Language Models as Recommender Systems: Evaluations and Limitations

Yuhui Zhang[1,2], **Hao Ding**[1], **Zeren Shui**[1,3], **Yifei Ma**[1], **James Zou**[1,2], **Anoop Deoras**[1], **Hao Wang**[4]
[1]AWS AI Labs [2]Stanford University [3]University of Minnesota [4]Rutgers University

Abstract

Pre-trained language models (PLMs) such as BERT and GPT learn general text representations and encode extensive world knowledge; thus, they can efficiently and accurately adapt to various downstream tasks. In this work, we propose to leverage these powerful PLMs as recommender systems and use prompts to reformulate the session-based recommendation task to a multi-token cloze task. We evaluate the proposed method on a movie recommendation dataset in zero-shot and fine-tuned settings where no or limited training data are available. In the zero-shot setting: we find that PLMs outperform the random recommendation baseline by a large margin; in the meantime, we observe strong linguistic bias when using PLMs as recommenders. In the fine-tuned setting: such bias is reduced with available training data; however, PLMs tend to under-perform traditional recommender system baselines such as GRU4Rec. Our observations demonstrate potential opportunities as well as current challenges in this novel direction.

1 Introduction

Natural language processing (NLP) has been undergoing a significant paradigm shift driven by the rapid developments of large pre-trained language models (PLMs) such as BERT [4] or GPT [14]. With the significant increase of model size and pre-training data amount, PLMs show the remarkable capability of understanding a variety of natural language tasks given task descriptions (a.k.a., prompts) and only a few or even zero demonstrations [3]. These PLMs are named as foundation models because their effectiveness incentivizes homogenization of methods for different downstream tasks [1].

In this work, we propose language model recommender systems (LMRecSys) that use powerful PLMs as recommender systems by reformulating recommendation as a language modeling task. Specifically, we convert a user's interaction sequence to a text inquiry (e.g., item sequence: $\langle \underline{1193}, \underline{661}, \underline{914} \rangle$ to text sequence: *A user watched One Flew Over the Cuckoo's Nest, James and the Giant Peach, My Fair Lady. Now the user wants to watch _ _ _.*, where _ is a special mask token.) and use PLMs to fill in the masks for recommendation.

Our work differs from two types of previous works that combine NLP and recommender systems. The first type of works uses textual information to augment item representations for better recommendation performance [12, 22, 21, 20]; for example, CDL [22] and HRNN [12] use text embeddings generated from item descriptions as one of the item features improve performance. The second type of works treats items in users' interaction sequences as text tokens in natural languages [6, 19, 18]. They adapt NLP training techniques such as masked language modeling to learn users' representations; for example, BERT4Rec [18] is trained to predict masked items in user's interaction sequences, similar to predicting masked tokens for BERT [4]. Both types of methods require training models from scratch on recommendation datasets. In contrast, our method, by converting the traditional item-based recommendation to this text-based cloze task [16], hopes to leverage PLMs to tackle two challenging problems for recommendation:

I (Still) Can't Believe It's Not Better Workshop at NeurIPS 2021.

