M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems

Zeyu Cui*, Jianxin Ma*, Chang Zhou, Jingren Zhou, Hongxia Yang
{zeyu.czy,jason.mjx,ericzhou.zc,jingren.zhou,yang.yhx}@alibaba-inc.com
DAMO Academy, Alibaba Group

ABSTRACT

Industrial recommender systems have been growing increasingly complex, may involve *diverse domains* such as e-commerce products and user-generated contents, and can comprise *a myriad of tasks* such as retrieval, ranking, explanation generation, and even AI-assisted content production. The mainstream approach so far is to develop individual algorithms for each domain and each task. In this paper, we explore the possibility of developing a unified foundation model to support *open-ended domains and tasks* in an industrial recommender system, which may reduce the demand on downstream settings' data and can minimize the carbon footprint by avoiding training a separate model from scratch for every task. Deriving a unified foundation is challenging due to (i) the potentially unlimited set of downstream domains and tasks, and (ii) the real-world systems' emphasis on computational efficiency. We thus build our foundation upon M6, an existing large-scale industrial pretrained language model similar to GPT-3 and T5, and leverage M6's pretrained ability for sample-efficient downstream adaptation, by representing user behavior data as plain texts and converting the tasks to either language understanding or generation. To deal with a tight hardware budget, we propose an improved version of prompt tuning that outperforms fine-tuning with negligible 1% task-specific parameters, and employ techniques such as late interaction, early exiting, parameter sharing, and pruning to further reduce the inference time and the model size. We demonstrate the foundation model's versatility on a wide range of tasks such as retrieval, ranking, zero-shot recommendation, explanation generation, personalized content creation, and conversational recommendation, and manage to deploy it on both cloud servers and mobile devices.

1 INTRODUCTION

Recommender systems are indispensable to many mobile applications and have become increasingly complex nowadays. On one hand, an industrial recommender system is typically divided into *a myriad of (sub)tasks*. For example, there are tasks such as retrieval and ranking for deciding which contents to deliver to a user [6], tasks such as keyword highlighting and explanation generation for polishing the way the contents are presented [61], and even tasks such as trend forecasting and AI-assisted content production for enriching the supply of contents [18]. On the other hand, it is common for a single mobile application to have contents from *diverse domains* due to business expansion. For example, the recommendable contents in Taobao can be products, short videos or search queries, while Alipay also deals with multiple domains such as funds, mini-apps, articles, and search queries.

Conference'17, July 2017, Washington, DC, USA
2022. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The mainstream paradigm to date remains developing independent algorithms for each subtask and building separate recommender systems for each domain. This begs the question: can we establish a single foundation model that has open-ended ability, i.e., the ability to support a potentially unlimited set of downstream tasks and domains in recommender systems after minimal adaptation? A foundation model for industrial recommender systems is attractive. First, improvements in a unified foundation model can directly translate to all tasks and all domains. It has been both empirically and theoretically demonstrated that an improvement in a single subtask alone may fail to bring any improvement in the system as a whole, e.g., in the case of bias reduction [51], which can be circumvented if we optimize the foundation model to improve the whole system. Second, a pretrained foundation model can reduce the need of task-specific data. This benefits settings where data are scarce, for example, when expanding a business into a new domain or testing novel business ideas such as interactive recommendation or personalized content creation. Third, a reusable foundation model could potentially reduce the carbon footprint. The compute-intensive pretraining stage need only be executed once and adaptation to downstream tasks is computationally lightweight, unlike the existing paradigm where intensive computation is involved in every task and every domain.

However, establishing a foundation model for recommender systems is challenging. It needs to support a wide range of domains and tasks with distinct characteristics, and faces strict restrictions on inference speed, memory consumption and storage requirement imposed by real-world systems.

In this paper, we extend our existing generative pretrained language model M6 [28–30] and present M6-Rec, a foundation model for recommender systems. We convert all tasks in recommender systems into either language understanding or language generation. Specifically, we collect anonymous user behavior data from recommender systems and store them as plain texts [1] such as "A male user in Beijing, who clicked product X last night and product Y this noon, was recommended product Z and did not click it." We observe this text-based approach brings excellent zero/few-shot learning ability, thanks to the fact that M6 is pretrained on a comprehensive massive-scale corpus with well-designed objectives.

M6-Rec then makes several architectural changes to M6 for real-world deployment and for further reducing the carbon footprint. First, we propose *option tuning*, an improved variant of prompt tuning [22], in place of fine-tuning. It adds negligible task-specific parameters and makes no modification to the pre-trained model's parameters, which alleviates catastrophic forgetting and makes it possible to deploy one single model for serving multiple tasks under

[1] We focus primarily on texts. However, it is straightforward to support images by converting an image into a sequence of tokens, as in DALL-E [40] and M6 [28].

