ABSTRACT

Much of the recent progress in sequential and session-based recommendation has been driven by improvements in model architecture and pretraining techniques originating in the field of Natural Language Processing. Transformer architectures in particular have facilitated building higher-capacity models and provided data augmentation and training techniques which demonstrably improve the effectiveness of sequential recommendation. But with a thousandfold more research going on in NLP, the application of transformers for recommendation understandably lags behind. To remedy this we introduce Transformers4Rec, an open-source library built upon HuggingFace’s Transformers library with a similar goal of opening up the advances of NLP based Transformers to the recommender system community and making these advancements immediately accessible for the tasks of sequential and session-based recommendation. Like its core dependency, Transformers4Rec is designed to be extensible by researchers, simple for practitioners, and fast and robust in industrial deployments.

In order to demonstrate the usefulness of the library and the applicability of Transformer architectures in next-click prediction for user sessions, where sequence lengths are much shorter than those commonly found in NLP, we have leveraged Transformers4Rec to win two recent session-based recommendation competitions. In addition, we present in this paper the first comprehensive empirical analysis comparing many Transformer architectures and training approaches for the task of session-based recommendation. We demonstrate that the best Transformer architectures have superior performance across two e-commerce datasets while performing similarly to the baselines on two news datasets. We further evaluate in isolation the effectiveness of the different training techniques used in causal language modeling, masked language modeling, permutation language modeling and replacement token detection for a single Transformer architecture, XLNet. We establish that training XLNet with replacement token detection performs well across all datasets. Finally, we explore techniques to include side information such as item and user context features in order to establish best practices and show that the inclusion of side information uniformly improves recommendation performance. Transformers4Rec library is available at https://github.com/NVIDIA-Merlin/Transformers4Rec/.

1 INTRODUCTION

Recommender systems improve users experience in online services like e-commerce, news portals, social networks, media platforms, and many others, by providing personalized suggestions and helping users deal with large catalogs of items and information overload. In recent years we have observed an increased research interest in sequential recommendation approaches [11] which explicitly model sequences of user interactions to better infer preference or context changes over time. In a number of these settings only the most recent interactions are available, and in all domains for fresh or anonymous users only the interactions from the current user session are available. This common scenario is addressed by a sub-genre of sequential recommendation known as session-based recommendation [35, 56], in which only a short sequence of very recent user interactions is available for predicting the next in-session interaction.

Over the past decade there has been a trend toward leveraging and adapting approaches proposed by Natural Language Processing (NLP) research like Word2Vec [37], GRU [7], and Attention [3] for recommender systems (RecSys). The phenomena is especially noticeable for sequential and session-based recommendation where the sequential processing of users interactions is analogous to the language modeling (LM) task and many key RecSys architectures have been adapted from NLP, like GRU4Rec [18] – the seminal
Recurrent Neural Network (RNN)-based architecture for session-based recommendation.

More recently, Transformer architectures [55] have become the dominant technique over convolutional and recurrent neural networks for language modeling tasks. Because of their efficient parallel training, these architectures scale well with training data and model size, and are effective at modeling long-range sequences. They have similarly been applied to sequential recommendation in architectures like SASRec [24], BERT4Rec [50] and BST [5] and to session-based recommendation in [6, 12, 33, 44, 51, 62, 66].

The HuggingFace (HF) Transformers library [57] was “established with the goal of opening up advancements in NLP to the wider machine learning community” and has become very popular among NLP researchers and practitioners, providing standardized implementations of the state-of-the-art Transformer architectures produced by the research community, often within days or weeks of their publication. In this paper we introduce Transformers4Rec1, an open-source library which adapts and extends the HF Transformers library for use in recommender systems. With Transformers4Rec, RecSys researchers and practitioners can easily experiment with the latest NLP Transformer architectures for sequential and session-based recommendation tasks and deploy those models into production.

We have leveraged and evolved the Transformers4Rec library to win two recent session-based recommendation competitions: the WSDM WebTour Workshop Challenge 2021, organized by Booking.com [48], and the SIGIR eCommerce Workshop Data Challenge 2021, organized by Coveo [42].

In this paper, we used the Transformers4Rec library to conduct a comprehensive empirical analysis of session-based recommendation, comparing different Transformer architectures and training approaches with popular session-based recommendation baselines. Our experiments which we discuss in detail in Section 4 on two e-commerce and two news portals datasets show that modern Transformers architectures improve the accuracy of session-based recommendation compared to the best neural and non-neural baselines. In addition we also examine the effect of applying different training techniques – Causal LM (CLM), Masked LM (MLM), Permutation LM (PLM), and Replacement Token Detection (RTD) for a single Transformer architecture - XLNet[63]. Finally, we explore several techniques to include side information such as item and user context features in order to establish best practices and improve recommendation performance even further. Specifically in this work we investigate the following research questions:

RQ1: Can transformer-based architectures provide accurate next-click predictions for the usually short user sequences found in the session-based recommendation task?
RQ2: How do the training techniques of CLM, MLM, PLM, and RTD comparatively perform for the task of session-based recommendation?
RQ3: What are effective approaches to integrate additional features, commonly referred to as side information, into transformer architectures in order to improve recommendation performance?

