arXiv:2305.11700v1 [cs.IR] 19 May 2023

Exploring the Upper Limits of Text-Based Collaborative Filtering Using Large Language Models: Discoveries and Insights

Ruyu Li[1]*, **Wenhao Deng**[1]*, **Yu Cheng**[1], **Zheng Yuan**[1], **Jiaqi Zhang**[1], **Fajie Yuan**[1]†
[1]Westlake University
{liruyu,dengwenhao,yuanfajie}@westlake.edu.cn

Abstract

Text-based collaborative filtering (TCF) has become the mainstream approach for text and news recommendation, utilizing text encoders, also known as language models (LMs), to represent items. However, existing TCF models primarily focus on using small or medium-sized LMs. It remains uncertain what impact replacing the item encoder with one of the largest and most powerful LMs, such as the 175-billion parameter GPT-3 model [4], would have on recommendation performance. Can we expect unprecedented results? To this end, we conduct an extensive series of experiments aimed at exploring the performance limits of the TCF paradigm. Specifically, we increase the size of item encoders from one hundred million to one hundred billion to reveal the scaling limits of the TCF paradigm. We then examine whether these extremely large LMs could enable a universal item representation for the recommendation task. Furthermore, we compare the performance of the TCF paradigm utilizing the most powerful LMs to the currently dominant ID embedding-based paradigm and investigate the transferability of this TCF paradigm. Finally, we compare TCF with the recently popularized prompt-based recommendation using ChatGPT[1]. Our research findings have not only yielded positive results but also uncovered some surprising and previously unknown negative outcomes, which can inspire deeper reflection and innovative thinking regarding text-based recommender systems. Codes & datasets will be released for further research.

1 Introduction

The explosive growth of online text data has highlighted the importance of text content recommendation in numerous domains, including e-commerce, news recommendation, and social media. Text-based collaborative filtering (TCF) has emerged as a critical technology that provides personalized recommendations to users based on textual data, such as product descriptions, reviews, or news articles [58, 48, 49]. The goal of TCF is to accurately capture the user's preferences and interests from textual data and provide tailored recommendations that align with their needs. TCF typically utilizes language models (LMs) as text encoders, integrated into a recommender architecture using collaborative filtering techniques [39, 12, 22] to generate user-item matching scores (see Figure 1). TCF's promising results have made it the mainstream approach for text-based recommendation [50, 58, 44].

By using LMs as item encoders, TCF naturally benefits from the latest advances in the field of natural language processing (NLP). Particularly, in recent years, large LMs such as GPT-3 [4] and

*Equal Contribution. Authorship order is determined by coin flip.

†Corresponding author. Author contributions: Fajie designed and supervised this research; Ruyu and Wenhao performed this research, in charge of key technical parts; Fajie, Ruyu, and Wenhao wrote the manuscript. Other authors have contributed to some of the experiments in this study.

[1]https://openai.com/blog/chatgpt

ChatGPT [1] have achieved revolutionary successes in modeling textual data. However, the text encoders in current TCF models are often small or medium-sized LMs, such as word2vec [31], BERT [7], and RoBERTa [29]. This may limit their recommendation capability, which raises some important questions: Can TCF achieve exceptional results by leveraging extremely large LMs with tens or hundreds of billions of parameters as text encoders? Is there an upper limit to TCF's performance by increasing the size of the text encoder to the extreme? Has the current state-of-the-art TCF with the largest LMs reached its full potential?

Clearly, the above questions are essential for directing research on the mainstream TCF paradigm. However, despite many TCF algorithms has been proposed in literature [49, 59, 25, 2, 51, 10], none of them have explicitly addressed the above questions. Therefore, rather than introducing a new algorithm as before, we aim to decipher the classic TCF models via a series of *audacious experiments* that require immense computational resources. Specifically, we explore the below sub-questions.

Q1: How does the recommender system's performance respond to the continuous increase in the item encoder's size? Is the performance limits attainable at the scale of hundreds of billions? To answer it, we conduct the empirical study by progressively expanding the size of the text encoder from 100 million to 175 billion on three recommendation datasets, using the two most representative recommender architectures: a simple two-tower CTR model DSSM [18] and a state-of-the-art sequential model SASRec [20] with Transformer [43] as the backbone.

Q2: Can super-large LMs, such as GPT-3 with 175-billion parameters, generate universal text representations? Developing universal foundation models is an ambitious goal in the field of NLP. Many studies have shown that the representations learned by these massive LMs are applicable to various NLP tasks. Unlike them, we use the user-centered recommendation as the downstream task to explore the universality of a 175-billion LM pre-trained on non-recommendation data.

Q3: Can recommender models with a 175-billion parameter LM as the item encoder easily beat the simplest ID embedding based models (IDCF), especially for warm item recommendation? IDCF is a prevailing recommendation paradigm that has dominated the recommender system (RS) community for more than a decade. It produces high-quality recommendations without relying on any item content information. However, recent studies [15, 8, 45, 14] suggest that ID features are the key barrier to achieving foundation or transferable recommender models since they are in general not shareable in practice. [58] discovered that to compete with IDCF, TCF has to retrain its text encoder on the recommendation dataset, otherwise there is still a big accuracy gap. However, their study only explored some medium-sized text encoders with around 100 million parameters. What would happen if we use the 175-billion GPT-3 as the text encoder? Would it be necessary to retrain GPT-3 to outperform IDCF for non-cold or warm item recommendation tasks?

Q4: How close is the TCF paradigm to a universal recommender model? In addition to performance, a frequently mentioned advantage of TCF is its potential transferability, which could enable cross-domain and cross-platform recommendations and establish a universal foundation model [3, 17] for the recommender system field. Thus, we aim to investigate the cross-domain recommendation capability of TCF with a text encoder of 175-billion parameters.

Q5: Will the classic TCF paradigm be replaced by a recent prompt engineering based recommendation method that utilizes ChatGPT (called ChatGPT4Rec)? With the emergence of ChatGPT, a series of recent work [9, 28, 6, 47] have leveraged the ChatGPT API and prompt to generate recommendations directly, eliminating the need for training and resulting in a highly efficient approach. An interesting question to explore is whether ChatGPT4Rec can outperform the traditional TCF paradigm in the typical recommendation setting and challenge the established TCF paradigm.

2 Background

LMs for Text. In recent years, significant advancements have been made in the development of LMs, with several landmark breakthroughs that have helped to shape the field of NLP. Word2vec, developed in 2013, revolutionized NLP by providing a scalable and efficient way of learning word embeddings from large text corpora. Since then, numerous improvements have been made to word representation models, such as GloVe [32], TextCNN [21], ELMo [33], and ULMFiT [16], etc. In 2018, the Bidirectional Encoder Representations from Transformers (BERT) model demonstrated state-of-the-art performance on a range of NLP tasks by introducing a pre-training approach based

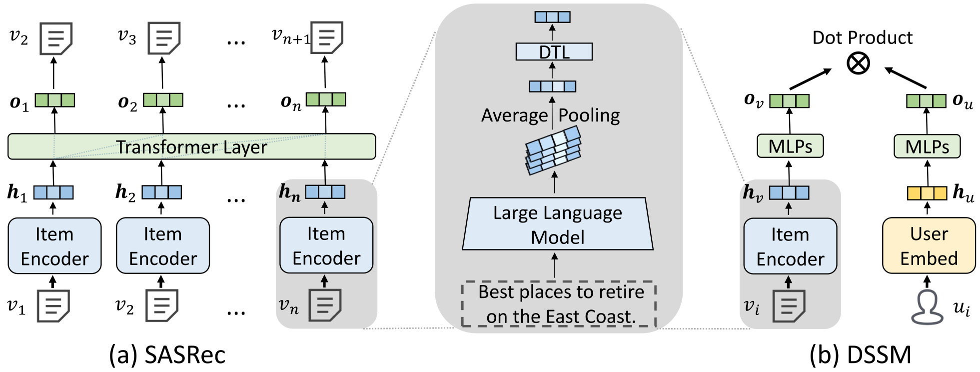

Figure 1: TCF with SASRec and DSSM as recommender backbones. The DTL block is the dense dimension transformation layers. Item or text encoder used in this study can be 175B parameters.

