Comparison of Transformer-Based Sequential Product Recommendation Models for the Coveo Data Challenge

Elisabeth Fischer
Daniel Zoller
Andreas Hotho
elisabeth.fischer@informatik.uni-wuerzburg.de
zoller@informatik.uni-wuerzburg.de
hotho@informatik.uni-wuerzburg.de
University of Würzburg
Data Science Chair
Würzburg, Germany

ABSTRACT

Providing good recommendations to keep users engaged and predict their behavior are crucial components in today's e-commerce businesses. To model user interests, various data sources can be utilized like user interactions including search behavior, product descriptions, but also product interactions like add-to-cart events or purchases. The SIGIR eCom - Coveo Data Challenge provides a new dataset containing such data points and calls for systems predicting the next product interaction as one of their challenge tasks. In this paper we report our approaches and results for the recommendation task of the challenge. We unify the various data sources from the Coveo Dataset to use it with sequential recommendation models and experiment with two datasets: One that includes all interactions and one that only consists of product interactions. For both datasets we train the transformer-based next-item recommender models SASRec and BERT4Rec. To integrate the available categorical metadata, we adapt KeBERT4Rec, which allows the addition of keyword descriptions, and experiment with two variants.

KEYWORDS

transformer models, next click prediction, sequential recommendation, categorical item information

ACM Reference Format:
Elisabeth Fischer, Daniel Zoller, and Andreas Hotho. 2021. Comparison of Transformer-Based Sequential Product Recommendation Models for the Coveo Data Challenge. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'21).* ACM, New York, NY, USA, 5 pages.

1 INTRODUCTION

Over the last decade online shopping has become widespread with more and more e-commerce businesses opening their doors. To keep users engaged, helping them navigating the growing range of products, leveraging recommender systems has become inevitable. Being able to recommend items, but also to predict purchases, cart abandonment or similar events is of huge interest for the industry, as shown by the announcement of the one million dollar challenge by Netflix in 2007 [1]. The SIGIR 2021 E-Commerce Workshop hosting the Coveo Data Challenge [11] targets two of these research challenges: The prediction of the next product interaction and the prediction of cart abandonments. The Coveo Data Challenge also introduces a new public e-Commerce dataset containing various browsing and search events as well as rich product metadata. In this paper we focus on predicting the product interaction with transformer-based models for sequential next-item recommendation. Furthermore, we also include categorical metadata in our models with adapted transformer models, that can encode additional metadata to improve recommendations for the user. We experiment with SASRec [6] and BERT4Rec [3]. To integrate some of the available products and interaction meta information in our models, we also include KeBERT4Rec [4] in our experiments.

The rest of the paper is structured as follows: In Section 2 we introduce the related work. Then, we describe the challenge and problem setting in Section 3. Section 4 attends the model descriptions and Section 5 our experimental setup. In Section 6 we present our results and our conclusion in Section 7.

2 RELATED WORK

The task of next-item prediction has led to many approaches using neural networks including CNNs [12], RNNs [5], recurrent CNNs [16] and self-attention networks [6]. Current state-of-the-art models for sequential item recommendation mostly rely on Transformer Networks [14]. In [6] the authors propose SASRec, which makes use of self-attention [14]. The Personalized Transformer introduced by [15] uses self-attention and fully connected layers and adds user embeddings. Similar to SASRec a model introduced by [10] adapts the BERT [3] model to the sequential recommendation task, called BERT4Rec. In [4] an extension of BERT4Rec is proposed, integrating categorical metadata of items into the model. NOVABert [8] by Liu et al. is another BERT based model, but it incorporates meta information via an attention-based information fusion, while KeBERT4Rec concatenates the embeddings of the meta information to the item embeddings. Bianchi et al. [2] create product embeddings and predict the next item with a setup like BERT4Rec. Overall, transformer models have shown to successfully learn user behavior for recommendation tasks.

3 COVEO DATA CHALLENGE

In this section we present an overview of the challenge data and define the problem setting of the challenge's recommendation task.

3.1 Dataset Overview

The Coveo Challenge [11] dataset consists of user interactions sampled from a mid-size unnamed shop. These interactions include besides product page visits also interactions like search requests. Moreover, the dataset contains further information about these interactions, for example, for search requests the vectorized search terms with their corresponding clicked products are provided. Catalog metadata for the products of the shop including categorical, descriptive and visual features for products is also available. In this paper we only leverage the categorical meta information available, that is the products' category, price bucket and the associated product action (e.g., added to cart) and omit the vector information about product descriptions and images.