Our contributions in this paper are threefold:

1https://github.com/NVIDIA-Merlin/Transformers4Rec

(1) We provide an overview of the relationship between the fields of NLP and Sequential / Session-Based recommendation, highlighting similarities and differences between the algorithms developed in each field.
(2) We introduce the open-source library Transformers4Rec which wraps the widely popular HuggingFace’s Transformers library and allows researchers and practitioners in the RecSys community to quickly and easily explore transformer architectures in the context of sequential and session-based recommendation.
(3) We perform an empirical analysis with broad experimentation of modern NLP based Transformer architectures for the task of session-based recommendation, establishing their performance relative to popular baselines. We further compare CLM, MLM, PLM, and RTD as training techniques for session-based recommendation using XLNet. Finally we examine three methods of adding side information to session-based recommenders.

Our motivation in the development of Transformers4Rec is to enable researchers and practitioners alike leverage the latest developments of NLP within the context of sequential and session-based recommendation and to take advantage of the vast amount of research and development happening there, bridging the gap between these two communities.

2 THE RELATIONSHIP BETWEEN NLP AND RECSYS RESEARCH

The field of NLP has evolved significantly over the past eight years, particularly due to the increased usage of deep learning. Mirroring this, state of the art NLP approaches have inspired RecSys practitioners and researchers to adapt those architectures, especially for sequential and session-based recommendation problems, as illustrated in Figure 1.

Early neural language models [36, 45] focused on learning representations where words with similar syntax and meaning are represented in the same regions of the vector space. The distributed vector representations methods were then extended to represent sentences and paragraphs [25]. In the RecSys domain these neural methods were adapted to learn item, user or context embeddings by taking their co-occurrence into account within the users’ history of interactions. Prod2Vec [14] learnt product representations using the Word2Vec skip-gram model [36]. Similar to Doc2Vec [25], MetaProd2Vec [54] extended Prod2Vec objectives by including items’ metadata in the neural network. The pre-trained embeddings were then used to recommend the most similar products to the query item.

Another popular NLP-inspired approach for Recommender Systems was the usage of RNNs for sequential and session-based recommendation, using the sequential nature of item interactions in a users’ session and their past visits. GRU4REC [18] applied a GRU model to session-based recommendation, generating next-click prediction for an active session. Unlike the relatively small fixed-size word vocabulary in NLP, the set of items in recommender systems is often very large and fast training of a scalable model is important. To overcome this GRU4REC included different ranking pairwise loss functions which allow for more efficient training, a technique...
that we also support in Transformers4Rec. Following GRU4Rec, some works explored techniques to include additional features (side information) other than item ids [20, 40]. RNN-based model were improved in [53] by employing data augmentation and a method to account for shifts in the input data distribution.

In 2016 the architectures that dominated NLP began to change when attention mechanisms [3] were introduced, demonstrating their effectiveness on long sequences. Attention was originally applied to recommender systems in the Neural Attentive Recommendation Machine (NARM) architecture [26] which incorporated an attention mechanism into an RNN architecture as an additional layer, proposing that attention was able to capture not only the user’s sequential behavior but also their main purpose in the current session. Conversely, Attentional FM [61] used an attention network to learn the importance of feature interactions in the non sequential Factorized Machines model.

2.1 Transformers for NLP

In 2017 the seminal Transformer architecture [55] was introduced as an efficient alternative to the RNN-based sequential encoder-decoder network with self-attention. The self-attention mechanism is capable of representing dependencies within the sequence of tokens, favors parallel processing and scales nicely for long sequences. With its host of benefits, transformer architectures became the default choice for the majority of NLP tasks, and many variants such as GPT-2 [47], BERT [10], XLNet [63] followed. All these models proposed novel pre-training approaches for language modelling and adapted the fundamental building block of self-attention to take into account language modeling specificities.

GPT-2 [47] pre-trained a stack of Transformer decoder blocks using a Causal LM approach where the next token is predicted given the context of the previous ones. To avoid information leakage from the right context, it was proposed a masked self-attention mechanism where each token has only access to previous tokens hidden states. BERT [10] used the Transformer encoder block instead and represented the input as a sum of the word’s embeddings and its absolute position embeddings. Unlike GPT2, BERT used the fully contextual self-attention mechanism where the word has simultaneously access to past and future contexts. To leverage information from both directions, BERT introduced Masked LM training which randomly masks 15% of the tokens in the input sequence, and requires the encoder model to predict the original tokens using non-masked information from its surroundings. However, the input of GPT2 and BERT is a fixed-size block of words requiring segmenting long texts into chunks without respecting the natural structure of sentences or paragraphs. Transformer-XL [9] solved the context fragmentation by introducing a segment recurrence mechanism where the hidden states of previous blocks are cached and used to extend the context of the new upcoming segment. Moreover Transformer-XL replaced the absolute position embeddings input by a relative positional encoding vector directly incorporated in the self-attention layer. Similar to GPT-2, the model was pre-trained using Causal LM and a masked self-attention mechanism.