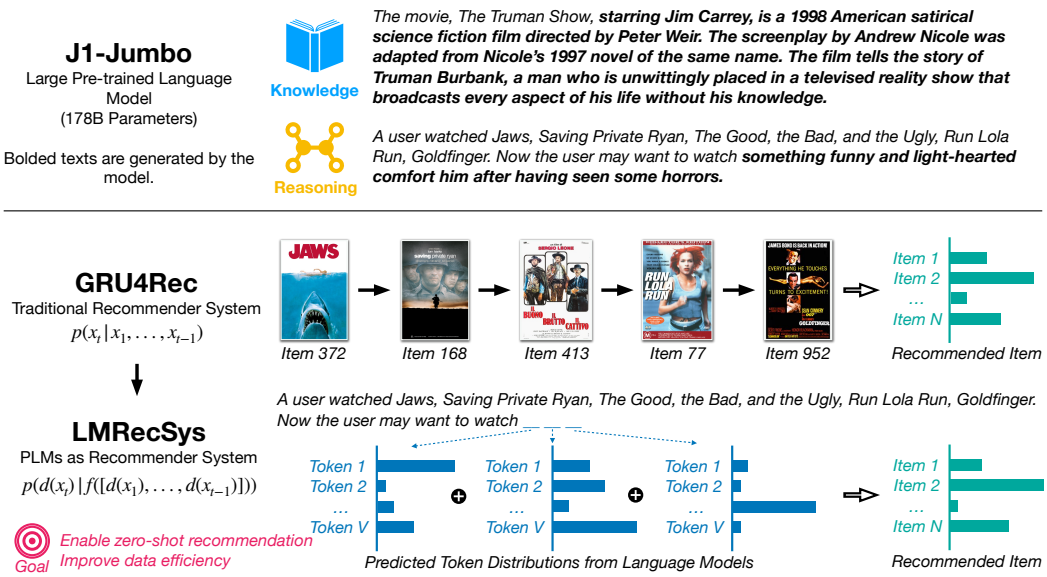

Figure 1: Motivation (**top**): large pre-trained language models possess both knowledge of items (generate the movie synopsis given the movie title) and reasoning capability (infer user interests based on the context); these are key factors to build a successful recommender system. Method (**bottom**): traditional sequential recommender operates on the item level, whereas our model use prompts to reformulate the recommendation task to a multi-token cloze task and operates on the token level; our method aims to enable zero-shot recommendation and improve data efficiency.

Zero-shot Recommendations: Generating personalized recommendations when no interaction data is available for training is an important business scenario, especially for startup companies. We refer to recommendation tasks in this setting as zero-shot recommendation [5]. Many recent works have demonstrated that PLMs acquire and store extensive knowledge into their parameters during the pre-training stage [2, 13]. They find that PLMs can even correctly answer more factoid questions than the models that explicitly use external knowledge bases [15]. We believe such knowledge could be beneficial to the zero-shot recommendation problem. As shown in Figure 1, we find that a large PLM can generate the correct movie description based on the given movie name and predict reasonable recommendations based on the movies watched by the user.

Data Efficiency: The general representations learned during the pre-training stage enable language models to easily adapt to a variety of downstream tasks. Moreover, recently-proposed prompt-based tuning approaches further improve the data efficiency [17] — PLMs can be 100x more data efficient when converting various natural language processing tasks such as text classification, question answering, natural language inference to cloze task with a prompt (in our example, *A user watched... Now the user wants to watch...* is a prompt) [11]. Similarly, we believe PLMs could also be used as data-efficient recommender systems.

We evaluate our method on movie recommendation tasks in zero-shot and fine-tuned settings. In the zero-shot setting, we find that LMRecSys outperforms random recommendations but has strong linguistic biases in its recommendations. By fine-tuning LMRecSys with prompt-tuning techniques, we improve the recommendation performance by a large margin and reduce the linguistic biases. However, the fine-tuned LMRecSys under-performs traditional state-of-the-art recommender system GRU4Rec [9]. Our attempts and observations shed light on the potential opportunities as well as current challenges in this novel direction.

2 Method

In recommendation, given a user with viewed item sequence $[x_1, x_2, ..., x_{t-1}]$, we want to predict the probability distribution of the next item $p(x_t|x_1, ..., x_{t-1})$. For traditional methods, each x_i is

embedded into a vector (item embedding), and sequence modeling techniques such as recurrent neural networks or self-attention is applied to model the item sequence (user embedding).