3.2 Problem Setting

In this paper we are only interested in session-based recommendation task of the Coveo Data Challenge. The task addresses the problem of predicting the next product interactions, given the previous interactions of a user. We denote the set of sessions with $\mathcal{S} = \{S_1, S_2, \ldots, S_{|\mathcal{S}|}\}$. Then, each session $S_i \in \mathcal{S}$ can be expressed as a sequence of interactions $S_i = \{e_1^i, e_2^i \ldots, e_{n_i}^i\}$, where $e_t^i \in \mathcal{E}$ is the interaction at relative time step t and $\mathcal{E}$ denotes the set of all interactions. Each interaction is associated with (1) a shop URL ($u \in \mathcal{U}$), and maybe (2) a product ($p \in \mathcal{P}$). Moreover, for each interaction the dataset provides categorical metadata in the form of (1) the action applied to the product ($a \in \mathcal{A}$), (2) the product category ($c \in C$), or (3) the product price bucket ($b \in \mathcal{B}$), where $\mathcal{U}$ denotes the set of all URLs, $\mathcal{P}$ the set of all products, $\mathcal{A}$ all possible actions regarding a product, C the set of all product categories and $\mathcal{B}$ the set of all product price buckets. Since, we will also consider sessions, where some interactions do not target product related sites, all metadata sets contain a NONE metadata value.

4 MODELS

We want to explore the capabilities of Transformer-based sequential next-item recommendation models for the provided challenge task. Therefore, we utilize SASRec, BERT4Rec and KeBERT4Rec, which we explain in this section. The overall structure of all three networks is shown in Figure 1. All models embed the items in the sequence, and utilize a positional embedding to encode the position of the item within the sequence. The embeddings are summed and then fed into a Transformer network, which consists of L layers of Transformer blocks (see [14] for a description of these blocks). Finally, a projection layer scales the output of the Transformer network to the item space to create prediction scores for each item.

4.1 SASRec

SASRec [6] was introduced by Kang and McAuley for the task of sequential recommendation by learning to predict the next interaction given the previous interactions. For that a combined item and positional embedding is created from the input sequence and

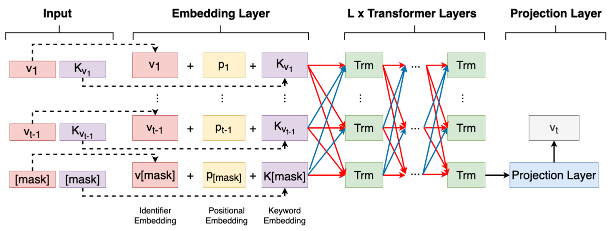

Figure 1: Overview of SASRec, BERT4Rec and KeBERT4Rec (adapted from [4]). All three models embed the item ids of the sequence and add a utility position embedding. While BERT4Rec and SASRec only rely on the item and position embedding as input, KeBERT4Rec also utilizes an embedding for the keywords. In all models, the embedding layers are combined via addition and fed into the transformer layers. The state of the last transformer is used to represent the sequence and to predict the next item. SASRec uses only the unidirectional forward connections (red arrows), while masking is only applied by BERT4Rec and KeBERT4Rec.

processed by a unidirectional Transformer network. To reduce the model size and prevent overfitting, the SASRec model uses the transposed input item embedding to scale the sequence representation of the Transformer network to the item space to make predictions. The model uses an adapted binary cross-entropy loss for training.

4.2 BERT4Rec

Following the BERT model [3] the BERT4Rec model replaces the unidirectional layers of SASRec with bidirectional connected layers and uses the Cloze [13] masking task for training the recommendation task, that is, the model must predict randomly masked items of a sequence. In contrast to SASRec, BERT4Rec first leverages the output of the last transformer layer and pushes it through a feed-forward network in the projection layer before it also utilizes the transposed input item embedding to get a probability for each item. In this paper, we use a variant of the projection layer, which is introduced in [4], and uses a second linear layer instead of the transposed item embedding to scale the output to the item space. The model is trained using a cross-entropy loss.