In addition to the context fragmentation issue, masking the input in BERT results produces a discrepancy between pre-training and fine-tuning. XLNet [63] defined yet another pre-training approach, called Permutation LM, which keeps the original input sequence and uses a permutation factorization at the level of the self-attention layer to define the accessible bidirectional context. At each iteration, the context of the target token is defined based on its position index in the permutation. As the target token is not masked, XLNet extends Transformer-XL self-attention mechanism with an additional stream, called query stream attention, which represents the relative position without accessing content.

BERT and XLNet pre-training tasks were defined over a small subset of tokens in the sequence. To learn from all input tokens, ELECTRA [8] added a discriminator network to BERT that is pre-trained to predict for every token whether it is an original or an artificial replacement. The Replacement Token Detection (RTD) task consists of replacing masked positions of the Masked LM task by random tokens. The general generator-discriminator architecture was jointly pre-trained using Masked LM and RTD tasks. In our empirical analysis we experiment with many of the Transformer architectures and training approaches described in this section for the session-based recommendation task.

2.2 Transformers for Sequential Recommendation

The empirical study conducted in RoBERTa [29] demonstrated the effectiveness of Transformer-based architectures when trained on long sequences and over large set of pre-training data, motivating RecSys researchers to use and adapt these architectures for sequential recommendation. AttRec was proposed in [64], utilizing the self-attention mechanism to infer the item-item relationship from the user’s historical interactions and estimate weights of each item in the user’s trajectories. They also proposed a collaborative metric learning component to model user long-term preference.
with other user contextual features. Sequential Deep Matching (SDM) [34] concatenates categorical input features like categories, brands and shops and feeds to a self-attention module. We include side information modelling in our work as a part of RQ3 where we investigate effective approaches to represent continuous numerical features and to combine them with categorical features.

Finally, the Contextual Self-Attention Network (CSAN) [21] explores user heterogeneous sequential behaviors, which include a diversity of actions and multi-modal content (e.g. structured data, image, text). They propose a feature-wise self-attention to extract different aspects of the sequence to model the complicated correlations.

2.3 Transformers for Session-based Recommendation

Transformers have been shown to outperform RNNs sequential recommendation tasks even in cases where user sessions are shorter than sequences used in NLP, and a number of works have applied Transformers and the Self-Attention mechanism to the task of session-based recommendation [6, 12, 33, 44, 51, 62, 66]. The authors of [66] argue that some items in a session might be irrelevant to users preferences or become disturbances for modelling. So they propose a preference-aware mask based on self-attention for better capturing the users preferences over items within sessions. Similarly [44] proposed a modified self-attention mechanism to estimate the items importance for a session. In [62], self-attention was combined with graph neural networks to capture both short and long-range dependencies and enhance session representations.

Most of aforementioned works on session-based recommendation with Transformers use an auto-regressive (CLM) approach, using only interactions before the target item as input. Only [6] use a training scheme similar to BERT [10] autoencoding (MLM), which allows the usage of future in-session interactions (privileged information) during training.

In our work we perform a comprehensive analysis on a wide range of Transformer architectures and training approaches. Our work also uniquely explores different techniques to leverage side information in order to improve recommendation accuracy. Finally, while the other works focus on the e-commerce domain, ours is the first work on Transformers for news recommendation, which poses some specific challenges like shorter sessions, intense user and item cold-start problem and rapid decay of item relevance.

3 TRANSFORMERS4REC

To effectively bridge the gap between language modeling (NLP) and sequential / session-based recommendation tasks, we have developed Transformers4Rec based upon HuggingFace (HF) Transformers library. HF Transformers is an open-source library [57] with over 400 contributors that provides standardized efficient implementations of recent Transformer architectures, currently 63 and counting. The library is designed for both research and production. Models are composed of three building blocks: (a) a tokenizer, which converts raw text to sparse index encodings; (b) a transformer architecture; and (c) a head for NLP tasks, like Text Classification, Generation, Sentiment Analysis, Translation, Summarization, among others. In our work, we leverage only the transformer architectures building block (b) and their configuration classes, adding specialized heads for the recommendation problem.

Sequential and session-based recommendation have some key differences from language models that are specifically addressed by our library: (a) the inclusion of additional side information; (b) the usage of ranking metrics for evaluation; and (c) the emulation of the intense concept drift faced in RecSys compared to NLP models through the use of temporal incremental training and evaluation, as will be described in more detail later. With Transformers4Rec, the RecSys community can immediately make use of state-of-the-art NLP research and leverage novel Transformer architectures, exploring them for different RecSys use cases and datasets, and identifying which ones perform better.
3.1 Transformers4Rec Pipeline

As shown in Figure 2, Transformers4Rec is an end-to-end RecSys framework that encompasses data pre-processing, model training and evaluation. The framework was developed in Python, with PyTorch and HF Transformers as its core dependencies, building upon them to provide a pipeline described next.