on a masked language modeling objective. BERT and its variants (RoBERTa [29], ALBERT [24], XLNet [53], TinyBERT [19], T5 [36], etc.) have become a dominant paradigm in the NLP community in recent years. More recently, ChatGPT, a conversational AI model based on the GPT-3 architecture, has gained significant attention due to its remarkable performance in various language tasks. Along this line, several other notable works have contributed to the advancement of LMs, including the Transformer architecture and the GPT series of models [34, 35, 4]. These advancements have not only improved the accuracy of NLP models but also opened up new avenues for research and applications in a wide range of domains outside of NLP.

LMs for Recommender Systems. Over the past decade, language models have been widely used in item recommendation tasks [17, 49], with two main lines of research in this area. The first involves using LMs to represent textual items [49, 48, 59, 58, 50], while the second involves using LMs as user encoders or recommendation backbones, such as SASRec, BERT4Rec [41], GRU4Rec [13], NextItNet [56], and Caser [42]. In this paper, we focus primarily on the first line of research. Among the various item encoders, lightweight word2vec and medium-sized BERT are the two most popular. The literature on this topic can further be classified into two categories: applying pre-extracted textual features (equivalent to free text encoder) [8, 2, 49] and end-to-end (E2E) training of text encoders [58, 52]. While E2E training typically achieves better results than using a frozen text encoder, the latter approach is much more computationally efficient than E2E training [58].

In addition, with the enormous success of ChatGPT, many recent studies have started using prompt [10, 26] techniques to guide ChatGPT in achieving personalized recommendations. This approach directly employs the ChatGPT API, without the need for separately training a model. However, this approach has some significant limitations. For example, when the number of candidate items reaches tens of thousands or even millions, ChatGPT may not be able to effectively recall and rank them.

3 Preliminary

We introduce some basic notations and describe two typical recommender paradigms: IDCF & TCF.

Definition. We define the set of users as $U = \{u_1, u_2, ..., u_m\}$ and the set of items as $V = \{v_1, v_2, ..., v_n\}$. The user-item interactions are represented by a binary matrix $R = \{r_{uv}\}$, where $r_{uv} \in \{0, 1\}$ indicates whether user u has interacted with item v.

In the standard collaborative filtering (CF) setup, we represent each user by a vector $\theta_u \in \mathbb{R}^k$ and each item by a vector $\beta_v \in \mathbb{R}^k$. The predicted interaction score between user u and item v is computed as $\hat{r}_{uv} = \theta_u^T \beta_v$. To obtain the user and item vectors, we typically optimize a loss function $l(r_{uv}, \theta_u^T \beta_v)$, where l can either be a pairwise BPR [38] loss or a cross-entropy classification loss [54].

In the popular ID-based CF (IDCF) models, θ_u and β_v, also known as userID and itemID embeddings, can be learned by backpropagating from the user-item interaction data. Following this path, various advanced recommender models have been developed. For instance, if we use a deep neural network to output the user vector θ_u and the item vector β_v, denoted by $g(u_i)$ and $h(v_i)$ respectively, the scoring function becomes $\hat{r}_{uv} = g(u_i) \cdot h(v_i)$, which is known as the two-tower DSSM model. Alternatively,

Table 1: Dataset characteristics

Dataset	#User	#Item	#Interaction	Item Example
MIND	200,000	54,246	2,920,730	Eagles fans rooting guide for Week 7. (News Title)
HM	200,000	85,019	3,160,543	Solid. White. Ladieswear. (Product Description)
Bili	50,000	22,377	723,071	Spoofs: Japanese guys fight kacoko. (Video Title)

if we represent a user by a sequence of k items that she has interacted with, the scoring function is $\hat{r}_{uv} = G(v_1, v_2, ..., v_k)^T \beta_v$, where $G(\cdot)$ is a sequential network, such as SASRec & BERT4Rec.

By utilizing a text encoder $f(v_i)$ to output item representation vectors from the description text, instead of relying on itemID embedding features, the IDCF model can be converted into the TCF model, as depicted in Figure 1. Clearly, the only difference between TCF and the typical IDCF model is in the item representation part. In contrast to IDCF, TCF has the advantage of being able to utilize both item textual content features and user-item interaction feedback data. In theory, the text encoder $f(v_i)$ can take the form of any language model, such as a shallow-layer word2vec model, a medium-sized BERT model, or a super-large GPT-3 model. The text encoder $f(v_i)$ can be either frozen or trained with the whole recommender model in an end-to-end (E2E) fashion.

However, in practice, due to computational costs, most real-world recommender systems adopt a two-stage approach where offline features are extracted beforehand (i.e., a frozen item encoder) and then incorporated into the recommender model for training. This is because joint or E2E training of text encoders usually requires significant computing power and training time.

4 Experimental Setups

4.1 Datasets, Models and Evaluation

Datasets. We evaluate TCF with LLM as text encoders on three real-world text datasets: the MIND news clicking recommendation dataset released by Microsoft [50], the HM clothing purchase dataset from the H&M[2] platform, and the Bili[3] comment dataset from an online video recommendation platform. For the MIND dataset, we represent items using their news article titles, while for the HM and Bili datasets, we utilize the corresponding titles and descriptions of products or videos to represent the items. In all three datasets, each positive user-item interaction is either a click, purchase, or comment, which served as implicit indicators of user preference.

Due to memory issues for some E2E training experiments, we constructed interaction sequences for each user by selecting their latest 23 items. We remove users with less than 5 interactions, simply because we do not consider cold user settings. After the basic pre-processing, we randomly selected 200,000 users (and their interactions) from both MIND and HM datasets, and 50,000 users from Bili.

Models and Training. To support our main arguments, we selected two representative recommendation architectures for evaluation: the two-tower DSSM model as an example of CTR (click-through rate) prediction models, and the SASRec model as an example of Transformer-style sequential models. Note we do not study other CTR prediction models, as they generally belong to the same category as DSSM, with the main difference being that many CTR models use single-tower backbone networks [57, 11, 5]. This distinction is not expected to significantly affect our conclusions [27, 63, 58]. During training, we utilize the popular batch softmax loss [54], which is widely adopted in industrial systems. For text encoders, we evaluated nine different sizes of GPT models as the text encoder, ranging from 125 million to 175 billion parameters. These GPT models were re-implemented by Meta AI and are interchangeably referred to as OPT [60]. As for hyper-parameters, we first perform a grid search for standard IDCF as a reference, After determining the optimal hyper-parameters for IDCF, we search them for TCF around these optimal values. We report details in Appendix B.

[2]https://www.kaggle.com/competitions/h-and-m-personalized-fashionrecommendations/overview

[3]URL: https://www.bilibili.com/. To create this dataset, we randomly crawled short video URLs (with durations of less than 10 minutes) from 23 vertical channels (including technology, cartoons, games, movies, food, fashion, etc.) in Bili. We then extracted the public comments on these videos as positive interactions. Finally, we chronologically combined all user interactions and removed duplicate interactions as the final dataset.

