problems at Amazon. He also specializes in field of forecasting, personalised recommendation and user browse modelling.

8 COMPANY DESCRIPTION

Amazon.com, Inc. is a multinational technology company focusing on e-commerce, cloud computing, online advertising, digital streaming, and artificial intelligence. Founded in 1994, Amazon has grown to become the world's largest online retailer and marketplace, cloud computing service through AWS, live-streaming service through Twitch, and Internet company as measured by revenue and market share. We cover search and information retrieval scenarios primarily focused in the field of e-commerce. Members of Amazon, appear frequently at information retrieval and knowledge discovery venues and have contributed significantly to these technologies.

REFERENCES

[1] Jacek Golebiowski, Felice Antonio Merra, Ziawasch Abedjan, and Felix Biessmann. Search filter ranking with language-aware label embeddings. In *Companion Proceedings of the Web Conference 2022*, WWW '22, pages 121–125. Association for Computing Machinery, 2022.

[2] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *ArXiv*, abs/1705.01509, 2017.

[3] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4), jul 2018.

[4] Shoujin Wang, Longbing Cao, Yan Wang, and Quan Z. Sheng. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)*, 54:1 – 38, 2019.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2015.

[6] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[7] Elisabeth Fischer, Daniel Zoller, Alexander Dallmann, and Andreas Hotho. Integrating keywords into bert4rec for sequential recommendation. In *Deutsche Jahrestagung für Künstliche Intelligenz*. Springer-Verlag, 2020.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[9] Nick Landia, Rachael Mcalister, Donna North, Saikishore Kalloori, Abhishek Srivastava, and Bruce Ferwerda. Recsys challenge 2022 dataset: Dressipi 1m fashion sessions. Association for Computing Machinery, 2022.

[10] Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. Improve transformer models with better relative position embeddings. In *Findings*, 2020.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[12] W. L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism Quarterly*, 30:415–433, 1953.

[13] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. Pre-training with whole word masking for chinese BERT. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514, 2019.

[14] David M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *ArXiv*, abs/2010.16061, 2020.

[15] Malte Ludewig and D. Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28:331–390, 2018.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2016.

APPENDIX

9 DETAILS ON MODEL TRAINING

FastText-PI (FT-PI): We train a supervised FastText model with the sequence of viewed product IDs as input and the target product ID as the label. The attribute values of the top-k predicted products in decreasing order of frequency are recommended as the refinement values. We auto-tune the model for 30 trials to optimise the F1-score on the validation set.

GRU4Rec-PI: GRU4Rec is a session-based recommendation model which models the browsing session of the customer to predict the next product interaction. For the HTJ-task, we use session-parallel mini batches where the first interaction of each session forms the first mini batch with the second interaction as the target, the second interaction forms the second mini batch and so on. Each time a session ends, the GRU hidden states are reset and a new session sequence is added. For the HTD-task, the entire sequence of customer interaction is passed through the multi-layer GRU and the hidden state of the last layer is fed as an input to the feed forward layer to predict the target or purchased product. The models are trained using the BPR loss such that the negative samples are sampled randomly from each mini batch. Similar to FT-PI, we use the attribute set of top-k predicted products as refinement recommendations.

Frequency-PA: It is the simplest baseline model which ranks refinement based on frequency of occurrence. Given a sequence of n products in a session $s = [a_1, a_2, \cdots a_n]$, we consider the window of r-most recent interactions $s_{\text{window}} = [a_{n-r}, a_{n-r+1}, \cdots a_n]$. We map each product in the window s_{window} to corresponding attribute-values to obtain the list of attribute-values interacted with in the r-most recent interactions. From the resulting list, $s_{window} = [k_1^{a_{n-r}} : v_1^{a_{n-r}}, k_1^{a_{n-r}} : v_2^{a_{n-r}}, \cdots, k_m^{a_n} : v_o^{a_n}]$ we compute the frequency of occurrence of each attribute-value and recommend the attribute-values in the descending order of count.

FastText-PA: In FT-PA, given the sequence of products, we first map each product to the corresponding attribute-values to create a 'flattened' sequence of attribute-values. The attribute-values are ordered in the flattened sequence based on their frequency of occurrence in our catalogue. The FastText model is then trained

directly to predict the attribute-values of the target product, given the sequence of attribute-values as input. Since the target product could be mapped to multiple attribute-values, we train a multi-label FastText model and provide the top-K predicted attribute-values as recommendations.

KeBERT4Rec-PA: KeBERT4Rec builds on BERT4Rec model by integrating product keywords in the sequential recommendation task. In our implementation, the product attribute-values are encoded into a unified embedding k_t such that the input to the BERT model h_t is given as: $h_t = e_t + p_t + k_t$ where e_t is the product embedding and p_t is the positional embedding. To generate the unified embedding, we represent the product attribute-values as a multi-hot vector which is scaled to the embedding size d of the model using a linear layer. The model is trained using the MLM strategy by randomly masking items and their respective attribute-values. We fine-tune the model to directly predict the attribute-values of the masked product. Similar to FT-PA, we train a multi-label model with cross-entropy loss to predict the top-k attribute-values as recommendations.

10 QUALITATIVE ANALYSIS

The ability of transformers architectures to attend to attribute-values relevant to the customer mission helps AttriBERT/AttriBERT+ outperform other models on both HTJ and HTD tasks. To illustrate the strength of the model, we analyse the attention mechanism in the presence of relevant and irrelevant attributes. Given a session with the customer mission as "face wash", we plot the average heatmap for one of the attention matrices (see Fig. **??**A). Next we introduce noise, by artificially replacing few relevant attribute-values with non-relevant ones. For e.g., we replaced few "Brand:Lotus Herbals" tokens with "Color:Blacks" tokens (i.e., attribute-values irrelevant for the "face wash" customer mission). The last column in the two attention matrices indicates the contribution of each attribute-value in predicting the masked attribute-value. We observe that the model gives a lower weightage (red squares in Figure 2B) to the irrelevant attribute-values in order to predict the label ([MASK]).