3.1.1 Data preprocessing and feature engineering. Data pre-processing is a common bottleneck of RecSys production pipelines. It is the focus of NVIDIA NVTabular library which provides GPU-accelerated preprocessing of terabyte sized recsys datasets. Transformers4Rec and NVTabular are seamless integrated for being co-developed. NVTabular not only supports common and advanced feature engineering techniques, but also specialized ops for sequential and session-based recommendation, like grouping time-sorted interactions by user or session and truncating the sequences to the first or last N interactions. The preprocessed data is saved to the structured and queryable Parquet format. Transformers4Rec leverages the NVTabular data loader which reads Parquet files directly into GPU memory, making model training and evaluation faster. Transformers4Rec also uses as input a configuration file to set which features should be used by the model, their type (e.g. continuous, categorical) and metadata (e.g. cardinality).

3.1.2 Model training and evaluation. The HF Transformers library provides its own optimized training and evaluation pipeline for NLP tasks, which is managed by the Trainer class. With Transformers4Rec we inherit from this class and specifically override the predict() and evaluate() methods to adapt them to the recommendation problem, keeping its original train() method, as it is identical for NLP and sequential recommendation. The Transformers4Rec Meta-Architecture, detailed in Section 3.2, is a highly configurable component which defines the computational graph for features processing, sequence masking and processing with Transformers, prediction heads and loss functions.

The evaluation of sequential recommendation and session-based recommendation is performed using traditional Top-N ranking metrics such as NDCG@N, Recall@N, Precision@N, MAP@N, among others, which are logged to a diverse number of formats as shown in the Outputs section of Figure 2. The library supports an incremental training and evaluation protocol [39, 40, 51], which emulates realistic production scenario where the RecSys model is retrained (fine-tuned) with streaming data within a specified frequency (e.g. once a day, once an hour) and deployed for inference of the sessions from the next time period. More details on this protocol are found in Section 4.1.

3.2 Transformers4Rec Meta-Architecture

The Transformers4Rec’s Meta-Architecture is presented in Figure 3. Input features, which can be sparse categorical features or continuous numerical features, are normalized and combined by the Features Processing module, which produces the interaction embedding. The sequence of the interaction embeddings is then masked by the Sequence Masking module according to the training approach (e.g., Causal LM, Masked LM) and fed to the Sequence Processing module, which contains stacked Transformer blocks, whose number of blocks and architecture type (e.g. GPT-2, Transformer-XL, XLNet, Electra) are also configurable. It outputs a vector for each position in the sequence, which is then projected to represent a sequence embedding. Finally, the Prediction head module can be configured for different tasks: items prediction (for item recommendation) or sequence-level predictions (classification or regression).

In this paper experiments we used the items prediction head. It was composed by an output layer using the tying embeddings technique (see Section 3.2.3) i.e., weight-tying the projection layer to the item embedding matrix weights, followed by a softmax layer to predict the relevance scores over all items. Cross-entropy loss was used, but other pairwise losses functions are available. Some options and techniques of the meta-architecture are described next.

---

2A TensorFlow implementation of Transformers4Rec is planned for a near future, as HF Transformers also offers a TF version.

3https://github.com/NVIDIA/NVTabular/
3.2.1 Input Feature Representation. NLP models represent words and sub-words as token ids. Similarly, in RecSys the item id is the most important to represent the interaction, but many other features are generally available to provide additional information from item metadata and user context. The Transformers4Rec supports multiple interaction-level features, that can be normalized and combined in different ways. The size of categorical embeddings can be proportional to the features cardinality or have a fixed size. Continuous numerical features can be represented as a real-valued scalar, as a linear projection from a scalar, or as a Soft One-Hot Encoding

\[ \text{concat}(x^{(u)}_{k}, f_{1:k}^{(u)}, \ldots, f_{l:k}^{(u)}) \]

The concatenation merge consists in simply concatenating the item id feature \( x^{(u)}_{k} \) with the other available input features for the interaction at position \( k \) as follows: \( m_{k} = \text{concat}(x^{(u)}_{k}, f_{1:k}^{(u)}, \ldots, f_{l:k}^{(u)}) \).

In an element-wise merge the additional features are first element-wise summed and then element-wise multiplied by the item embedding as follows: \( m_{k} = x^{(u)}_{k} \odot (1 + f_{1:k}^{(u)} + \ldots + f_{l:k}^{(u)}) \).

In order to achieve this the embeddings of item id and all additional features must share the same dimension. In our library we support the element-wise merge approach proposed by [4], which adds 1 to the resulting summation of additional features embeddings. As those embeddings are randomly initialized by 0-mean Gaussian distributions, the multiplicative term will have mean 1 and can act as a mask/attention mechanism over the item embedding.