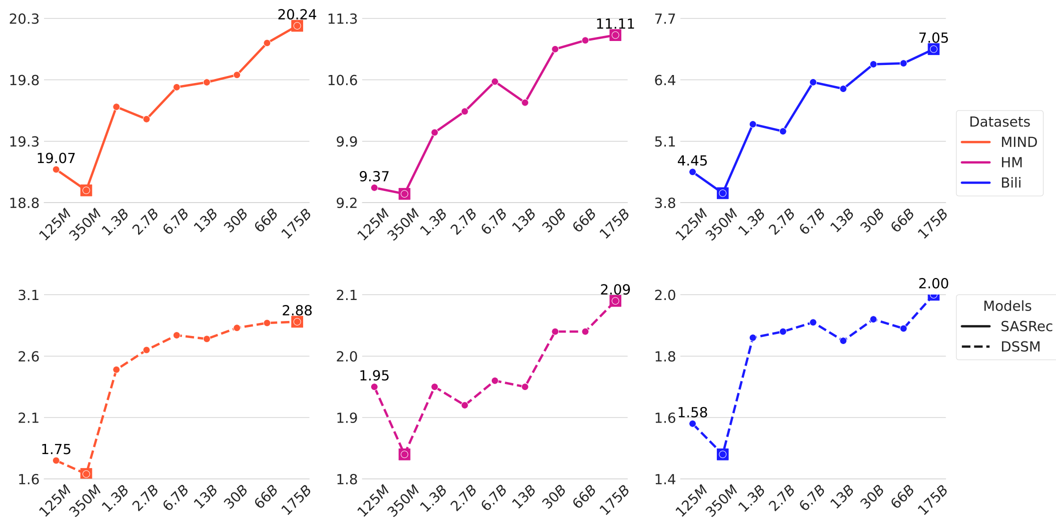

Figure 2: TCF's performance (y-axis: HR@10(%)) with 9 text encoders of increasing size (x-axis). SASRec (upper three subfigures) and DSSM (bottom three subfigures) are used as the backbone.

Evaluation. We evaluate the performance of all models using two popular top-K ranking metrics, namely HR@10 (Hit Ratio) and NDCG@10 (Normalized Discounted Cumulative Gain) [58]. NDCG@10 is reported in Appendix C for saving space. The latest user-item interaction was used for evaluation, while the second-to-last interaction was used for hyper-parameter searching, and all other interactions were used for training. All items in the pool are used for evaluation, suggested by [23].

5 Q1: Has the TCF paradigm hit a performance ceiling?

To answer Q1, we conduct experiments by increasing the size of text encoders in the TCF models, ranging from 125 million (125M) to 175 billion (175B) parameters. We use SASRec & DSSM as recommender backbones. The results are given in Figure 2. All LMs are frozen in this study.

As shown, TCF models generally improve their performance by increasing the size of their text encoders. For instance, with the SASRec as the backbone, TCF improved the recommendation accuracy from 19.07 to 20.24 on MIND, from 9.37 to 11.11 on HM, and from 4.45 to 7.05 on Bili, resulting in improvements of 6.1%, 18.6%, and 58.4%, respectively. Similar observations can also be made for the DSSM backbone. Furthermore, based on the observed performance trend, we can conclude that the TCF models' performance has not yet converged when increasing the size of their text encoders, such as from 13B to 175B. These results suggest that **(answer to Q1) the TCF model with a 175B parameter LM may not have reached its performance ceiling**. In other words, if we had an even larger LM as the text encoder, TCF's performance could potentially be further improved. This is a highly desirable property because it indicates that **using more powerful LMs (if developed in the future) as text encoders can result in higher recommendation accuracy.**

Interestingly, we find that the TCF model with the 350M parameter LM shows the worst results for all three datasets and with both DSSM and SASRec backbones, despite not being the smallest text encoder. This could happen because the scaling relationship between text encoder size and performance is not necessarily strictly linear. However, by examining the pre-training code and official documentation, we discovered that the 350M-parameter OPT was implemented with several differences compared to all other versions.[4] This provides an explanation for our results. Additionally, beyond the discussion scope of this paper, we also observe that TCF with the SASRec backbone largely outperforms that with the DSSM backbone. A similar finding has also been reported in much previous literature [58, 13, 41, 62]. One possible reason for this is that representing users using their interacted items is more effective than using solely the userID feature. Another reason could be that

[4]For instance, in all other pre-trained models, the layernorm layer is implemented before the attention layer, while in the 350M model, it is opposite. Plus, its embedding & hidden layer dimensions are also set differently.

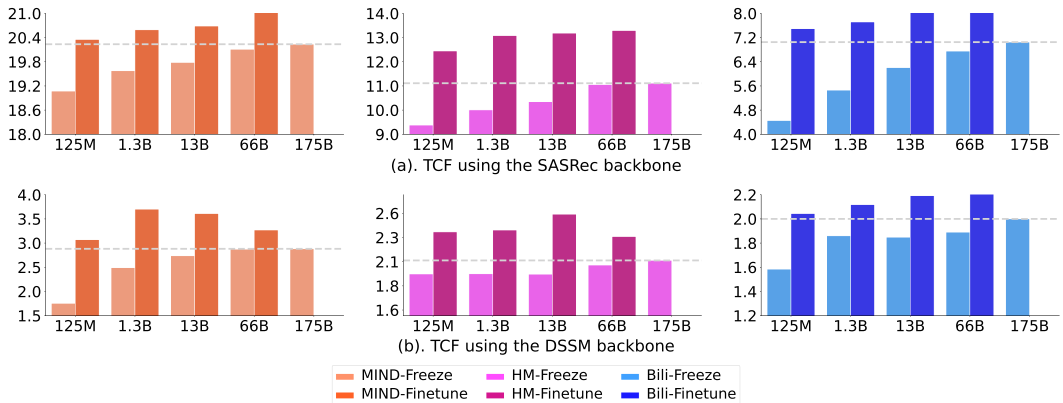

Figure 3: TCF with retrained LM vs frozen LM (y-axis: HR@10(%)), where only the top two layers are retrained. The 175B LM is not retrained due to its ultra-high computational cost.

the SASRec architecture, based on the sequence-to-sequence (seq2seq) training approach, is more powerful than the DSSM architecture, which predicts the <user, item> pair.

Table 2: Accuracy comparison (HR@10) of IDCF and TCF using the DSSM & SASRec backbones. *FR* is TCF using frozen LM, while *FT* is TCF using fine-tuned LM.

Data	SASRec			DSSM		
	IDCF	175B^{FR}	66B^{FT}	IDCF	175B^{FR}	66B^{FT}
MIND	20.05	20.24	**21.07**	**3.99**	2.83	3.27
HM	12.02	11.24	**13.29**	**6.79**	2.09	2.35
Bili	7.01	6.88	**8.15**	**2.27**	2.00	2.01

Table 3: Zero-shot recommendation accuracy (HR@10). $175\text{B}_{\text{zero}}$ means zero-shot accuracy of TCF with 175B LM. ‘train’ is to retrain TCF on these data.