4.3 KeBERT4Rec

To encode additional categorical metadata provided by the challenge with the BERT4Rec model, we use KeBERT4Rec [4], that encodes keyword descriptions of items into the BERT4Rec model. It is built upon BERT4Rec and adds the possibility to add a categorical attribute for items to the model. The model embeds keywords and then adds the representation to the item and positional embedding. As in BERT4Rec the embedded sequence is pushed through layers of transformers and a linear layer predicts the item. We investigate two variants to include multiple categorical metadata of an interaction to the model.

4.3.1 Single Feed Forward (KB_{single}). We one-hot encode each categorical metadata of the interaction and concatenate all one-hot vectors of each attribute into a single vector. Using a linear feed-forward layer we scale this vector to the embedding size of the item and the positional embedding and add them to create the input for the Transformer network.

4.3.2 Multi Feed Forward (KB_{multi}). Because each used meta data is a categorical one, we adapt KeBERT4Rec to model any number of categorical meta data. Therefore, we embed each categorical meta data in the challenge dataset using a separate feed-forward layer to the embedding size of the item and position embedding for each embedding. The input of the transformer network is then the sum of all metadata embeddings and the item and position embedding.

5 EXPERIMENTAL SETUP

In this section we describe how we prepare our datasets and the setup for evaluation. We also report the metrics used for evaluation, the hyperparameters for the model and the training setup.[1]

5.1 Data Preparation

We combine all browsing interactions, search query information and categorical meta data into a single sequence dataset containing interactions ordered by the provided timestamp for each session. From the search meta information we extract the list of clicked products and add each of them as another interaction according to the click order between the search request and the following request into the session. To these added interactions, we assign the clicked product and introduce and assign a new product interaction type *clicked* to the already given product interaction types *detail, add, purchase* and *remove*. Besides the interaction type $a \in \mathcal{A}$, we also add the category $c \in C$ and price bucket $b \in \mathcal{B}$ of each product for each interaction if the information is available. All introduced next-item recommendation models rely on an item set $\mathcal{I}$ that must be embedded. This item set is constructed using two approaches resulting in two datasets. First, we consider all interactions of a sequence and use the identifier of the product if the user interacted with a product, otherwise the URL of the page viewed. We refer to this dataset as the *Full Dataset*. Moreover, we experiment with a dataset, named *Product Dataset*, where we remove all interactions that do not have an associated product (i.e., where the product interaction type is not set). Here we can use the product identifiers as the item identifiers for the models. Further, we remove all sessions with only one interaction as well as all items with only one interaction from both datasets. The statistics for both datasets are shown in Table 1. Figure 2 shows a frequency distribution of items seen in both datasets. Both distributions show a high number of infrequent items and a long tail of more frequent items, but the *Full Dataset* contains about six times more items than the *Product Dataset*, and most of them are rare.

5.2 Evaluation Setting

For all experiments we split the two constructed datasets randomly in 80% train, 10% test and 10% validation data. We evaluate all our models offline on the test data. Also, we report the scores achieved on the closed challenge test data set. Next product interactions are generated by providing the previous session to the model and returning the top 20 ranked products (for the *Full Dataset* we only consider the top product recommendations, and ignore recommendations for non-product interactions). For the challenge submission, in the case that our models cannot provided recommendations for the given session, because the session only contains non-product interactions (e.g., only search interactions), we recommend the most popular products. We found that out of the 332247 sessions in the challenge test set for the *Full Dataset* there are 487 sessions without direct product interaction and 227755 sessions for the *Product Dataset*.

[1] Our code for preprocessing the dataset, training and evaluation the models is available at https://professor-x.de/sigir_challenge21/recommender.zip.

Table 1: Statistics of the *Full Dataset* and *Product Dataset* after pre-processing. With $\overline{S}$ we denote the average session length and $\mathcal{I}$ the considered item set for the dataset.