3.2.3 Tying Embeddings. The NLP community commonly uses a technique that ties the input embedding weights with the output projection layer matrix [22, 46], motivated by the fact that input and output of models are in the same space (words). For RecSys, a particularly important benefit is the reduction of model parameters, as the number of embeddings of high-cardinality categorical features is much larger than in NLP. Having this explicit weight-tying also helps the network to regularize; in particular rare item embeddings benefit from the more direct output layer updates at each training step. Perhaps most importantly in the RecSys use case, tying embeddings introduces a matrix factorization operation between the item embeddings and the final representation of the user or session. This technique was originally adapted from NLP to RecSys in [17] and demonstrated to be particularly effective in the NVIDIA.AI team solution for Booking.com Challenge [48]. A formal description of the tying embeddings technique is available in our online Appendix A[1].

In early experiments, we found out that tying embeddings improved the NDCG@20 of both GRU and of all Transformer architectures, by an average of 6.7% for an e-commerce dataset and 1% for a news dataset. Thus we enabled tying embeddings by default for all experiments reported in this paper and strongly suggest its use in neural-based sequential recommendation architectures.

3.2.4 Regularization. Our Meta-Architecture supports a number of regularization techniques like Dropout [49], Weight Decay [30], Softmax Temperature Scaling [16], Layer Normalization[2], Stochastic Shared Embeddings [58], and Label Smoothing [52]. In our experiments we optimize all regularization techniques through the hyperparameter optimization carried out in our empirical evaluation. While many methods failed to make a significant difference, we found that the Label Smoothing was particularly useful at improving both train and validation accuracy. This technique was also shown to improve models generalization and calibration in [43].
3.2.5 Loss functions. Our framework supports training with many different loss functions, including cross-entropy (XE) and the following pairwise losses: BPR, TOP1 [19], BPR-max and TOP1-max [17]. In the experimentation for this paper we do not explore different loss functions, but stick with a single loss function for our empirical analysis (cross-entropy) as we explore many other variables.

3.2.6 Extensibility. The Meta-Architecture modules are regular PyTorch modules\(^3\) that can be combined or replaced with other custom modules. It is possible for example to have multiple input sequences being processed by separate Transformer architectures and then having the outputs of those towers combined. Multi-task learning is also enabled by creating a custom prediction head combining multiple loss functions.

4 EMPIRICAL ANALYSIS OF TRANSFORMERS FOR SESSION-BASED RECOMMENDATION

In this section, we present an empirical analysis of different Transformer architectures and training approaches for the task of session-based recommendation. We describe our methodology and discuss our research questions results.

4.1 Methodology

This section describes the training and evaluation protocols, including metrics and the hyperparameter tuning process used for the experiments. We also describe the datasets and their preprocessing.

4.1.1 Incremental Training and Evaluation. Online services receive a continuous stream of user interactions which makes the available dataset larger every day [65]. This scenario increases the time and computational resources required to train models and presents engineering challenges for large-scale recommender systems. In this work, as in [39, 40, 51, 65], our experiments are performed using incremental retraining. For each algorithm, a sliding window with a single time unit (e.g. day or hour) is provided in temporal order and only once, to train the algorithms incrementally. For a nearest neighbor algorithm like V-SkNN, STAN or VSTAN (described in Section 4.1.5), this means that the algorithm needs to keep in memory a sample of sessions from past time windows. For neural networks, this means fine-tuning the parameters of a model already trained with past data.

Experiments are performed using incremental evaluation, as done in [39, 40, 51], which allows us to emulate a common production environment scenario where the recommendation algorithms are continuously trained and deployed once a day or even once an hour to serve recommendations for the next time period. For our experiments sessions are split into time windows \(T_i\) with a length of one day for ecommerce datasets and one hour for news datasets. Evaluation is performed for each next time window \(T_{i+1}\) with \(i \in \{1, \ldots, n - 1\}\), using sessions from past time windows for training \([T_1, \ldots, T_i]\). The sessions of each time window are split 50:50 between validation and test set. The time window validation sets are used for hyperparameter tuning and test sets for reporting metrics. The final reported metrics are the average of five independent runs with different random seeds, using the best configuration found during the hyperparameter optimization process. For each run, the metrics are the averages of all time windows, i.e., Average over Time (AoT), to benefit algorithms that are consistent at providing accurate recommendations over time.

4.1.2 Hyperparameter optimization. For each set of experiment group, composed by an algorithm, training approach and dataset, we perform bayesian hyperparameter optimization for 100 trials - running five in parallel - and optimizing towards maximizing NDCG@20 of the validation set. To reduce the possibility of overfitting over specific days, the hyperparameter tuning process performs incremental training and evaluation only on the first 50% of the available days for each dataset (see Table 1) and the reported metrics are computed on the test set for all available days. In our results we report the average of five runs using the best hyperparameters. Our Online Appendix C [1] details the hyperparameters search space explored in our experiments, along with their best values found.