Model	MIND	HM	QB
Random	0.02	0.01	0.18
$175\text{B}_{\text{zero}}$	0.13	0.39	4.30
$175\text{B}_{\text{train}}$	20.24	11.11	29.90

6 Q2: Can the 175B parameter LM achieve universal text representation?

We wonder whether a language model with 175B parameters possess a degree of universality in text encoding. Unlike the traditional NLP tasks, we examine this property using text recommendation as a downstream task. Assuming that a k-dimensional text representation β_v encoded by the 175B parameter LM is an ideal universal representation, any application involving text representation can directly choose a subset or the entire set of features from β_v by providing a weight vector w that represents the importance of these elements, i.e., $y = w^T \beta_v$. For example, in a basic matrix factorization setting, w could represent user preference weight to item features, i.e. $w = \theta_u$. If all factors of user preference can be observed by the features in β_v, we only need to learn their linear relationship. Moreover, for a perfect universal vector β_v, using a frozen representation should be just as effective as fine-tuning it on a new dataset, or even superior to fine-tuning. From an optimization perspective, using the frozen representation requires fewer training parameters than fine-tuning, and the training process is generally easier if desired item features have been fixed in advance.

Based on the above analysis, we only need to compare the frozen item representation with the fine-tuned item representation for this study. It should be noted that previous studies such as [58] have investigated this issue, but they only examined text encoders with a size of 100 million parameters. Given that the frozen representation by a 175B LM is much more powerful (see Table 5), it remains unclear whether their findings hold when the encoder is scaled up by a factor of 1000.

As shown in Figure 3, TCF models (both SASRec and DSSM) outperform their frozen versions when the text encoders are retrained on the recommendation dataset. Surprisingly, TCF with a fine-tuned 125M LM is even more powerful than the same model with a frozen 175B LM. This result potentially suggests that **(answer to Q2) even the item representation learned by an extremely large LM (e.g., GPT-3) may not result in a universal representation, at least not for the text**

Table 4: Warm item recommendation (HR@10). 20 means items < 20 interactions are removed. TCF_{175B} uses the pre-extracted features from the 175B LM. Only the SASRec backbone is reported.

Data	MIND			HM			Bili		
#Interaction	20	50	200	20	50	200	20	50	200
IDCF	20.56	20.87	23.04	13.02	14.38	18.07	7.89	9.03	15.58
TCF_{175B}	20.59	21.20	22.85	12.03	12.68	16.06	7.76	8.96	15.47

recommendation task. Another key insight is that although large LMs have revolutionized so many NLP problems, there is still a significant domain gap between recommender systems and NLP - namely, inferring user preferences looks more challenging. We suspect that the text representation even extracted from the strongest and largest LM developed in the future may not perfectly adapt to the recommender system dataset. Retraining the LM on the target recommendation data appears to be necessary for optimal results. However, from a positive perspective, since large LMs have not yet reached the performance limit, if future more powerful LMs are developed, the performance of frozen text representation may become more close to fine-tuning. For instance, we observe that SASRec with a 175B LM (compared to the 125M LM) is already very close in performance to the fine-tuned 66B LM, with relative accuracy gaps of 3.92%, 16%, 13.5% on HM, and Bili, respectively. This is a promising discovery since fine-tuning such a large LM is very challenging in practical scenarios.[5] Note although we did not retrain all parameters of the largest LM, we did evaluate the performance using medium-sized LMs (e.g., 1.3B & 13B) by optimizing all layers and the top two layers, which are comparable.

It is worth noting that the above conclusions are based on the assumption that user-item interaction feedback serves as the gold standard for the recommendation task, but this may not always be the case in practice. As a limitation, this study does not address this issue, as the entire theory of modern recommender systems is currently based on this assumption.

7 Q3: Can IDCF be easily surpassed by TCF with a 175B parameter LM?

TCF is a classical paradigm for text-based recommender systems, while IDCF is the dominant paradigm in the entire field of recommender systems. Can TCF models with a 175B parameter LM easily beat IDCF models with free and learnable item vectors? While many prior studies have reported that TCF models achieved state-of-the-art results, few have explicitly compared their models with corresponding IDCF counterparts *under the same backbone networks and experimental settings (including samplers and loss functions)*. Moreover, many of them focus on cold item setting, with fewer studies explicitly examining regular (with both cold and warm items) or popular item settings. Recently, [58] discovered that TCF can be comparable to IDCF by jointly training a 100-million parameter LM, but frozen representations still significantly underperformed. Therefore, a natural question is whether our conclusions would differ if we use a 175B parameter LM as the item encoder?

As shown in Table 2, we observe that even with the 175B parameter LM and fine-tuned 66B parameter LM, TCF is still substantially inferior to IDCF when using DSSM as the backbone. These results are consistent with [58]. As explained, the DSSM architecture and training approach are not very friendly to TCF, and both IDCF and TCF with DSSM perform worse than the seq2seq-based SASRec model. In contrast, we find that TCF with the SASRec backbone performs comparably to IDCF on MIND and Bili datasets, even when the LM encoder is frozen, as shown in Table 2 and 4. This is a significant advancement, as no previous study has *explicitly claimed* that TCF by freezing a NLP encoder can attain performance comparable to its IDCF counterparts for *warm* or *popular* item recommendation.[6] This is probably because item text encoders in prior literature, such as BERT and word2vec, are inadequate in generating effective text representations comparable to IDCF, see Table 5.

[5]Even when fine-tuning only the top two layers, it still requires 10-100 times more training time than using pre-extracted frozen features. Beyond this paper's scope, we note that two recent papers proposed promising solutions to reduce the huge computational cost of retraining LMs for the recommendation task, see [51, 52].

[6]We have simply omitted the results for cold and new item recommendation as TCF has been shown to substantially outperform IDCF in these settings in numerous literature, e.g., in [58].

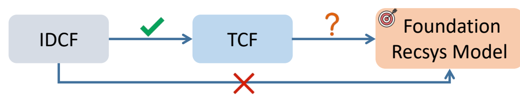

Figure 4: Route to foundation recommender models (FRM). The cross indicates that the IDCF paradigm have no chance to achieve FRM, the tick indicates that for text-centric RS, TCF can basically replace IDCF, and the question mark indicates that whether the TCF paradigm can achieve the widely recognized FRM remains still unknown.

Table 5: TCF's results (HR@10) with representative text encoders in the last 10 years. Text encoders are frozen and the SASRec backbone is used. Advances in NLP benefit RS.

Model	Date	MIND	HM	Bili
word2vec	2013	15.21	8.08	2.66
BERT_{large}	2018	18.99	9.68	3.56
T5_{XXL}	2019	19.56	9.21	4.81
OPT_{175B}	2022	20.24	11.11	7.05

The reason for the weaker performance of TCF on HM is that textual information alone is insufficient to fully represent the product item, as factors such as price and quality are also critical in enticing user clicks and purchases on HM. However, in the case of news recommendation, we can generally assume that users are primarily drawn to the textual content (i.e., titles) of items, although this may not always be the case. That is the reason we believe TCF with frozen text encoders performs on par with IDCF is surprising as IDCF can implicitly learn latent factors beyond textual features but feature representation pre-extracted from a NLP encoder cannot. Furthermore, we notice that SASRec with a fine-tuned text encoder can clearly outperform IDCF on all three datasets. However, as mentioned, such end-to-end training using a text encoder is computationally expensive, despite its effectiveness.

The answer to Q3 is that, for *text-centric* recommendation, TCF with the SASRec backbone and utilizing a 175B-parameter frozen LM can achieve similar performance to standard IDCF, even for popular item recommendation. However, even by retraining a super-large LM item encoder, TCF with a DSSM[7] backbone has little chance to compete with its corresponding IDCF. The simple IDCF still remains a highly competitive approach in the warm item recommendation setting. If the computation can be reduced, joint training of a powerful sequential recommender model (i.e., SASRec) with its text encoder can lead to markedly better results than IDCF.