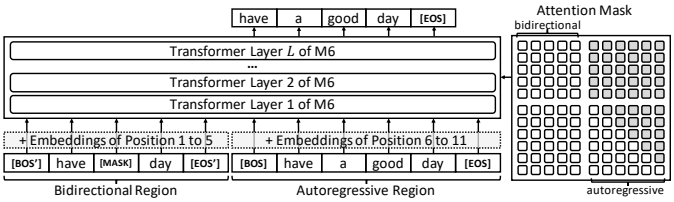


Figure 1: The text infilling objective used by the pretraining procedure of M6. [MASK] represents an undetermined number of unknown tokens. [BOS] and [EOS] mean the beginning and the end of a sentence, respectively. The autoregressive language modeling loss is imposed on the outputs of the autoregressive region, and not on the bidirectional region.

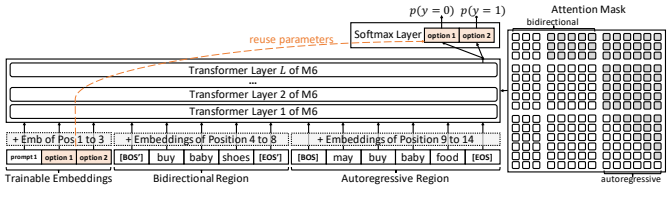


Figure 2: We propose option tuning, a variant of prompt tuning that enjoys better convergence. It reuses the last few soft prompts to serve as the parameters of the softmax classification layer, namely *soft options*. The example in the figure has three soft prompts, two of which are soft options. It optimizes only the soft prompts and the special tokens such as [EOS], while freezing the remaining parameters.

a limited hardware budget via mixed-task inference [22]. Option tuning addresses prompt tuning's slow convergence, and is able to outperform fine-tuning when combined with adapter tuning [15] despite tuning merely 1% parameters. Second, we implement a multi-segment variant of late interaction [19] when adapting M6-Rec to tasks that require low-latency real-time inference, where most of the heavy computation is pre-computed offline and cached. Finally, to make M6-Rec deployable on edge devices such as mobile phones, we further employ techniques such as parameter sharing [20], pruning [58], quantization [57], and early-exiting [16, 48] to reduce the model size and accelerate the inference speed.

In summary, our main contributions are:

- We present M6-Rec, a foundation model that not only supports sample-efficient open-domain recommendation, but also unifies all subtasks in a recommender system ranging from content supply, delivery, to presentation.
- We improve prompt tuning to outperform fine-tuning while remaining parameter-efficient, and share experiences of making M6-Rec deployable in real-world systems via late interaction, parameter sharing, pruning, and early-exiting.
- M6-Rec supports zero/few-shot learning and performs well in abundant tasks and domains, ranging from classic tasks such as retrieval and ranking, to novel use cases such as personalized product design and conversational recommendation, and is deployed on both cloud servers and edge devices.

2 RELATED WORK

In this section, we discuss the connection between M6-Rec and the existing works on recommender systems and pretrained models.

2.1 Pretraining for Recommendation

The existing works [52, 55, 56] on pretraining for recommendation are mainly about transferable user modeling, e.g., transferring knowledge from large-scale scenarios to scenarios with lesser data [60]. They have three major limitations:

First, the existing pretraining approaches for recommendation usually focus solely on behavior data and neglect non-behavior web corpora. We believe that the knowledge mined from large-scale web corpora can complement behavior data and help reduce the reliance on behavior data. For example, with the aid of web knowledge, a system may recommend turkeys when it is Thanksgiving even if we collect no click behaviors related with turkeys or Thanksgiving, i.e., in a zero-shot manner. We thus build M6-Rec upon M6 [28, 29], an existing language model pretrained on web corpora.

Second, the existing approaches typically require that pretraining and downstream tasks involve the same set of items due to the use of item IDs' embeddings, which prevents them from getting wider adoption. We thus decide to avoid item IDs in M6-Rec by using items' texts instead, and open up the possibility of open-domain recommendation where a target domain consists of unseen items.

Finally, the existing approaches deal only with scoring tasks, e.g., rating the compatibility between a user and an item, and ignore tasks that require generation, e.g., dialog-based recommendation where generating a text sentence is required. Our M6-Rec tackles this limitation by building upon M6, a generative language model.

2.2 Language Models as Foundations

Language models such as BERT [7], GPT [2, 38] and T5 [39] transfer knowledge from web corpora to downstream tasks and are thus deemed foundation models [1]. Most of them are built upon the Transformer [49] because its training paradigm is parallelizable and costs less time. There have been two recent trends in making language models more powerful foundations: (i) some researchers explore the scaling law and construct increasingly larger-scale models [11, 21, 30, 42, 46, 59, 62], and (ii) the others strive for the unification of multiple modalities [8, 28, 32, 37, 40, 44, 63].

However, the research on combining language models and user behavior data remains less developed. There are industrial works that fine-tune pretrained language models, for example based on BERT [33] or ERNIE [31, 69], to produce representations for text contents and search queries. Yet these works are limited in terms of supported domains and tasks, i.e., they target a single use case such as web searches and concern only scoring tasks, with no intention to support diverse domains or generation tasks.