Dataset	$\|\mathcal{S}\|$	$\|\mathcal{I}\|$	$\overline{S}$	$\|\mathcal{A}\|$	$\|C\|$	$\|\mathcal{B}\|$
Full	4,004,558	257,308	8.82	6	172	11
Product	1,652,985	40,156	5.52	5		

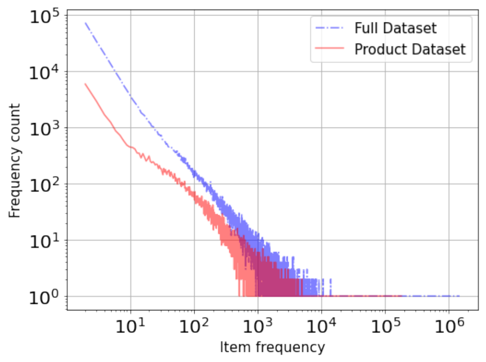

Figure 2: Frequency distributions of the item frequency for Full Dataset and Product Dataset

5.3 Metrics

We report the official metrics of the challenge for our models, the *Mean Reciprocal Rank* (MRR) of the immediate next product, the *F1-Score* for predicting all subsequent product interactions in a session and the *Coverage* (Cov) and *Popularity Bias* (PB). F1-Score, Cov and PB are evaluated based on a cut-off value of 20. For the full description of the metrics we refer to challenge description in [11]. Also, we report common metrics for the next product recommendation task, namely the *Normalized Discounted Cumulative Gain* (NDCG) and the *Recall* for different cut-offs k in addition to the MRR.

5.4 Model Hyperparameters and Training

An extensive parameter study was not possible due to the short time of the challenge, so we set our parameters as follows: We use 2 transformer layers with 8 heads. The dropout [9] rate is set to 0.1 for all dropouts in the transformer model. The maximum sequence length is set to 50, so only about 2% of the sessions are cut. The hidden size of the transformers is set to 256 for models trained on the *Full Dataset* and 128 on the *Product Dataset*. Batch size and the maximal number of epochs is set to 32 and 20 for the *Full Dataset*

Table 2: Results of the sequential recommendation Transformer models SASRec, BERT4Rec and the two considered variants of KeBERT4Rec trained on the two dataset, evaluated on the offline test set and the Coveo Data Challenge test set (Chall.). Best values for each test set and metric are in bold.

(a) Results for the Full Dataset.

	Metric	SASRec	BERT4Rec	KB_{multi}	KB_{single}
Offline	Recall@1	0.081	0.762	**0.770**	0.767
	Recall@10	0.835	0.906	0.906	**0.907**
	Recall@20	0.883	0.929	0.929	**0.929**
	NDCG@10	0.479	0.835	**0.838**	0.837
	NDCG@20	0.492	0.841	**0.844**	0.843
	MRR	0.368	0.815	**0.819**	0.818
Chall.	MRR	**0.110**	0.089	0.052	0.000
	F1@20	**0.058**	0.042	0.024	0.000
	Cov@20	0.369	0.380	**0.474**	0.000

(b) Results for the Product Dataset.

	Metric	SASRec	BERT4Rec	KB_{multi}	KB_{single}
Offline	Recall@1	0.277	0.378	0.382	**0.385**
	Recall@10	0.661	0.711	**0.713**	**0.713**
	Recall@20	0.729	0.769	0.770	**0.771**
	NDCG@10	0.468	0.542	0.545	**0.546**
	NDCG@20	0.485	0.557	0.559	**0.561**
	MRR	0.415	0.495	0.498	**0.500**
Chall.	MRR	0.093	**0.136**	0.134	0.135
	F1@20	0.050	**0.054**	0.053	0.053
	Cov@20	**0.416**	0.404	0.408	0.413

and 128 and 50 for the *Product Dataset*. All networks are optimized using Adam [7] with a learning rate of $5 \cdot 10^{-4}$, beta1 of 0.90, beta2 of 0.998 and a weight decay of 0.01.

6 EXPERIMENTS

In this section we present the results on the offline test set and on the challenge test data obtained by training the Transformer models on two different datasets and discuss the obtained results.

6.1 Full Dataset Results

As we can observe from Table 2a, we can achieve a Recall@10 of over 0.8 on our local test set using all Transformer models, when training the sequential recommendation models on all types of interactions. Only the Recall@1 is low for SASRec with only 0.08. BERT4Rec outperforms SASRec on each metric by several percent, at least about 5% for the Recall and over 40% regarding the MRR. Both KeBERT4Rec models only show a slight increase compared to the BERT4Rec model, although they have additional categorical meta data about the products available. The difference between the variants is ambiguous, with the KB_{multi} performing better or the same for most metrics like the Recall@1, but performing worse for Recall@10. For the challenge test data the MRR drops to a fracture of the measured one on the local test set, especially for the BERT4Rec based models. Moreover, all BERT4Rec models are in this test set outperformed by SASRec, which achieves an MRR of 0.110. Also, SASRec scores the best F1-Score with about 0.06. However, KB_{multi} achieves with about 0.5 the better Coverage. All PB@20 values are (rounded) zero for all models on all test sets, therefore we do not report them in the table. The difference between the MRR on the local and the challenge test data for the BERT4Rec models is significant and needs further investigation as well as why the KB_{single} variant is not able to make any useful predictions.