8 Q4: How close is the TCF paradigm to a universal recommender model?

In this paper, we are particularly interested in comparing with the dominant IDCF paradigm. This is because ID features (including userIDs and itemIDs) are considered as a primary obstacle to the transferable or foundation recommender models due to their non-sharability [58, 15, 37, 45, 14, 8, 40]. We argue that to achieve foundation models in recommender systems may require satisfying two conditions (see Figure 4): (1) abandoning userID and itemID features, and (2) achieving effective transferability across domains and platforms. Based on the above results, we conclude that for text-centric recommender systems, TCF-based sequential recommender models can basically substitute IDCF methods. However, regarding (2), it remains uncertain whether TCF has impressive transfer learning ability, especially when its item representations are extracted from a extremely large LM.

Taking inspiration from the remarkable success of zero-shot learning in NLP, our goal is to assess the zero-shot transfer learning capability of TCF, considering that items with text features may be inherently transferable. Following [8], we first pre-train a SASRec-based TCF model with the 175B parameter frozen LM as item encoder in a large-scale text recommendation dataset[8]. We then directly evaluate the pre-trained model in the testing set of MIND, HM and QB[9]. The results, presented in Table 3, indicate that while TCF models outperform random item recommendation by achieving an accuracy improvement of 6-40x, they still fall notably short of TCF models that have been retrained on the new data. We note that user behaviors in the source Bili dataset may differ significantly from

[7] A very recent study [37] suggested that standard CTR models, such as DSSM and DeepFM [11], may be replaced by the seq2seq generative architecture, such as SASRec and GRec [55].

[8] The dataset contains 158 million user-item interactions, 8 million users, and 400,000 item text descriptions. It was also collected from the Bili platform (see Footnote 3 for data collection strategy).

[9] QQ Browser (QB) is a feed recommendation dataset from which we extracted short video titles, similar to items from Bili. It contains 5546 items 17809 users and 137979 interactions.

Table 6: ChatGPT4Rec vs TCF. *FR* & *FT* means freezing and fine-tuning LM respectively.

Data	Task 1-HR@1				Task 2-HR@10			
	Random	ChatGPT	$\text{TCF}_{175\text{B}}{}^{FR}$	$\text{TCF}_{66\text{B}}{}^{FT}$	Random	ChatGPT	$\text{TCF}_{175\text{B}}{}^{FR}$	$\text{TCF}_{66\text{B}}{}^{FT}$
MIND	25.00	25.68	96.48	96.58	10.00	9.86	97.07	97.9
HM	25.00	29.59	88.18	90.63	10.00	12.21	83.79	90.33
Bili	25.00	24.51	77.64	81.05	10.00	8.50	70.80	73.34

those of the e-commerce recommendation HM and new recommendation MIND datasets. However, it should be similar to that of QB as they are similar types of item recommendations.

Our finding is consistent to that reported in [8]: **(answer to Q4) while TCF models with large LMs do exhibit a certain degree of transfer learning capability, they still fall significantly short of being a universal recommender model, as we had initially envisioned.** For a universal recommender system model, not only should item representations be transferable, but also the matching relationship between users and items needs to be transferable. However, the matching relationship is closely related to the exposure strategy of the specific recommender system. Therefore, compared to NLP and computer vision (CV), the transferability of recommender system models is even more challenging. This also explains why, to date, there have been no pre-trained models in recommender systems that have achieved the same level of fame and recognition as BERT and ChatGPT in the NLP field. However, this does not necessarily indicate that TCF have no potential to become a universal recommender model. It will require the collective effort of the entire recommender system community, such as utilizing highly diverse and extremely large pre-training data along with more advanced training and transfer learning techniques.

9 Q5: ChatGPT4Rec vs TCF.

Beyond the TCF paradigm, building text recommender models by leveraging prompt strategies is also becoming increasingly popular [10, 46, 26, 61]. Recently, due to the tremendous success of ChatGPT, a number of preprint papers have explored the use of prompt engineering with ChatGPT for recommender systems [9, 28, 6, 47]. Here we aim to study whether prompt-based techniques on ChatGPT, referred to as ChatGPT4Rec[10], can outperform the classical TCF paradigm.

We randomly selected 1024 users from the testing sets of MIND, HM, and Bili, and created two tasks for ChatGPT. In the first task (Task 1 in Table 6), ChatGPT was asked to select the most preferred item from four candidates (one ground truth and three randomly selected items), given the user's historical interactions as a condition. The second task (Task 2 in Table 6) was to ask ChatGPT to rank the top-10 preferred items from 100 candidates (one ground truth and 99 randomly selected items, excluding all historical interactions), also provided with the user's historical interactions as input. We begin by asking ChatGPT if it understands the request, in order to ensure the quality of the prompts. Both the prompts and its answer are included in Appendix D. The results are given in Table 6, which illustrate ChatGPT's poor performance compared to TCF in typical recommendation settings. Similar bad results have also been reported in [28]. Despite that, we believe with more finely-tuned prompts, ChatGPT may have the potential for certain recommendation scenarios. Another major drawback of ChatGPT is that it cannot generate recommendations from an item pool with millions of items due to limited memory. Thus, **the answer to Q5 is that based on its current performance and limitations, ChatGPT is unable to substitute the classical TCF paradigm.**

10 Conclusion

This paper does not describe a new text recommender algorithm. Instead, it extensively explores the performance limits and several core issues of the prevailing text-based collaborative filtering (TCF) techniques. From a positive perspective, TCF has not yet reached its performance ceiling. With the further advancement of the representation capacity of NLP large models, TCF is expected to achieve

[10]We use gpt-3.5-turbo API in https://platform.openai.com/docs/models/gpt-4

better performance. However, it is regrettable that even with an item encoder of tens of billions of parameters, it still needs to be re-adapted to new data for optimal recommendations. Plus, the current cutting-edge TCF models do not exhibit strong transferability that was anticipated, indicating that building large foundation recommender models may be an even more daunting task than in NLP and CV fields. Nonetheless, TCF with text encoders of 175 billion parameters is already a significant leap forward, as it fundamentally challenges the dominant ID-based CF paradigm, which is considered the biggest obstacle to developing foundation recommender models, although not the only one.

References

[1] Rachith Aiyappa, Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. Can we trust the evaluation on chatgpt? *arXiv preprint arXiv:2303.12767*, 2023.

[2] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, 2022.

[3] 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 preprint arXiv:2108.07258*, 2021.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[6] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. Uncovering chatgpt's capabilities in recommender systems. *arXiv preprint arXiv:2305.02182*, 2023.

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

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

[9] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*, 2023.

[10] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

[11] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[14] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171, 2023.

[15] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 585–593, 2022.

[16] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[17] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item ids for recommendation foundation models. *arXiv preprint arXiv:2305.06569*, 2023.

[18] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[20] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[21] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2015.

[22] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[23] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1748–1757, 2020.

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[25] Jian Li, Jieming Zhu, Qiwei Bi, Guohao Cai, Lifeng Shang, Zhenhua Dong, Xin Jiang, and Qun Liu. Miner: Multi-interest matching network for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 343–352, 2022.

[26] Lei Li, Yongfeng Zhang, and Li Chen. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4):1–26, 2023.

[27] Xiangyang Li, Bo Chen, HuiFeng Guo, Jingjie Li, Chenxu Zhu, Xiang Long, Sujian Li, Yichao Wang, Wei Guo, Longxia Mao, et al. Inttower: the next generation of two-tower model for pre-ranking system. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3292–3301, 2022.