4.1.3 Metrics. We evaluate the algorithms for their ability to predict the last interacted item in a session. As sessions lengths range between 2 and 20 after preprocessing, this task is equivalent to next-click prediction in the sense that recommendations for all positions of sequences are represented in the evaluation. The following information retrieval metrics are used to compute Top-20 accuracy of recommendation lists containing all items: Normalized Discounted Cumulative Gain (NDCG@20) and Hit Rate (HR@20), which is equivalent to Recall@n when there is only one relevant item in the recommendation list. NDCG accounts for rank of the relevant item in the recommendation list and is a more fine-grained metric than HR, which only verifies whether the relevant item is among the top-n items. Because of this we optimize for NDCG@20 in our hyperparameter tuning and consider it to be the primary metric in our study.

4.1.4 Datasets and Preprocessing. We have selected for experiments the e-commerce and news domains, where session-based recommendation is very suitable. In the news domain, many users browse anonymously with only their last interactions available. In the e-commerce domain, besides the user cold-start problem, user sessions tend to be targeted to a specific purchase need, so interactions from the current session provide more useful information than past interactions for the user context. We have selected two datasets for each of the domains, described as follows:

- **REES46 eCommerce**\(^6\) - This dataset contains seven months of user session from a multi-category online store, including events like views, add-to-card and purchase events. As this is a large dataset, we use in our experiments only events from the month of Oct. 2019.
- **YOOCHOOSE eCommerce**\(^7\) - This dataset was released for the RecSys Challenge 2015 and is composed by clicks and buying events from user sessions on e-commerce. We use only the use interactions table, as they form the majority of data and we are interested in next-click prediction.

\(^3\) Or Keras layers in the future TensorFlow version of Transformers4Rec.

\(^6\)https://www.kaggle.com/mkechinov/e-commerce-behavior-data-from-multi-category-store

\(^7\)https://2015.recsyschallenge.com/challenge.html
be trained incrementally like the Transformers4Rec framework\(^{10}\). In the case of the GRU4Rec implementation, which does not support incremental training, we used two approaches to train individual models for each evaluation time window \(T_i\): (1) Full Training (FT) - Trains a model on all available time windows prior to the evaluation window from \(T_i\) to \(T_{i-1}\), and (2) Sliding Window Training (SWT) - Trains a model with the last \(w\) time windows, from \(T_{i-w}\) to \(T_{i-1}\). We defined the sliding window size \(w\) as 20\% of the number of days available for each dataset, i.e. 6 days for REES46 eCommerce, 36 days for YOOCHOOSE, and 3 days for both G1 and ADRESSA.

### 4.2 RQ1: Can transformer-based architectures provide accurate next-click predictions for the shorter user sequences found in the session-based recommendation task?

Transformers have been shown to outperform RNNs for long sequences in both NLP and sequential recommendation task, however as discussed in Section 2.3, the average session length (see Table 1) is much shorter in a session-based setting. In this section we explore the effectiveness of Transformers for session-based recommendation. The results for all research questions are shown in Table 2 for easy comparison. For each research question, the best results for a metric are printed in bold and marked with a * if they significantly\(^{11}\) outperform all other algorithms. The baselines for each research question are highlighted in light gray, and for RQ1 the best performing baselines are underlined.

For RQ1 we focus on the results of Transformers architectures trained with their original training approaches (under parenthesis) relative to the neural and non-neural baselines. From Table 2 we see that the Session k-NN algorithms (V-SkNN, STAN, and VSTAN) are indeed strong baselines for session-based recommendation, with higher HR@20 than some of the Transformer architectures, however GRU4REC is the best baseline for both e-commerce datasets in terms of NDCG@20, under both FT and SWT training approaches. For the news datasets GRU4REC underperforms compared to the Session k-NN methods, which was also observed in [39], however GRU is the best performing baseline for the ADRESSA news on both NDCG@20 and HR@20, and among the best for G1 news.

Turning our attention to the Transformer architecture results we see that they achieve the best NDCG@20 across all four datasets, but by a much larger margin (+8.95\% NDCG) on both e-commerce datasets than on the news datasets. We believe that this is due to the fact that the e-commerce sessions are longer than news sessions (Table 1), requiring more complex models. No single Transformer architecture or training technique performs best across all datasets raising the question of how to select the appropriate algorithm without the extensive experimentation done in this paper. In RQ2 we attempt to isolate the effect of training technique by training the XLNet architecture with the available options, producing a clearer choice.

---

\(^{10}\)To make V-SkNN and VSTAN support incremental training, we updated their implementations to compute the IDF statistics incrementally.