6.2 Product Dataset Results

In our next experiment we use the *Product Dataset* to limit the training of the models on actual product interactions. The results are shown in Table 2b. All models are able to learn the user behavior, with Recall@20 over 0.7 for all models. The BERT4Rec and KeBERT4Rec models perform better than the SASRec regarding all metrics, and achieve at least 4% higher Recall values. The difference in performance gets even clearer when looking at the ranking metrics NDCG and MRR. With the KeBERT4Rec models, we see again only small improvements in comparison the base BERT4Rec model. KB_{single} seems to work a bit better here, but the improvements are still very small, with a gain in MRR of only 0.2. On the challenge test data, we see the MRR drop in comparison to our local test data again, but this was partly to be expected as we cover only about two thirds (see Section 5.2). The MRR drops by more than half from 0.49 to 0.13 for BERT4Rec, but overall it is performing the best, even when compared to results of SASRec on the *Full Dataset*. SASRec achieves only a MRR of 0.09 which is worse compared to the scored obtained by the models trained using the *Full Dataset*. Both KeBERT4Rec variants are not able to improve the results on the challenge test set and perform even slightly worse by up to 0.2% compared to BERT4Rec. The number of epochs was not limiting for the experiments on this dataset, as the validation loss showed no further improvement. Overall, when considering our offline test sets, we can conclude from the obtained results, that adding other interactions, that do not target products, can improve the next product recommendation setting when using BERT4Rec models. In contrast to that the results of the challenge test set let us draw the opposite conclusion. Also, it seems, that SASRec handles these additional interactions differently compared to the BERT4Rec models, so we should investigate this in future work.

7 CONCLUSION

In this work, we presented our experiments with transformer-based sequential models for the Coveo Data Challenge. We present results for two datasets, one including non-product interactions and one excluding them. We find that transformer-based models like BERT4Rec and SasRec are able to learn user behavior, but the challenge test data shows that treating events and products the same might mislead the models. Adapting the training and models to differ between events and products could help use the full potential of transformer models. The inclusion of event meta data did not bring the expected gain, so further investigation and potential parameter tuning would be needed. As we limited our approach to categorical data, there is also the other meta data left to explore.

REFERENCES

[1] James Bennett, Stan Lanning, and Netflix Netflix. 2007. The Netflix Prize. In *In KDD Cup and Workshop in conjunction with KDD*.

[2] F. Bianchi, Bingqing Yu, and Jacopo Tagliabue. 2020. BERT Goes Shopping: Comparing Distributional Models for Product Representations. *ArXiv* abs/2012.09807 (2020).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[4] Elisabeth Fischer, Daniel Zoller, Alexander Dallmann, and Andreas Hotho. 2020. Integrating Keywords into BERT4Rec for Sequential Recommendation. In *KI 2020: Advances in Artificial Intelligence*.

[5] Balázs Hidasi, Alexandros Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks.

[6] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[7] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[8] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation. http://arxiv.org/abs/2103.03578 cite arxiv:2103.03578Comment: Accepted at AAAI 2021.

[9] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958. http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[10] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 1441–1450. https://doi.org/10.1145/3357384.3357895

[11] Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Bingqing Yu, Patrick John Chia, Federico Bianchi, and Giovanni Cassani. 2021. SIGIR 2021 E-Commerce Workshop Data Challenge. arXiv:2104.09423 [cs.IR]

[12] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM'18*. ACM Press. https://doi.org/10.1145/3159652.3159656

[13] Wilson L. Taylor. 1953. "Cloze procedure": a new tool for measuring readability. *Journalism & Mass Communication Quarterly* 30 (1953), 415–433.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[15] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Fourteenth ACM Conference on Recommender Systems*. 328–337.

[16] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S.Sheng S.Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent Convolutional Neural Network for Sequential Recommendation. In *The World Wide Web Conference on - WWW'19*. ACM Press. https://doi.org/10.1145/3308558.3313408