[28] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*, 2023.

[29] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[32] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[33] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

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

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[36] 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. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[37] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *arXiv preprint arXiv:2305.05065*, 2023.

[38] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[39] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[40] Kyuyong Shin, Hanock Kwak, Kyung-Min Kim, Minkyu Kim, Young-Jin Park, Jisu Jeong, and Seungjae Jung. One4all user representation for recommender systems in e-commerce. *arXiv preprint arXiv:2106.00573*, 2021.

[41] 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 *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[42] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[44] Congcong Wang, Paul Nulty, and David Lillis. A comparative study on word embeddings in deep learning for text classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, pages 37–46, 2020.

[45] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190*, 2022.

[46] Lei Wang and Ee-Peng Lim. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153*, 2023.

[47] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023.

[48] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 6389–6394, 2019.

[49] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1652–1656, 2021.

[50] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606.

[51] Shitao Xiao, Zheng Liu, Yingxia Shao, Tao Di, Bhuvan Middha, Fangzhao Wu, and Xing Xie. Training large-scale news recommenders with pretrained language models in the loop. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4215–4225, 2022.

[52] Yoonseok Yang, Kyu Seok Kim, Minsam Kim, and Juneyoung Park. Gram: Fast fine-tuning of pre-trained language models for content-based collaborative filtering. *arXiv preprint arXiv:2204.04179*, 2022.

[53] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[54] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 269–277, 2019.

[55] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. Future data helps training: Modeling future contexts for session-based recommendation. In *Proceedings of The Web Conference 2020*, pages 303–313, 2020.

[56] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 582–590, 2019.

[57] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. *arXiv preprint arXiv:2210.10629*, 2022.

[58] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835*, 2023.

[59] Qi Zhang, Jingjie Li, Qinglin Jia, Chuyuan Wang, Jieming Zhu, Zhaowei Wang, and Xiuqiang He. Unbert: User-news matching bert for news recommendation. In *IJCAI*, pages 3356–3362, 2021.

[60] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[61] Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. Language models as recommender systems: Evaluations and limitations. 2021.

[62] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.

[63] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. Bars: Towards open benchmarking for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2912–2923, 2022.

A Text Encoder details

Table 7: List of Large LMs and their details

Name	Model Size	Parameters	Architecture	Source
BERT	Large	340M	Encoder-only	`https://huggingface.co/bert-large-uncased`
T5Encoder	XXL	5.5B	Encoder-only	`https://huggingface.co/t5-11B`
OPT	125M	125M	Decoder-only	`https://huggingface.co/facebook/opt-125m`
	350M	350M	Decoder-only	`https://huggingface.co/facebook/opt-350m`
	1.3B	1.3B	Decoder-only	`https://huggingface.co/facebook/opt-1.3b`
	2.7B	2.7B	Decoder-only	`https://huggingface.co/facebook/opt-2.7b`
	6.7B	6.7B	Decoder-only	`https://huggingface.co/facebook/opt-6.7b`
	13B	13B	Decoder-only	`https://huggingface.co/facebook/opt-13b`
	30B	30B	Decoder-only	`https://huggingface.co/facebook/opt-30b`
	66B	66B	Decoder-only	`https://huggingface.co/facebook/opt-66b`
	175B	175B	Decoder-only	`https://github.com/facebookresearch/metaseq/tree/main/projects/OPT`

B Hyper-parameter tuning

Before tuning hyper-parameters for TCF, we grid search IDCF on each dataset as a reference. Specifically, we search for learning rates within the range of {*1e-3, 1e-4, 1e-5, 5e-5*} and hidden dimensions from {*64, 128, 256, 512, 1024*} for both DSSM and SASRec; we search batch size within {*64, 128, 256, 512*} for SASRec and {*1024, 2048, 4096*} for DSSM; we set a fixed dropout rate of 0.1, and tune the weight decay within {*0.01, 0.1*}; we search the number of Transformer layers in SASRec within {*1, 2, 3, 4*}, and the number of attention heads within {*2, 4, 8*}. After determining the optimal hyper-parameters for IDCF, we search the TCF around these optimal values with the frozen text encoder (using the 125M variant) by the same stride. To ensure a fair comparison of the scaling effect, we employ the same hyper-parameters for all TCF models with different sizes of frozen text encoder (i.e., pre-extracted features). For TCF models with E2E learning of text encoders, we kept the optimal hyper-parameters the same as those with frozen encoder, except for the learning rates. We separately tune the learning rate, as larger text encoders typically require a smaller learning rate. The details are given below. We utilize the AdamW optimizer [30] for all models.

Table 8: Optimal hyper-parameters for IDCF, including learning rate (lr), embedding size (k), batch size (bs), the number of Transformer layers (l), the number of attention heads (h), and weight decay (wd). The dimension of feed forward layer in Transformer block is $4 \times k$.

Data	SASRec						DSSM					
	lr	k	bs	l	h	wd	lr	k	bs	l	h	wd
MIND	1e-4	512	64	2	2	0.1	1e-5	256	4096	2	2	0.1
HM	1e-3	128	128	2	2	0.1	1e-4	1024	1024	2	2	0.1
Bili	1e-3	128	256	2	2	0.1	1e-3	1024	1024	2	2	0.1

Table 9: Optimal hyper-parameters for TCF with frozen text encoder.

Data	SASRec						DSSM					
	lr	k	bs	l	h	wd	lr	k	bs	l	h	wd
MIND	1e-4	512	64	2	2	0.1	1e-5	256	4096	2	2	0.1
HM	1e-4	512	64	2	2	0.1	1e-3	1024	1024	2	2	0.1
Bili	1e-3	128	64	2	2	0.1	1e-3	512	1024	2	2	0.1

Table 10: The learning rate of item encoder for TCF with E2E learning. The search range is suggested by the original paper of OPT.

Data	SASRec				DSSM			
	125M	1.3B	13B	66B	125M	1.3B	13B	66B
MIND	1e-4	1e-4	8e-5	3e-5	1e-4	1e-4	1e-4	1e-4
HM	1e-4	1e-4	1e-4	8e-5	1e-4	1e-4	1e-4	1e-4
Bili	1e-4	1e-4	3e-5	3e-5	1e-4	1e-4	1e-4	1e-4

C More results on NDCG@10

Table 11: Warm item recommendation (NDCG@10). 20 means items < 20 interactions are removed. TCF_{175B} uses the pre-extracted features from the 175B LM. Only SASRec backbone is reported.

Data	MIND			HM			Bili		
#Inter.	20	50	200	20	50	200	20	50	200
IDCF	11.36	11.47	12.71	8.47	9.35	12.07	4.41	5.01	8.30
TCF_{175B}	11.38	11.61	12.56	7.44	7.90	10.33	4.34	4.84	7.97

Table 12: Accuracy (NDCG@10) comparison of IDCF and TCF using DSSM and SASRec. *FR* represents using frozen LM, while *FT* represents using fine-tuned LM.

Data	**Metric**	SASRec			DSSM		
		ID	$TCF_{175B}{}^{FR}$	$TCF_{66B}{}^{FT}$	ID	$TCF_{175B}{}^{FR}$	$TCF_{66B}{}^{FT}$
MIND	NDCG@10	11.06	11.09	**11.77**	**1.72**	1.42	1.58
HM	NDCG@10	7.76	6.91	**8.20**	4.19	1.08	1.22
Bili	NDCG@10	3.93	3.77	**4.56**	1.12	1.01	1.06

Table 13: Zero-shot recommendation accuracy (NDCG@10). $175B_{zero}$ means zero-shot accuracy of TCF with 175B LM. 'train' is to retrain TCF on these data.