The concurrent proof-of-concept work of Sileo et al. [43] shows that a pretrained model such as GPT-2 [38] does contain knowledge useful for recommendation, by using a text prompt to guide the non-finetuned GPT-2 to rank movies. We corroborate their findings and experiment with data from more domains.

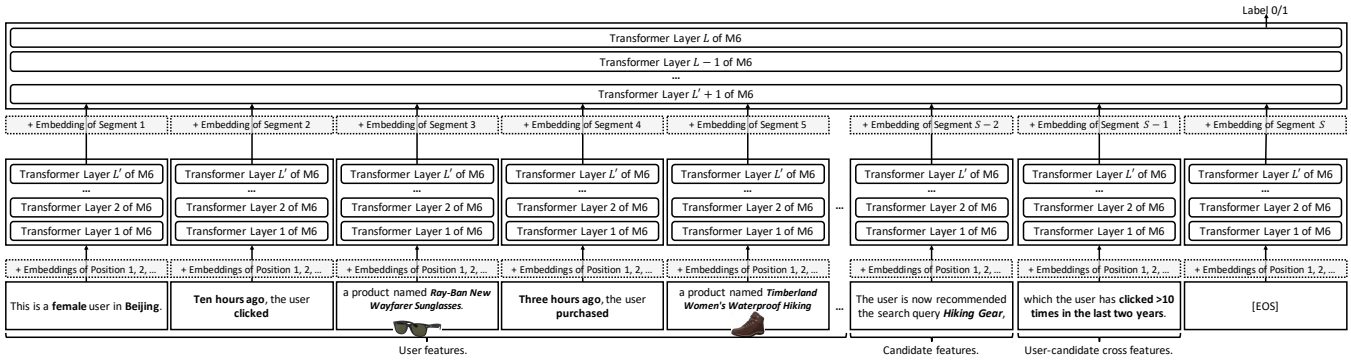


Figure 3: The figure showcases our implementation of late interaction for click-through rate (CTR) prediction. To support tasks that emphasize low-latency real-time inference, M6-Rec pre-computes and caches the first L' layers' results, while computing the last $L - L'$ layers for performing late interaction when the request arrives. A user's features as a whole usually change frequently. We thus split a user's features into finer-grained segments that are more static, e.g., by representing each clicked item as an individual segment, so that we can dynamically incorporate the user's latest activities. The special token [BOS′] **before the user feature text and the two special tokens** [EOS′][BOS] **before the candidate feature text are omitted for clarity.**

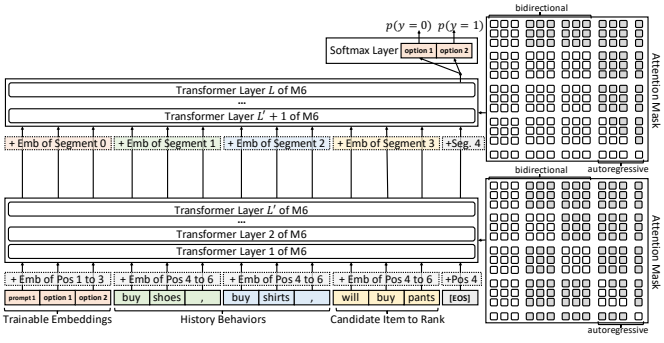


Figure 4: How to train M6-Rec using one single forward pass when option tuning and multi-segment late interaction are combined, rather than using separate forward passes for the multiple segments. This example has three soft prompts, two of which are soft options. The example user has two past behaviors, and we are predicting if the user will buy pants.

WebGPT [34] demonstrates that user behavior data are valuable for pretrained language models. Specifically, WebGPT collects human behavior data in a simplified text-based web browsing environment, and use the behavior data for teaching GPT-3 [2] how to browse the web when answering questions. Our work explores the opposite direction, i.e., we investigate if a pretrained language model can be a valuable foundation for behavior-related tasks.

2.3 Efficient Language Foundations

Low-latency inference is critical for a responsive user experience. Early exiting [16, 48] is a common technique for reducing the latency of deep models. Previous works [14, 53, 67] have adopted early exiting for BERT-like bidirectional language models, and we show in this work that it is also applicable to GPT-like autoregressive language models. Late interaction [19] is another alternative in settings where caching intermediate results are possible, which segments Transformer into a deep stage that precomputes intermediate representations and a shallow stage for performing feature interaction. Previous works [19, 69] consider the late interaction between two coarse-grained entities, and M6-Rec extends it to handle the interaction among multiple finer-grained segments.

Reducing the model size keeps hardware costs down and is mandated for resource-limited edge devices. Many strategies have been explored, e.g., parameter sharing [20], distillation [17, 41, 47, 50], pruning [5, 12, 58], and quantization [57]. Still, the existing tiny language models usually have over >10M parameters, while we estimate that it needs to be around 2M to avoid degrading the user experience when deploying a model to our users' mobile phones.