\(^{11}\)As errors around the reported averages were normally distributed, we used paired Student’s t-tests with \(p < 0.05\) and Bonferroni correction for significance tests.
Table 2: Experimental Results: RQ1 / RQ2 / RQ3

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>REE46 eCommerce</th>
<th>YOOCHOOSE eCommerce</th>
<th>G1 news</th>
<th>ADRESSA news</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V-SiNN</td>
<td>0.2187 0.4662</td>
<td>0.2957 0.5119</td>
<td>0.3511 0.6601</td>
<td>0.3596 0.7210</td>
</tr>
<tr>
<td>STAN</td>
<td>0.2194 0.4797</td>
<td>0.3082 0.5196</td>
<td>0.3570 0.6681</td>
<td>0.3635 0.7246</td>
</tr>
<tr>
<td>VSTAN</td>
<td>0.2200 0.4578</td>
<td>0.3097 0.5206</td>
<td>0.3586 0.6668</td>
<td>0.3617 0.7241</td>
</tr>
<tr>
<td>GRU4Rec (FT)</td>
<td>0.2231 0.4414</td>
<td>0.3442 0.5891</td>
<td>0.2596 0.5029</td>
<td>0.3007 0.6652</td>
</tr>
<tr>
<td>GRU4Rec (SWT)</td>
<td>0.2265 0.4359</td>
<td>0.3431 0.5885</td>
<td>0.2666 0.5183</td>
<td>0.2967 0.5948</td>
</tr>
<tr>
<td>GRU (CLM)</td>
<td>0.2139 0.4315</td>
<td>0.2975 0.6129</td>
<td>0.3549 0.6632</td>
<td>0.3799 0.7133</td>
</tr>
<tr>
<td>GPT-2 (CLM)</td>
<td>0.2165 0.4338</td>
<td>0.2975 0.6055</td>
<td>0.3560 0.6620</td>
<td>0.3790 0.7398</td>
</tr>
<tr>
<td>Transformer-XL (CLM)</td>
<td>0.2197 0.4404</td>
<td>0.3585 0.6133</td>
<td>0.3294 0.6192</td>
<td>0.3811* 0.7382</td>
</tr>
<tr>
<td>BERT (MLM)</td>
<td>0.2218 0.4672</td>
<td>0.3750* 0.6349*</td>
<td>0.3549 0.6549</td>
<td>0.3725 0.7221</td>
</tr>
<tr>
<td>ELECTRA (RTD)</td>
<td>0.2430 0.4768</td>
<td>0.3722 0.6294</td>
<td>0.3588 0.6600</td>
<td>0.3729 0.7226</td>
</tr>
<tr>
<td>XLNet (PLM)</td>
<td>0.2422 0.4760</td>
<td>0.3681 0.6282</td>
<td>0.3551 0.6634</td>
<td>0.3673 0.7212</td>
</tr>
<tr>
<td>RQ2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLNet (PLM) - original</td>
<td>0.2422 0.4760</td>
<td>0.3681 0.6282</td>
<td>0.3551 0.6634*</td>
<td>0.3673 0.7212</td>
</tr>
<tr>
<td>XLNet (CLM)</td>
<td>0.2108 0.4219</td>
<td>0.3557 0.6079</td>
<td>0.3551 0.6508</td>
<td>0.3770 0.7378*</td>
</tr>
<tr>
<td>XLNet (RTD)</td>
<td>0.2546* 0.4886*</td>
<td>0.3776 0.6373</td>
<td>0.3609 0.6611</td>
<td>0.3816 0.7329</td>
</tr>
<tr>
<td>XLNet (MLM)</td>
<td>0.2428 0.4763</td>
<td>0.3776 0.6384*</td>
<td>0.3607 0.6605</td>
<td>0.3822 0.7349</td>
</tr>
<tr>
<td>RQ3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLNet (MLM) - item id</td>
<td>0.2428 0.4763</td>
<td>0.3776 0.6384</td>
<td>0.3607 0.6605</td>
<td>0.3822 0.7349</td>
</tr>
<tr>
<td>Concat. merge</td>
<td>0.2522 0.4782</td>
<td>- -</td>
<td>0.3652 0.6714</td>
<td>0.3912* 0.7488*</td>
</tr>
<tr>
<td>Concat. merge + SOHE</td>
<td>0.2542 0.4858</td>
<td>- -</td>
<td>0.3675* 0.6721*</td>
<td>0.3886 0.7463</td>
</tr>
<tr>
<td>Element-wise merge</td>
<td>0.2529 0.4854</td>
<td>- -</td>
<td>0.3614 0.6678</td>
<td>0.3892 0.7433</td>
</tr>
</tbody>
</table>

4.3 RQ2: How do the training techniques of Causal LM, Masked LM, Permutation LM, and Replacement Token Detection perform comparatively for the task of session-based recommendation?

RQ1 shows the effectiveness of Transformer-based language models for session-based recommendation, however these language models include two components: (1) the pre-training approach and (2) the Transformer architecture. In RQ2 we wanted to isolate the gain of performance related to the different training approaches and compare their effectiveness when considering the short sequences of the session-based recommendation task. The XLNet architecture was designed to leverage the best of both auto-regressive language modeling and auto-encoding with its Permutation Language Modeling training method. It can also be used with MLM and RTD, so in RQ2 we used XLNet as the Transformer block in our meta-architecture and trained the model with three methods different from its original PLM: CLM, MLM and RTD, introduced in Section 2.1.