Model	Date	MIND	HM	Bili
Word2vec	2013	7.52	4.81	1.30
$BERT_{large}$	2018	10.45	6.01	1.83
$T5_{XXL}$	2019	10.72	5.50	2.54
OPT_{175B}	2022	11.17	6.88	3.95

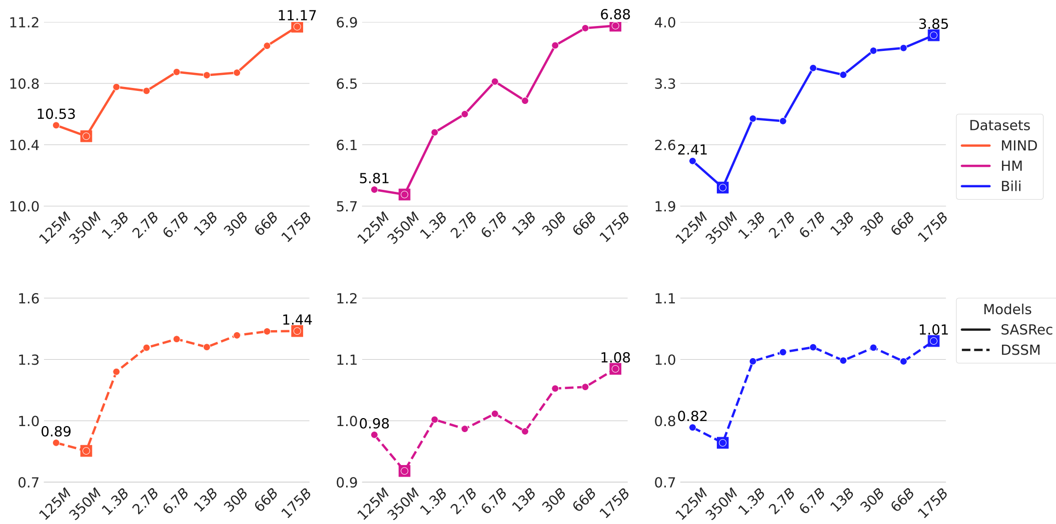

Figure 5: TCF's performance (y-axis: NDCG@10(%)) with 9 text encoders of increasing size (x-axis). SASRec (upper three subfigures) and DSSM (bottom three subfigures) are used as the backbone.

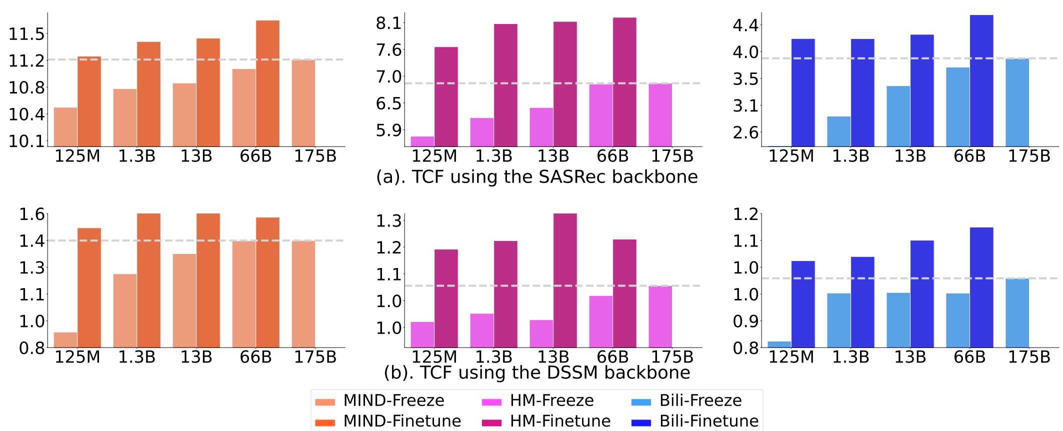

Figure 6: TCF with retrained LM vs frozen LM (y-axis: NDCG@10(%)), where only the top two layers are retrained. The 175B LM is not retrained due to its ultra-high computational cost.

D The prompts of ChatGPT4Rec

The output by ChatGPT in Figure 7 indicates that ChatGPT fully understands the recommendation request. Figure 8, 9 and 10 are prompts for ChatGPT on MIND, HM, and Bili respectively.

Do you understand our request, is it clear? "Please return a list of the top 10 most relevant video titles from the unordered candidate pool as a personalized recommendation, based on the user's viewed history, formatted as a JSON array. Do not include any titles from the user's viewed history in the recommendations. Please note that the candidate pool is not in any particular order, any ranking based on the order of items in the pool should be assumed.

User viewed history:
1.MC Lucky Gun PVP! When you fire a gun in "Desert Ash" and duel with lucky cubes!
2.Minecraft Kep1er WADADA Dance Relay Open
...
12.Central Conservatory of Music professors perform famous songs! The roof beautifully.

Unordered candidate news title pool (no order should be assumed):
- Times Youth League Ma Jiaqi
- Teach you to make homemade strawberry ice cream,with delicate taste and no ice residue,
...
- Pea Shooter: There's something wrong with the zombies here today!

Top 10 news titles from candidates pool to recommend to the user as JSON array (from highest to lowest priority, titles only): ".

Yes, I understand your request. You would like to receive a list of the top 10 most relevant video titles as personalized recommendations for a user. The recommendations should be based on the user's viewed history, and the titles should not include any videos that the user has already viewed. The candidate pool is an unordered list, and any ranking based on the order of items in the pool should be assumed.To clarify, you have provided a sample user viewed history and an unordered candidate video pool. You are requesting the top 10 video titles from the candidate pool to be recommended to the user. The recommendations should be returned as a JSON array, sorted from highest to lowest priority based on relevance. Only the titles should be included in the JSON array.Is that correct?

Figure 7: Verifying that ChatGPT understands the request.

```
By analyzing the user's click history as a personalized news recommender. Please recommend the
news titles they are most likely to click from the given 4 options according to their click
history.

User click history:
1. The Strangest State Taxes Across America
2. Manchester Town Committee chairman struck, killed by car
…
14. Julian Edelman On Josh Gordon: It's An Unfortunate Situation

Options:
A. Hannah Did 30-Minute Workouts at Home and Lost 117 Pounds in Less Than 12 Months
B. Tyler Lockett injury update: Seahawks WR hospitalized with 'pretty severe’ swelling in leg
C. Mom pushes her obnoxious boyfriend on family
D. What You Should Eat First Thing in the Morning to Help You Lose Weight

Provide a recommendation from options with no explanation (Format: [A-D]. “News Title"):
```

```
B. Tyler Lockett injury update: Seahawks WR hospitalized with 'pretty severe’ swelling in leg
```