In language model recommender systems (LMRecSys), we first map each item to its tokenized word description (e.g., movie name) $d(x_i) = [w_{i_1}, w_{i_2}, ..., w_{i_L}]$ and use a prompt $f(\cdot)$ to convert the item sequence into the text sequence $c = f([d(x_1), d(x_2), ..., d(x_{t-1})])$ as the context. We then use the language model to estimate the probability distribution of the next item by multi-token inference, i.e., aggregating the probability of each token in the item description $p(d(x_t)|c) = p(w_{t_1}, w_{t_2}, ..., w_{t_L}|c)$. Note that with this model, we assume a one-to-one correspondence between x and $d(x)$.

The key challenge of our proposed method is multi-token inference. As language models are designed to predict a single token distribution at each time step, estimating the probability distribution of the next item that corresponds to multiple tokens (e.g., "Star Wars" in movie recommendations) remains an open problem in natural language processing research.

First, different items have text descriptions of different lengths. We can pad each item description to the same fixed length (e.g., the maximum length of all the item names) and compute the average probability of each token. Moreover, another critical choice is whether to estimate the probability distribution of each token in the item description independently or dependently. That is, whether we want to model $p(w_{t_1}, w_{t_2}, ..., w_{t_L}|c) = \prod_j p(w_{t_j}|c)$or $\prod_j p(w_{t_j}|w_{t_{<j}}, c)$. Technically, it is achieved by whether to use the language model to fill in all the masks at a time or in an auto-regressive fashion. We refer to the independent estimation as O(1) inference as we only need a single forward pass for the language model[1] and the dependent estimation as O(LN) inference as we need L forward passes for each item and LN passes for N items. While the second estimation is intuitively more accurate, it is computationally inefficient when we have a large collection of items.

When fine-tuning the PLMs for recommendation, we maximize the probability of the ground-truth item $p(d(x_t)|c)$ via the cross-entropy loss. For evaluation, we use MRR@K and Recall@K (R@K) metrics based on top K predicted items.

3 Experiments

Research Questions: We are hereby to answer the following questions:

- **Q1** Can we use pre-trained language models for zero-shot recommendation? (Section 3.1)
- **Q2** Can we fine-tune pre-trained language models to improve recommendations? (Section 3.2)

Dataset: We train and evaluate the model on a widely-used movie recommendation dataset MovieLens-1M (ML1M) [8], which consists of 6040 users, N=3883 movies, and 1M interactions.

3.1 Zero-shot Recommendations

We start with zero-shot (i.e., no interaction data is available for training) language model recommender systems (LMRecSys). For each user, we provide the model with their first 5 watched movies and use the model to predict their 6^{th} watched movie. We use the movie title as the item description and pad or truncate all the titles to L=10 tokens. We use the prompt *A user watched A, B, C, D, E. Now the user may want to watch [F].*, where each letter represents a full movie title. We compare different inference methods, model sizes, and prompts.

We compare our LMRecSys with two zero-shot baselines **Random** and **BERT-Base ItemKNN** (Using BERT to generate item embeddings and computing nearest neighbors) and two supervised baselines **POP** (Popularity-based model) and **GRU4Rec** [9] (GRU over item embeddings generated based on item ID). Table 1 shows zero-shot model performances on the MovieLens-1M dataset, while Table 2 shows the case study of different inference methods (full results see Supplementary Table 4).

3.1.1 Results & Analysis

Multi-token inference methods significantly influence the results. For O(1) inference, we pad or truncate all the titles to L=10 tokens, leave 10 masks, and use the language model to fill all the masks by one pass. BERT O(1) shows even worse performance than the random recommendation.

[1]Since the context c is the same for all the N candidate items, we only need 1 forward pass instead of N.

We find that this is because not all the movie names are exactly 10 tokens, always adding a period after 10 masks makes the text ungrammatical, and language models can hardly fill these ungrammatical texts that are unseen during pre-training. We use a simple solution by leaving 1-L masks, feeding them into the model for L times, and selecting the output for each movie based on its length [10]. We name this method as O(L) inference, which achieves 2x more R@20 than O(1) inference and 1x more R@20 than random. Inputs are guaranteed to be grammatically correct for O(L)/O(LN) inference as we only add a period after the exact name. GPT2 O(LN) shows much better performances than other inferences because of the more accurate probability estimation but much more computational cost.