Parameter-efficient adaptation introduces as few task-specific parameters as possible when adapting a pretrained model to downstream tasks. Zero-parameter methods that use text prompts perform worse than full-model fine-tuning, but enjoy the merits of few- and zero-shot learning [2]. Other methods, such as adapters [15], prefix tuning [27], and prompt tuning [22], attempt to minimize the performance gap when reducing the number of task-specific parameters, and it is convenient to deploy one single model for serving multiple tasks simultaneously with these methods. Concurrent to our work, He et al. [13] unify adapters and prefix tuning, and discover that combining the two brings performance comparable to fine-tuning. Some researchers [45] have reported the slow convergence issue of prompt tuning. We address the issue by adding soft options. We also observe competitive performance after combining adapters and prompt tuning, even though prompt tuning cannot be precisely incorporated into He et al.'s unified framework.

3 METHOD: M6-REC

Here we briefly describe M6-Rec's backbone M6 and give examples on how M6-Rec converts recommendation-related tasks into language understanding or generation. We also propose option tuning and multi-segment late interaction for energy-efficient adaptation here, and share experiences on deploying M6-Rec to edge devices.

3.1 M6: An Industrial-Strength Backbone

M6 [28–30] is a series of visual-linguistic pretrained models. We choose M6 as our backbone, because (i) M6 supports both Chinese and English, (ii) M6 is a multi-modal model which aligns well with our plan to incorporate multi-modal features in the future, even though we primarily consider only texts in this paper for simplicity, and (iii) M6 has achieved widespread success in Alibaba Group's ecosystem when deployed into real-world businesses.

We use the base version of M6 that has around 300M parameters, unless otherwise stated. M6-base consists of a single Transformer of $L = 24$ layers, $H = 16$ attention heads, and $d = 1024$ hidden states. It uses a sequence-to-sequence attention mask similar to UniLM [10], where the source sentence falls within a region that uses the bidirectional mask and the target sentence is in a follow-up region that uses the unidirectional mask (see Figure 1).

Pretraining of M6 consists of (i) a text infilling objective [23], which benefits downstream scoring tasks by providing the ability to assess the plausibility of a text or an event, and (ii) an autoregressive language generation objective, which is critical for downstream text generation tasks. M6 implements the infilling objective by masking small spans in a sentence and autoregressively generating the unmasked sentence given the masked one, whereas the language generation objective is implemented by masking the whole sentence. We have omitted the visual modality of M6 here for clarity.

3.2 Behavior Modeling as Language Modeling

We now illustrate how M6-Rec converts all *downstream* tasks [2] into language understanding or generation tasks by representing user behavior data as natural language plain texts for fine-tuning.

Scoring tasks. The most common tasks in a recommender system are about scoring the plausibility of an event, for example, click-through rate (CTR) prediction or conversion rate (CVR) prediction where the goal is to estimate the probability of a user clicking or purchasing an item. M6-Rec expresses an example sample for CTR prediction as follows and sends it into M6's Transformer:

> [BOS′] December. Beijing, China. Cold weather. A male user in early twenties, searched "winter stuff" 23 minutes ago, clicked a product of category "jacket" named "men's lightweight warm winter hooded jacket" 19 minutes ago, clicked a product of category "sweatshirt" named "men's plus size sweatshirt stretchy pullover hoodies" 13 minutes ago, clicked … [EOS′] [BOS] The user is now recommended a product of category "boots" named "waterproof hiking shoes mens outdoor". The product has a high population-level CTR in the past 14 days, among the top 5%. The user clicked the category 4 times in the last 2 years. [EOS]

The text between [BOS′] and [EOS′] describes the user-side features, which corresponds to the first region of M6's input, i.e., the bidirectional region in Figure 1. The text between [BOS] and [EOS] describes the features related with the candidate item as well as the features related with both the user and the candidate item, which corresponds to the second region of M6's input, i.e., the autoregressive region in Figure 1. M6-Rec then uses the output vector of M6's Transformer at [EOS]'s position for summarizing the sample, sends the vector into a linear softmax classifier (which has two classes in the case of CTR prediction), and minimizes the cross-entropy loss.

Generation tasks. Content generation has become an important topic in modern recommender systems. M6-Rec uses the following plain text format to support both personalized product design [18] and explainable recommendation [61]:

> [BOS′] … [EOS′] [BOS] The user now purchases a product of category "…" named "…". Product details: … The user likes it because … [EOS]

The omitted text between [BOS′] and [EOS′] describes a user's attributes and past behaviors. The omitted text after "Product details:" is the detailed description of the product provided the seller. The product description is included here to provide basic facts for drafting an explanation text when making a recommendation decision. The omitted text after "because" is a brief sentence summarizing the user's opinion on the product, mined from the user's review using existing tools such as aspect-based sentiment analysis. The text after "because" serves as the ground-truth for explanation generation. Given a few samples of this format, M6-Rec tunes M6's Transformer using the same autoregressive language modeling loss as that of the pretraining stage shown in 1.

Generation task: explanation generation. We provide the input text prior to and including the word "because" during inference, and let the model generates the follow-up text. We can then use the generated text as the explanation for the recommendation.