Table 2 shows the results of the XLNet model when trained with different approaches. First, we notice that MLM outperforms CLM for all datasets. This could be explained by the fact that the former has access to future in-session interactions (privileged information) during training and therefore better represents the context of the items. In addition, MLM randomly masks different positions of the session for each training step, working as a data augmentation approach. For both PLM and MLM, partial prediction plays a role of reducing optimization difficulty by only predicting tokens with sufficient context. Interestingly, MLM outperforms the original pre-training approach used by the XLNet model (PLM) for all datasets on NDCG@20, suggesting that MLM performs better when context is limited. Finally, XLNet (RTD) extends XLNet (MLM) with the additional RTD task that learns from all the items in the sequence and therefore provides more effective training signals. From Table 2 we see that XLNet (RTD) has the highest NDCG@20 on three datasets of all combinations explored in both RQ2 and RQ1 and achieves 99.69% of the best performing algorithm for both NDCG@20 and HR@20 across all datasets, making it an excellent default choice when hyperparameter tuning across all architectures and training techniques isn’t an option. It improves the benefit from Transformers further on REE46 e-commerce and YOOCHOOSE e-commerce (+14.15% NDCG@20 and +9.75% NDCG@20 respectively relative to the best baseline), and by a small amount on G1 news, where ELECTRA (also trained on RTD) achieved the best results in RQ1. Our finding is particularly interesting as this combination of Transformer architecture and training technique - XLNet (RTD) - is (to our knowledge) novel and could be beneficial to the NLP community as well.

4.4 RQ3: What are effective approaches to integrate additional features, commonly referred to as side information, into transformer architectures in order to improve recommendation accuracy?

It is a common practice in RecSys to leverage additional tabular features of item metadata and user context, providing the model more information for meaningful predictions. In this research question, we have compared three different approaches to include side information into Transformer architectures: (1) Concatenation merge, (2) Concatenation merge with continuous features represented by the Soft-One Hot Encoding (SOHE) technique, and (3) Element-wise merge, described in Section 3.2.2. We empirically observed that applying layer normalization technique [2] individually to each feature representation before merging was essential to obtain improvements in accuracy when including continuous features, aligned with what we also observed in [42], so we used this normalization technique. In Online Appendix B [1], we describe the feature engineering steps and why the YOOCHOOSE dataset was not included in this analysis. We chose for this analysis XLNet (MLM), based on its performance for RQ2 over all datasets, leaving as future work whether XLNet (RTD) could further improve performance. All proposed techniques that include additional features improve the performance of XLNet (MLM) compared to using only the item id feature; in Table 2 we see a relative improvement of NDCG@20 of 2.13% on average for the news datasets and of 4.72% for the REE46 e-commerce dataset which includes more informative features. Concatenation Merge using Soft-One Hot Encoding (SOHE) provided
the highest NDCG@20 and HR@20 for REES46 e-commerce and G1 news, suggesting that the SOHE technique is able to provide effective representation of continuous features to be combined with categorical embeddings. For ADRESSA news, the Concatenation Merge performed the best, which was similarly observed in [38].

5 CONCLUSION

In this paper we have presented Transformers4Rec, an open source library designed to enable RecSys researchers and practitioners to quickly and easily explore the latest developments of the NLP for sequential and session-based recommendation tasks. Our contributions are threefold. First, to motivate our efforts we provided an overview of the relationship between the fields of NLP and sequential / session-based recommendation, highlighting the corresponding algorithms developed in each field. Second, the library itself, which wraps HF Transformers, providing all of the necessary functionality required to use Transformer architectures in a RecSys setting. Third, we have performed experiments on the applicability of modern NLP Transformer architectures to the task of session-based recommendation, comparing CLM, MLM, PLM, and RTD as training techniques for session-based recommendation for XLNet, and examining three methods of adding side information to Transformers.

Through our experiments for the session-based recommendation task we have found that Transformer architectures have superior performance (+14.15% NDCG@20 and +9.75% NDCG@20 respectively relative to the best baseline) across REES46 and YOOCHOSE e-commerce datasets while performing similarly to baselines on G1 and ADRESSA news datasets. In particular, we discovered that XLNet trained with RTD, a novel combination of Transformer architecture and training technique proposed in this paper, is effective at the task of session-based recommendation with the best NDCG@20 on three datasets and achieving 99.6% of the best performing algorithm for both NDCG@20 and HR@20 across all datasets. Finally, by exploring different methods to leverage side information with Transformers, we found that aggregation by concatenation outperformed using only the item id feature by +4.72% NDCG@20 (+13.96% relative to the best popular baseline) on REES46 e-commerce and by +2.13% NDCG@20 (+2.75% relative to the best baseline) averaged across G1 and ADRESSA news for the XLNet (MLM) architecture suggesting that it is a valuable addition to session-based recommenders. We hope that these experiments encourage more exploration into the use of Transformers within the RecSys domain and that the Transformers4Rec library helps to foster an easier exchange of ideas and research between the NLP and RecSys communities.

REFERENCES