Model size and prompt have small influences on the results. Model size and prompt have a significant impact on performances for many NLP tasks [3]. However, for our task, the R@20 only improves by 0.72% from GPT2-Small (117M parameters) to GPT-XL (1542M parameters). We also design a weaker prompt (*A, B, C, D, E, [F].*) and a stronger prompt (*A user watched movies A, B, C, D, E. Now the user may want to watch the movie [F].*) for comparison. However, for our task, using different prompts achieves very close performances.

Method	# Params	R@20
Zero-shot		
Random	0	0.0052
BERT-Base ItemKNN	110M	0.0599
BERT-Base O(1)	110M	0.0030
BERT-Base O(L)	110M	0.0094
BERT-Base (DAPT) O(L)	110M	0.0137
GPT2-Small O(LN)	117M	0.0667
GPT2-Medium O(LN)	345M	0.0617
GPT2-Large O(LN)	762M	0.0587
GPT2-XL O(LN)	1542M	0.0739
GPT2-Small O(LN) Weak Prompt	117M	0.0627
GPT2-Small O(LN) Strong Prompt	117M	0.0653
GPT2-Small O(LN, sum)	117M	0.0320
GPT2-Small O(LN) Calibrated	117M	0.0733
Supervised		
POP	0	0.1507
GRU4Rec	-	0.1664

Table 1: Zero-shot model performances on ML-1M.

3.1.2 Case Study

Language models show clear linguistic biases for recommendations. Despite higher performances achieved by LMRecSys compared to random predictions, we observe that models have strong linguistic biases for their top predictions and bottom predictions. Different biases are associated with different inference methods because of different probabilities modeled. O(1) inference favors generic and grammatical sequences, O(LN) inference favors uniquely long and grammatical sequences, and O(LN, sum) inference (the version of O(LN) without length normalization) favors short and grammatical sequences. All the inference methods penalize ungrammatical sequences. In other words, language models are more "language modeling" (predicting fluent and grammatically correct text) than "collaborate filtering" (recommending similar items based on user preferences). Please refer Table 2 and Supplementary Table 4 for more details.

Calibration slightly reduces linguistic biases. As language models show clear linguistic biases for recommendations, here we use a simple calibration method by subtracting the base probability predicted by the language model from the original probability $p'(d(x_t)|c) = p(d(x_t)|c) - \alpha p(d(x_t))$ [23]. Technically, the base probability is computed by feeding the model with a null context instead of the user-viewed context. By only considering the changing amount in predicted scores, movies with rare words in titles will be less penalized. The coefficient α is tuned on a small set of 50 randomly selected examples. We find that the calibration improves the R@20 by 0.66% when choosing $\alpha = 0.40$. However, linguistic bias is only slightly mitigated as shown in the Supplementary Table 4.

3.2 Fine-tuned Recommendations

We explore fine-tuning LMRecSys to improve recommendations. After obtaining the item probability distribution using different multi-token inference methods, we use the cross-entropy loss to maximize the probability of the ground-truth item. Due to computational limits, here we only use inference methods that have constant complexity. We compare different models and different amounts of training data (by controlling the session length K). Table 3 shows fine-tuned model performances (full results see Supplementary Table 5).[2]

[2]Here we use a different data processing method by only keeping the last K movies that a user watched and rated as 5 stars to reduce the popularity bias in the dataset. We name this dataset as ML-1M-5Star.