Generation task: personalized product design. This task is equivalent to filling in the two blanks in the text "[BOS′] … [EOS′] [BOS] The user now purchases a product of category ___ named ___". The model thus needs to predict the category name and the product's title based on the user's attributes and past behaviors, which helps us identify the keywords that the user may potentially be interested in. Explicitly providing the category can force M6-Rec to design a product of that category. We can further send the title into a text-to-image synthesis pipeline such as M6-UFC [63].

Generation task: personalized search query generation. M6-Rec implements it as filling in the blank of "[BOS′] … [EOS′] [BOS] The user now searches the query ___" based on a user's past behaviors.

Generation task: conversational recommendation. M6-Rec supports this task by marking the speaker of each sentence:

> [BOS′] … [EOS′] [BOS] USER: Hi! SYSTEM: What kind of movie do you like? USER: I like horror movies. SYSTEM: How about *The Shining (1980)*? … [EOS]

We may put information about the user along with some basic facts between [BOS′] and [EOS′] if necessary.

Zero-shot scoring tasks. The format we described previously for scoring tasks works well if we have a few samples for training. However, it does not support zero-shot learning due to the need of training a classification layer. Fortunately, language models can estimate the likelihood of an event in a zero-shot manner, as long as the event is described in natural language. For example, to estimate

[2] We have tried mixing the user behavior texts into M6's pretraining dataset and resuming *pretraining* on the mixed dataset. However, this extra pretraining step brought only marginal gains in many cases and is thus not used for simplicity.

whether a user who clicks hiking shoes prefers trekking poles or yoga knee pads, we can construct the following two sentences:

> [BOS′] A user clicks hiking shoes [EOS′] [BOS] also clicks trekking poles [EOS]
> [BOS′] A user clicks hiking shoes [EOS′] [BOS] also clicks yoga knee pads [EOS]

M6-Rec sends each sentence into M6's Transformer and obtains the probability of each token. Let $p_1, p_2, p_1', p_2', p_3'$ be the output probabilities corresponding to token "trekking", "poles", "yoga", "knee", "pads", respectively. M6-Rec compares the plausibility of the two events by comparing $\frac{\log p_1 + \log p_2}{2}$ and $\frac{\log p_1' + \log p_2' + \log p_3'}{3}$. Here we have normalized the log likelihood by the number of tokens.

M6-Rec supports other zero-shot tasks, such as building tag-based user profiles in a zero-shot manner, in a similar vein.

Retrieval tasks. Industrial recommender systems include a task called retrieval, also known as matching or candidate generation, which is executed prior to ranking tasks such as CTR and CVR prediction [6]. Retrieval is about retrieving a small subset of candidate items from a pool of millions or even billions of items, usually via k-nearest neighbor (kNN) search after projecting the users and items into a vector space. M6-Rec feeds the information of a user into M6's Transformer using the following format:

> [BOS′] December. Beijing, China. Cold weather. A male user in early twenties, searched "winter stuff" 23 minutes ago, clicked a product of category "jacket" named "men's lightweight warm winter hooded jacket" 19 minutes ago, … [EOS′] [BOS] [EOS]

The output corresponding to [EOS′] is linearly projected into a 128-dimensional vector and l_2-normalized to form the user's vector representation $\mathbf{x}$. An item's text is fed into the model as follows:

> [BOS′] [EOS′] [BOS] A product of category "boots" named "waterproof hiking shoes mens outdoor". High CTR among the top 5%. Product details: … [EOS]

Again, the output at [EOS] is projected into a 128-dimensional vector and l_2-normalized to serve as the item's vector representation $\mathbf{y}$. We use two different learnable matrices for linearly projecting user vectors and item vectors, respectively. And l_2-normalization enables more accurate and efficient kNN search. M6-Rec uses contrastive learning to tune the model, by minimizing

$$\sum_{\langle \mathbf{x},\mathbf{y} \rangle} -\log \frac{\exp\left(\mathbf{x}^\top \mathbf{y}/\tau\right)}{\exp\left(\mathbf{x}^\top \mathbf{y}/\tau\right) + \sum_{\mathbf{y}' \in \mathcal{Y}'} \exp\left(\mathbf{x}^\top \mathbf{y}'/\tau\right)},$$

where $\langle \mathbf{x}, \mathbf{y} \rangle$ is a positive pair, for example if user $\mathbf{x}$ has clicked item $\mathbf{y}$, while $\mathcal{Y}'$ is a set of randomly sampled items. Following the literature [64], we use the items in the same mini-batches to form $\mathcal{Y}'$ and set the temperature hyperparameter τ to 0.07.

3.3 Late Interaction for Low-Latency Inference

Some components in recommender systems have a strict restraint on inference latency in order to meet users' real-time needs. Techniques such as distillation [17, 41, 47, 50] and early exiting [53, 67] can avoid high latency, but cannot fully leverage a giant foundation model's capacity since they reduce or use only a subset of parameters. We thus design *multi-segment late interaction*, an extended variant of the original late interaction [19], to cope with the issue.