(a) Example of Task 1 in Table 6) for MIND

```
Please return a list of the top 10 most relevant news titles from the unordered candidate pool
as a personalized recommendation, based on the user's click history, formatted as a JSON array.
Do not include any titles from the user's click history in the recommendations. Please note
that the candidate pool is not in any particular order, any ranking based on the order of
items in the pool should not be assumed!

User click history:
1. The Strangest State Taxes Across America
2. Manchester Town Committee chairman struck, killed by car
…
14. Julian Edelman On Josh Gordon: It's An Unfortunate Situation

Unordered candidate news title pool (no order should be assumed):
- Authorities: Dive boat owner faked invoices, photos to collect lionfish bounties
- 18-Year-Old Dead After Accidental Shooting In Fort Smith
…
- Wall Street eyes Washington next week

Top 10 news titles from candidates pool to recommend to the user as JSON array (from highest
to lowest priority, titles only):
```

```
[
    "He Never Smoked a Day in His Life. At 28, He Was Diagnosed With Lung Cancer.",
    "Vladimir Tarasenko to undergo surgery, out for at least 5 months",
    "Another anti-Muslim group wants to hold event at Trump's Palm Beach resort, Mar-a-Lago",
    "Unique friendship continues to grow 1 year after Camp Fire",
    "Deadline for public input on proposed HUD rule nears",
    "Former President Jimmy Carter recovering after brain surgery",
    "Twitter reacts to the return of Markelle Fultz",
    "Why aren't more women getting mammograms?",
    "Social Security: 5 smart ways to get more benefits",
    "For men with breast cancer, early detection is critical"
]
```

(b) Example of Task 2 in Table 6) for MIND

Figure 8: Prompt for MIND

```
Act as a personalized shopping recommender by analyzing a user's purchase history. Based on
their preferences, recommend the most likely product title they will click from 4 given options.

User purchase history:
1. Stripe. Light. Blue. Ladieswear. Blouse in a modal and cotton weave with a round …
2. 2. Solid. Dark. Grey. Lingeries/Tights. Microfibre thong briefs with a low waist, …
…
11. Solid. Dark. Blue. Ladieswear. Top in linen jersey with a V-neck and short sleeves.

Options:
A. All over pattern. Dusty Light. Grey. Sport. Ankle-length sports tights in fast-drying …
B. Solid. Medium Dusty. Brown. Ladies Accessories. Suede ankle boots with open toes and …
C. Solid. Light. Yellow. Lingeries/Tights. Push-up bra in lace with underwired, padded …
D. "Lace. Dusty Light. Orange. Divided. Jumper in a soft, fine-knit viscose blend with …

Provide a recommendation from options with no explanation (Format: [A-D]. “Product Title"):
```

```
A. All over pattern. Dusty Light. Grey. Sport. Ankle-length sports tights in fast-drying …
```

(a) Example of Task 1 in Table 6) for HM

```
Please return a list of the top 10 most relevant product titles from the unordered candidate
pool as a personalized recommendation, based on the user's purchase history, formatted as a
JSON array. Do not include any titles from the user's purchase history in the recommendations.
Please note that the candidate pool is not in any particular order, any ranking based on the
order of items in the pool should not be assumed!

User purchase history:
1. Stripe. Light. Blue. Ladieswear. Blouse in a modal and cotton weave with a round
2. Solid. Dark. Grey. Lingeries/Tights. Microfibre thong briefs with a low waist, …
…
11. Solid. Dark. Blue. Ladieswear. Top in linen jersey with a V-neck and short sleeves.

Unordered candidate product title pool (no order should be assumed):
- Stripe. Dark. Green. Ladieswear. Long shirt in woven fabric with a collar, buttons down
- Solid. Bright. Red. Ladieswear. Trousers in woven fabric with a high waist and an …
…
- Solid. Dark. Black. Ladies Accessories. Ankle boots in imitation leather with platform …

Top 10 news titles from candidates pool to recommend to the user as JSON array (from highest
to lowest priority, titles only):
```

```
[
  "All over pattern. Dusty Light. White. Ladieswear. Blouse in an airy weave with a",
  "All over pattern. Dusty Light. Pink. Baby Sizes 50-98. Two pairs of leggings in",
  "Solid. Bright. Metal. Ladies Accessories. Metal studs in the shape of gift bows.",
  "Solid. Light. Metal. Ladies Accessories. Narrow metal rings in various sizes and designs.",
  "Stripe. Dark. Green. Ladieswear. Long shirt in woven fabric with a collar, buttons down",
  "Solid. Bright. Red. Ladieswear. Trousers in woven fabric with a high waist and an",
  "Solid. Medium Dusty. Brown. Children Accessories, Swimwear. Boots in imitation leather …",
  "All over pattern. Medium Dusty. Brown. Menswear. Boxer shorts in a cotton weave with",
  "Solid. Dusty Light. Orange. Ladies Accessories. Toe-post mules in imitation leather …",
  "Solid. Dark. Khaki green. Divided. Fitted top in glossy jersey with a deep V-neck"
]
```

(b) Example of Task 2 in Table 6) for HM

Figure 9: Prompt for HM

```
Act as a personalized video recommender by analyzing a user's viewed history. Based on their
preferences, recommend the most likely video title they will click from 4 given options.

User viewed history:
1. MC Lucky Gun PVP! When you fire a gun in "Desert Ash" and duel with lucky cubes! minecraft
2. Kep1er WADADA Dance Relay Open
…
22. Central Conservatory of Music professors perform famous songs! The roof beautifully …

Options:
A. The finished product "hey mama" of the two-hour quick-arrangement of the art candidates …
B. Ultra-short-throw laser TV Epson 735F contrast ratio 2500000: 1 HD restore color visual …
C. Monster Hunter Rise" large expansion "Monster Hunter Rise: Dawn" promotional video
D. Piano x Yuri on Ice | Yuri on ICE

Provide a recommendation from options with no explanation (Format: [A-D]. "Video Title"):
```

```
B. Ultra-short-throw laser TV Epson 735F contrast ratio 2500000: 1 HD restore color visual …
```

(a) Example of Task 1 in Table 6) for Bili

```
Please return a list of the top 10 most relevant video titles from the unordered candidate
pool as a personalized recommendation, based on the user's viewed history, formatted as a JSON
array. Do not include any titles from the user's viewed history in the recommendations. Please
note that the candidate pool is not in any particular order, any ranking based on the order of
items in the pool should not be assumed!

User viewed history:
1. MC Lucky Gun PVP! When you fire a gun in "Desert Ash" and duel with lucky cubes! minecraft
2. Kep1er WADADA Dance Relay Open
…
22. Central Conservatory of Music professors perform famous songs! The roof beautifully …

Unordered candidate video video pool (no order should be assumed):
- [Times Youth League Ma Jiaqi
- Teach you to make homemade strawberry ice cream, with delicate taste and no ice residue, …
…
- Pea Shooter: There's something wrong with the zombies here today! [These plants are a bit …

Top 10 news titles from candidates pool to recommend to the user as JSON array (from highest
to lowest priority, titles only):
```

```
[
  "Teach you to make homemade strawberry ice cream, with delicate taste and no ice …",
  "[Is this really a human] popping john super control mechanical dance",
  "[Blade of Demon Slayer I Purgatory Kyojuro] Big Brother's 66-Second Heart Challenge",
  "[High-energy Sekiro] The most handsome Iai Kendo, performed in the game!",
  "[Undertale] Stronger Than You Response (ver. Frisk) - Animati",
  "[Hexagonal Palace Lantern]Make it with only a few pieces of paper! I don't buy lanterns …",
  "[Tutorial] Chaoshan Bamboo Oil-Paper Lantern",
  "[Quansheng Dance Studio] Stunning Four ❤ \"Mango\" Chinese Jazz Choreography MV",
  "[Final Fantasy XIV Spring Festival]Gu Raha Tia's Unknown Nursery Rhyme (model …",
  "[Polandball]The country that has been invaded by Germany for the longest time"
]
```

(b) Example of Task 2 in Table 6) for Bili

Figure 10: Prompt for Bili