11 EVALUATION METRICS

In this Section, we describe the evaluation procedure and the key metrics that will be used to evaluate our models:

- Precision@K: The fraction of attribute-values which are in the target set (T), out of the top-K ranked attribute-values (R).

$$P@K = \frac{|T \cap R|}{K}$$

- Recall@K: The fraction of attribute-values which are captured in the top-K ranked attribute-values (R) out of the target set (T).

$$R@K = \frac{|T \cap R|}{|T|}$$

- Mean Reciprocal Rank@K: In the ranked list of top-K attribute-values, let the rank of each attribute-value belonging to target set (T) be $i_T = \{i_{t_1}, i_{t_2}, \cdots, i_{t_{|T|}}\}$ such that $t \in T$. If $t_j \in R$, then $i_{t_j} = \infty$.

$$\text{MRR@K} = \frac{1}{|T|} \sum_{j=1}^{|T|} \frac{1}{i_{t_j}}$$

- Popularity Bias@K: High accuracy values can, depending on the measurement method, correlate with the tendency of an algorithm to recommend mostly popular items [ref]. To assess this tendency, we report the average popularity score for the attribute-values of the top-k recommendations $r \in R$. This average score is the mean of the individual popularity scores of each recommended refinement picker. We compute these scores based on the training set by counting how often each item appears in the training sessions and by then applying min-max normalization to obtain a score between 0 and 1.

$$PopBias@K = \frac{1}{K} \sum_{j=1}^{K} \text{NormalizedCount}(r_i)$$

- Coverage@K: We report the fraction of the set of unique attribute-values (P) captured in the set of top-K (R) across each of the N samples in the evaluation set.

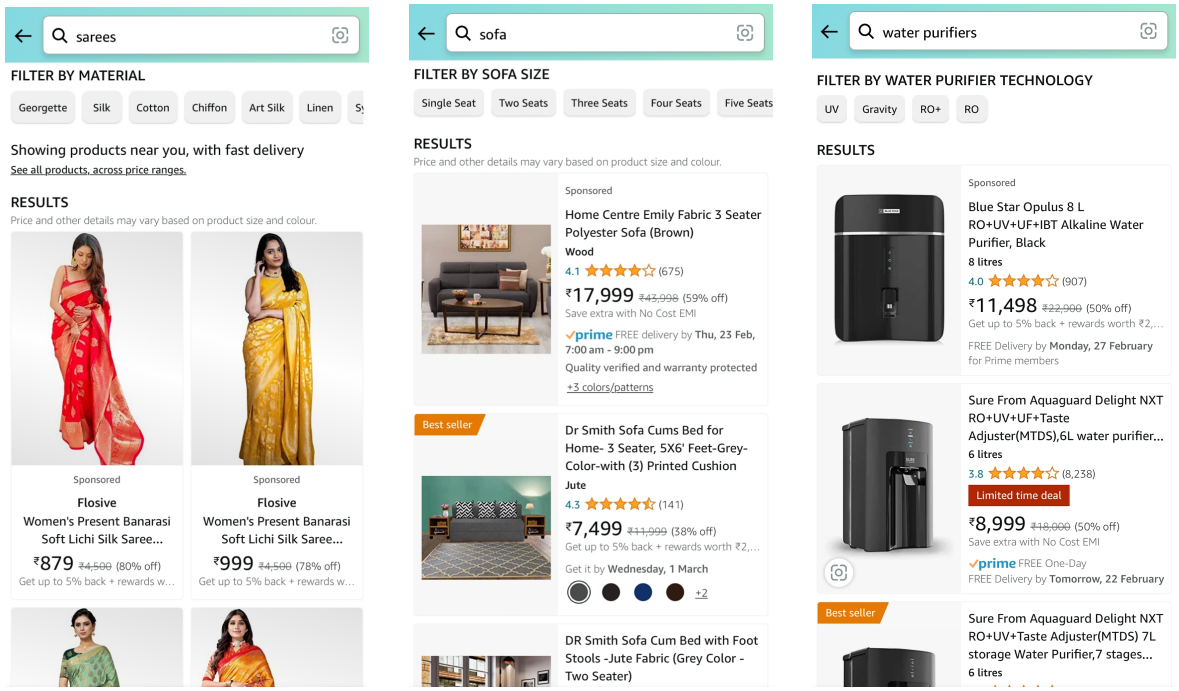
$$C@K = \frac{|\bigcup_{i=1}^{N} R_i|}{|P|}$$

12 EXAMPLE REFINEMENT RECOMMENDATION SYSTEM

Figure 2 shows customer experience of refinement recommendation system for search keywords 'sarees', 'sofa' and 'water purifier'. The system recommends 'material' as a top refinement with 'Georgette', 'Silk', 'Cotton', 'Chiffon', 'Art Silk' ans 'Linen' as top picker values in that ranked order. Similarly, it recommends refinement 'sofa size' for 'sofa' and 'water purifier technology' for 'water purifier'.

13 DRESSIPI DATASET

Dressipi dataset [9] was released as part of RecSys Challenge 2022, which provides the sequence of product interactions within a customer session as well as the metadata associated with the corresponding products. The original task in the challenge was to predict the product purchased at the end of a customer session, given the partially-incomplete sequence of product interactions. We modify the problem as a HTD task, given the sequence of products in a session, we re-define the task as identifying the attributes-values of the product purchased at end of the session.



(a) sarees (b) sofa (c) water purifier

Figure 2: Examples of refinement recommendation system on search page