We illustrate the design of multi-segment late interaction in Figure 3 using CTR prediction as an example. The core idea is to precompute and cache the computation results of the Transformer's first L' layers, and run the Transformer's last $L - L'$ layers in real time when a user request arrives. We use $L - L' \leq 3$ for low latency.

M6-Rec with late interaction does not run the first L' layers directly on the whole request when processing a request such as "Male. Clicked X. Clicked Y. Will click Z?". Rather, it chunks the request into several segments, i.e., "Male.", "Clicked X.", "Clicked Y.", and "Will click Z?". It processes the segments individually, and caches the segments' results for future reuse. Segmentation brings high reusability. For instance, when processing another request such as "Female. Clicked Y. Will click Z?", the already cached results of segment "Clicked Y." and segment "Will click Z?" are reused.

The first L' layers do not model the interaction among the segments of a request. We hence concatenate these segments' results precomputed by the first L' layers to form a single sequence, and run the last $L - L'$ layers of M6's Transformer on the said sequence to take the interaction into account. Note that the first token in every segment is always considered as if it is at position 1, because the segments are processed separately by the first L' layers. As a result, we need to superimpose a set of learnable embeddings on the first L' layers' outputs to let the last $L - L'$ layers know which are from the first segment and which are from the second segment, etc., which is similar to how position embeddings work.

3.4 Option Tuning for Efficient Adaptation

We propose option tuning, which improves upon prompt tuning [22] by addressing the slow convergence issue reported by some researchers [45]. We then propose option-adapter tuning, which augments option tuning with adapters [15] and outperforms fine-tuning despite tuning orders of magnitudes less parameters.

Prompt tuning prepends the input sequence with a predetermined number of trainable embeddings that serve as "soft prompts" in place of the discrete text prompts used by GPT-3, and tunes the soft prompts for downstream adaptation while freezing the pre-trained model's parameters. Our variant of prompt tuning reuses the last C soft prompts (depicted as "option embeddings" in Figure 2) to serve as the softmax classification layer's parameters, where C is the number of classes. Our approach empirically converges better than using a separate learnable linear classification layer. We name our approach *option tuning*, since it is in spirit providing the answer options in the soft prompts describing the downstream task.

We then combine option tuning and adapter tuning. Specifically, we add a low-rank adapter to the feedforward network (FFN) of every Transformer layer. That is, the feedforward network at the l-th layer $\mathrm{FFN}^{(l)}(\cdot)$ is replaced by $\overline{\mathrm{FFN}}^{(l)}(\cdot)$ as follows:

$$\overline{\mathrm{FFN}}^{(l)}(\mathbf{Z}) = \mathrm{FFN}^{(l)}(\mathbf{Z}) + \lambda^{(l)} \cdot \left[\sigma(\mathbf{Z}\,\mathbf{W}_1^{(l)} + \mathbf{b}_1^{(l)})\,\mathbf{W}_2^{(l)} + \mathbf{b}_2^{(l)}\right],$$

where $\mathbf{Z} \in \mathbb{R}^{B \times d}$ is the input of the FFN, B is the batch size, and $\lambda^{(l)} \in \mathbb{R}, \mathbf{W}_1^{(l)} \in \mathbb{R}^{d \times r}, \mathbf{b}_1^{(l)} \in \mathbb{R}^{1 \times r}, \mathbf{W}_2^{(l)} \in \mathbb{R}^{r \times d}, \mathbf{b}_2^{(l)} \in \mathbb{R}^{1 \times d}$. are the trainable parameters of the adapter. Typically $r \ll d$. We name this hybrid approach *option-adapter tuning*.

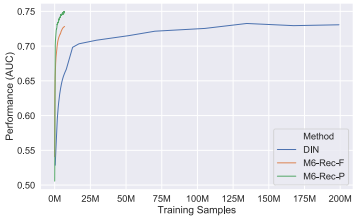


Figure 5: Performance on CTR prediction on AlipayQuery when the amount of data varies. M6-Rec-F and M6-Rec-P tune the model using fine-tuning and the variant of prompt tuning proposed by us (i.e. option-adapter tuning), respectively.

3.5 Model Compression for Mobile Devices

One of the frontiers of modern recommender system is edge computing, i.e., deploying algorithms on IoT devices or users' mobile phones to ensure responsiveness and protect user privacy. However, edge devices have varying and usually limited resources.

To reduce the model size, we use MiniLM's relation-based approach [50] to distill the 300M M6-base into a 10M tiny model which is named M6-Edge. Distillation is performed during the pretraining stage rather than the tuning stage. M6-Edge uses the same input-output schema and pretraining tasks as M6-base. M6-Edge has $L = 24$ layers, $H = 16$ attention heads, and 768 hidden states. It follows ALBERT [20] in that (i) the 24 layers share the same set of parameters and (ii) the token embeddings are linearly projected to a space of 768 dimensions from 128-dimensional embeddings.