Method	Probability	Top 3 Predictions	Top Prediction Bias
BERT-Base O(1)	$\prod_i \sqrt[L]{p(w_i\|c)}$	(1) If.... (2) Them! (3) The Show	Generic and grammatical word sequences
GPT2-Small O(LN)	$\prod_i \sqrt[L]{p(w_i\|c, w_{<i})}$	(1) Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (2) Star Wars: Episode VI - Return of the Jedi (3) Star Wars: Episode V - The Empire Strikes Back	Uniquely long and grammatical word sequences
GPT2-Small O(LN, sum)	$\prod_i p(w_i\|c, w_{<i})$	(1) Raiders of the Lost Ark (2) Jurassic Park (3) Ghostbusters I	Short and grammatical word sequences

Table 2: Case study of different inference methods. *Input: A user watched Raiders of the Lost Ark, Star Wars: Ep VI-Return of the Jedi, Ran, Ghostbusters II, Gandhi. Now the user may want to watch _. Answer: Indochine*

3.2.1 Results & Analysis

LMRecSys under-performs GRU4Rec. LMRecSys with different PLMs and inference methods achieve slightly worse performances than baseline GRU4Rec after fine-tuning, probably because learning to predict the next item as multiple tokens is intrinsically more challenging than predicting the next item as a whole. RoBERTa achieves slightly better performances than BERT probably because of 10x more pre-training data.

Method	MRR@20	R@20
Random	-	0.0052
POP	0.0202	0.0817
GRU4Rec	0.0302	0.0986
BERT O(1)	0.0177	0.0843
RoBERTa O(1)	0.0204	0.0843
BERT O(L)	0.0181	0.0852
BERT (DAPT) O(L)	0.0187	0.0872

Table 3: Fine-tuned model performances on ML-1M-5Star.

Domain-adaptive pre-training (DAPT) [7] shows small improvements. To inject domain-specific knowledge, we continue pre-training the BERT on 520K movie synopses crawled from the web. We concatenate each synopsis with its title (i.e., *[synopsis]. This is the movie [Title].*), mask 30% random words or only the entire title for 1:1 ratio, and train the model to predict the masked words. We find that this hybrid masking strategy is better than a purely random masking strategy and leads to small improvements for both zero-shot and fine-tuned recommendations. The limited improvements may be due to the small data size compared to the general pre-training data.

Fine-tuned LMRecSys learn how to recommend but linguistic biases still exist for bottom predictions. From Supplementary Table 4, no clear bias is observed for the top predictions, but models still penalize movies with ungrammatical names (e.g., non-English words).

4 Discussion

In this work, we leverage PLMs as recommender systems and transform the recommendation task to the multi-token cloze task in order to enable zero-shot recommendation and improve data efficiency. Based on our observations, there are several open research questions along this new direction:

- **Multi-token inference:** Language models can accurately predict a single token distribution, but inferring the probability distribution of a span corresponding to multiple tokens remains a challenge. We find that performances vary significantly using different inference methods.
- **Linguistic biases disentanglement:** Language models tend to predict generic tokens to make the sentence fluent. How to disentangle this linguistic bias with the true probability is important for various downstream tasks using prompt-based methods.
- **Domain-knowledge evaluation and injection:** Understanding how much domain knowledge language models have and how to inject domain knowledge are both open questions. The simple approach to inject knowledge, domain-adaptive pre-training, only leads to small improvements.
- **Scales of language models:** Model size has a significant impact on the zero-shot/few-shot capability [3]. We find that J1-Jumbo (178B parameters) understands movie contents much better than J1-Large (7.5B) or GPT-2 XL (1.5B) through qualitative studies. However, we cannot evaluate these large models since token distribution is not available from APIs.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv:2108.07258*, 2021.

[2] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. In *ACL*, 2019.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *OpenAI*, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[5] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. Zero-shot recommender systems. *arXiv:2105.08318*, 2021.

[6] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *KDD*, 2015.

[7] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, 2020.

[8] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *TIIS*, 2015.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[10] Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. X-factr: Multilingual factual knowledge retrieval from pretrained language models. In *EMNLP*, 2020.

[11] Teven Le Scao and Alexander M Rush. How many data points is a prompt worth? In *NAACL*, 2021.

[12] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. Temporal-contextual recommendation in real-time. In *KDD*, 2020.

[13] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *EMNLP*, 2019.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[16] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, 2021.