We then perform a post-distillation pretraining step to further reduce the size and inference time of M6-Edge, using pruning and early exiting. We use gradual magnitude pruning [68] for unstructured weight pruning. For every embedding table and every linear layer, we gradually prune its weights of the lowest magnitude until reaching an 80% sparsity ratio, while the training loss is continuously optimized in the process. This reduces M6-Edge's size from 10M to 2M. When early exiting at layer k, the outputs of the k-th layer instead of the final layer are used for training and inference. Let $\mathcal{L}_k$ be the loss computed when early exiting at layer k. In this post-distillation pretraining step, we follow the literature [67] and optimize the accumulated loss $\sum_{k=1}^{L} \frac{2k}{k(k+1)} \mathcal{L}_k$ instead of just $\mathcal{L}_L$. The accumulated loss is also used for downstream tasks.

Some low-end devices can only afford one or two layers while other high-end ones may use more. Early exiting-based inference can use as many layers as possible depending on the hardware. We also perform 8-bit quantization [57] when deploying M6-Edge.

4 EXPERIMENTS

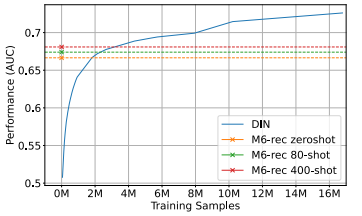
We benchmark M6-Rec on a comprehensive set of tasks, demonstrate its zero-shot learning ability, and conduct ablation studies.

4.1 Performance on Diverse Tasks

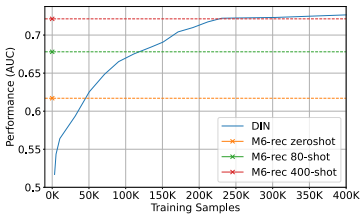
M6-Rec can outperform conventional methods in various stages.

4.1.1 Ranking. Most systems use a deep sequential model such as DIN [65] for CTR prediction. DIN stores most features values as integer IDs, learns the embeddings of the IDs, and models the interaction between features via attention modules. We compare M6-Rec with DIN. Both methods consider user behavior sequences and use text features such as item titles and category names. DIN

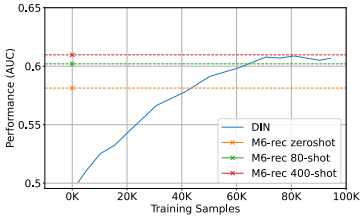
Figure 6: Personalized product design for Taobao. M6-Rec generates the ideal product title based on a user's past behaviors. The text-to-image synthesis pipeline of M6-UFC [63] is then used. M6-Rec predicts that this example user is a middle-aged housewife based on the user's past behaviors.



(a) On dataset AlipayQuery.



(b) On dataset Amazon-Movie [36].



(c) On dataset Amazon-Cloth [36].

Figure 7: The ability to perform zero-shot recommendation.

Table 1: Performance on explainable recommendation. R1-P, R1-R, R1-F, R2-P, R2-R, and R2-F denote the Precision, Recall, and F1 scores of ROUGE-1 and ROUGE-2, and ↑ indicates higher scores are better while ↓ indicates the opposite.

	Explainability			Text Quality								
Method	**FMR↑**	**FCR↑**	**DIV↓**	**USR↑**	**BLEU-1↑**	**BLEU-4↑**	**R1-P↑**	**R1-R↑**	**R1-F↑**	**R2-P↑**	**R2-R↑**	**R2-F↑**
Transformer [49]	0.10	0.01	3.26	0.00	9.71	0.59	19.68	11.94	14.11	2.10	1.39	1.55
NRT [26]	0.12	0.07	2.93	0.17	12.93	0.96	21.03	13.57	15.56	2.71	1.84	2.05
Att2Seq [9]	0.12	0.20	2.74	0.33	12.56	0.95	20.79	13.31	15.35	2.62	1.78	1.99
PETER [25]	0.12	0.21	1.75	0.29	12.77	1.17	19.81	13.80	15.23	2.80	2.08	2.20
ACMLM [35]	0.10	0.31	2.07	0.96	9.52	0.22	11.65	10.39	9.69	0.71	0.81	0.64
NETE [24]	0.71	0.19	1.93	0.57	18.76	2.46	33.87	21.43	24.81	7.58	4.77	5.46
PETER+ [25]	0.77	0.31	1.20	0.46	19.75	3.06	34,71	23.99	26.35	9.04	6.23	6.71
M6-Rec	**0.98**	**0.44**	**0.89**	**0.89**	**20.38**	**3.59**	**43.77**	**33.02**	**34.16**	**17.91**	**13.73**	**13.78**

Table 2: Performance on click-through rate (CTR) prediction, measured in terms of AUC.

		Datasets	
Method	**Method Type**	**AlipayQuery↑**	**TaoProduct↑**
DIN	ID embeddings	0.7332	0.7611
M6-Rec	Text semantics	**0.7508**	**0.7995**

Table 3: Performance on the kNN retrieval task in terms of HitRate@100, conducted on the AlipayMiniApp dataset. We report the performance on all the test cases as well as the performance on the test cases that involve unseen items, e.g., cases where users have searched new queries or cases where the candidate mini-apps do not appeared in the training set.