[17] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, 2021.

[18] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

[19] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *KDD*, 2016.

[20] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Relational stacked denoising autoencoder for tag recommendation. In *AAAI*, 2015.

[21] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*, 2016.

[22] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *KDD*, 2015.

[23] Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *ICML*, 2021.

A Appendix

Method	Probability	Top 3 Predictions	Top Prediction Bias	Bottom 3 Predictions	Bottom Prediction Bias
BERT-Base O(1)	$\prod_i \sqrt[L]{p(w_i\|c)}$	1. If.... 2. Them! 3. The Show	Generic and grammatical word sequences	3. Shaft 2. Trois 1. Nowhere	Ungrammatical sequences
GPT2-Small O(LN)	$\prod_i \sqrt[L]{p(w_i\|c, w_{<i})}$	1. Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb 2. Star Wars: Episode VI - Return of the Jedi 3. Star Wars: Episode V - The Empire Strikes Back	Uniquely long and grammatical word sequences	3. Children Are Watching us, The (Bambini ci guardano, I) 2. Under the Domin Tree (Etz Hadomim Tafus) 1. Goodbye, 20th Century (Zbogum na dvadesetiot vek)	Ungrammatical sequences
GPT2-Small O(LN, sum)	$\prod_i p(w_i\|c, w_{<i})$	1. Raiders of the Lost Ark 2. Jurassic Park 3. Ghostbusters II	Short and grammatical word sequences	3. Garden of Finzi-Contini, The (Giardio dei Finzi-Contini, Il) 2. Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) 1. Goodbye, 20th Century (Zbogum na dvadesetiot vek)	Ungrammatical sequences
GPT2-Small O(LN) Calibrated	$\prod_i \sqrt[L]{p(w_i\|c, w_{<i})} - \alpha \prod_i \sqrt[L]{p(w_i\|w_{<i})}$	1. Star Wars: Episode VI - Return of the Jedi 2. Star Wars: Episode V - The Empire Strikes Back 3. Close Encounters of the Third Kind	Uniquely long and grammatical word sequences	3. Children Are Watching us, The (Bambini ci guardano, I) 2. Under the Domin Tree (Etz Hadomim Tafus) 1. Goodbye, 20th Century (Zbogum na dvadesetiot vek)	Ungrammatical sequences
BERT-Base O(1)Fine-tuned	$\prod_i \sqrt[L]{p(w_i\|c)}$	1. The Insider 2. Saving Private Ryan 3. Boys Don't Cry	No clear bias observed	3. Black Tights (Les Collants Noirs) 2. Halfmoon (Paul Bowles - Halbmond) 1. Beloved/Friend (Amigo/Amado)	Ungrammatical sequences

Table 4: Case study of different inference methods. *Input: A user watched Raiders of the Lost Ark, Star Wars: Episode VI - Return of the Jedi, Ran, Ghostbusters II, Gandhi. Now the user may want to watch ___. Answer: Indochine*

Method	K=4		K=5		K=6		K=7	
	MRR@20	R@20	MRR@20	R@20	MRR@20	R@20	MRR@20	R@20
Random	-	0.0052	-	0.0052	-	0.0052	-	0.0052
POP	0.0202	0.0817	0.0196	0.0828	0.0199	0.0793	0.0198	0.0763
GRU4Rec	0.0302	0.0986	0.0325	0.1210	0.0370	0.1505	0.0410	0.1542
BERT-Base O(1)	0.0177	0.0843	0.0223	0.0959	0.0257	0.1051	0.0319	0.1210
BERT-Base O(L)	0.0181	0.0852	-	-		-	-	-
RoBERTa-Base O(1)	0.0204	0.0843	0.0209	0.1012	0.0265	0.1087	0.0289	0.1250
RoBERTa-Base (DAPT) O(L)	0.0187	0.0872	-	-		-	-	-

Table 5: Fine-tuned model performances on ML-1M-5Star with different session lengths K.