		Test Sets	
Method	**Method Type**	**All Items↑**	**Unseen Items↑**
YouTubeDNN	ID embeddings	54.4%	*fail*
TwinBERT	Text semantics	69.6%	49.6%
M6-Rec	Text semantics	**74.1%**	**57.0%**

Table 4: Performance on personalized content creation on dataset AlipayQuery, where the goal is to generate search queries in real time for recommending to a user.

	Metrics				
Method	**Dist-2↑**	**Dist-3↑**	**Dist-4↑**	**BLEU-2↑**	**BLEU-3↑**
KOBE	0.086	0.106	0.129	0.262	0.222
M6-Rec	**0.093**	**0.115**	**0.148**	**0.278**	**0.244**

segments each text into phases and stores the phases as integer IDs, while M6-Rec directly operates on the text. DIN additionally uses item IDs as features, which are widely deemed useful.

Table 2 shows that M6-Rec outperforms DIN on AlipayQuery and TaoProduct, two large-scale datasets collected in Alipay's search query recommender and Taobao's product recommender. Figure 5 further shows that M6-Rec needs less than a million samples to outperform the baseline that requires hundreds of millions of samples.

4.1.2 Retrieval. We then use M6-Rec for the retrieval (aka. candidate generation) of mini-apps in Alipay based on users' locations and past behaviors such as searching and visiting of mini-apps. We compare M6-Rec with TwinBERT [33] and YouTubeDNN [6].

Table 5: Performance on conversational recommendation.

	Metrics				
Method	**PPL↓**	**BLEU-2↑**	**BLEU-3↑**	**Dist-3↑**	**Dist-4↑**
Transformer	20.44	0.026	0.014	0.27	0.39
KBRD [3]	17.90	0.060	0.024	0.30	0.45
KGSF [66]	10.73	0.033	0.022	0.40	0.46
M6-Rec	**10.25**	**0.122**	0.021	**0.46**	**0.64**

Table 6: Effects of multi-segment late interaction for CTR prediction, conducted on TaoProduct.

Method	**AUC↑**	**Latency (ms)↓**
M6-Rec ($L = 24$)	0.7995	57
M6-Rec, distilled ($L = 3$)	0.7566	16
M6-Rec, late-inter ($L = 24, L - L' = 1$)	0.7299	10
M6-Rec, late-inter ($L = 24, L - L' = 3$)	**0.7731**	16

Table 7: Performance of our aggressively distilled tiny foundation model for edge deployment, i.e., M6-Edge.

		Tasks		
Method	**#Params**	**TNEWS↑**	**IFLYTEK↑**	**CSL↑**
M6-base	327M	0.598	0.631	0.852
ALBERT-zh-base	12M	0.550	0.564	0.785
M6-Edge	10M	**0.552**	**0.586**	**0.831**
ALBERT-zh-tiny	4M	0.534	0.488	0.750
M6-Edge, Pruned	2M	**0.537**	**0.559**	**0.798**

Table 8: Ablation study of our tuning methods, i.e. option tuning and option-adapter tuning, on the CLUE benchmark using the distilled 10M M6-Edge. Adding soft options to prompt tuning leads to option tuning, and option-adapter tuning is equal to prompt tuning with soft options and adapters.

		Tasks		
Tuning Method	**#Params**	**TNEWS↑**	**IFLYTEK↑**	**CSL↑**
Fine-Tuning	100%	0.544	0.575	0.829
Adapter Tuning	1%	0.542	0.574	0.825
Prompt Tuning	1%	0.534	0.531	0.760
Prompt Tuning +Soft Options	1%	0.544	0.565	0.813
Prompt Tuning +Soft Options +FFN Adapters	1%	**0.552**	**0.586**	**0.831**

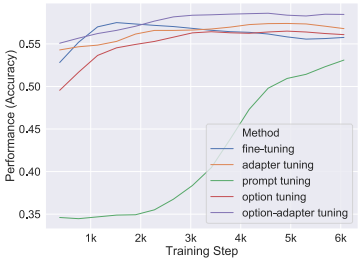


Figure 8: Effects of our modifications to prompt tuning, i.e., adding soft options and adapters. Option tuning outperforms prompt tuning, and option-adapter tuning outperforms all.

TwinBERT fine-tunes BERT for retrieval. And YouTubeDNN is a traditional baseline that maps feature values to IDs and produces user/item embeddings based on the IDs' embeddings.

Table 3 shows that M6-Rec outperforms both baselines. Moreover, when the test cases contain feature values unseen during training, such as new search queries and new mini-apps, the ID-based YouTubeDNN fails, while M6-Rec still performs well.

We also tested M6-Rec in a live experiment, where we use M6-Rec in place of a TwinBERT-like baseline for retrieving mini-apps. M6-Rec achieved over 1.0% relative improvement online in terms of CTR and was fully deployed in Alipay's system since July 2021.

4.1.3 Explanation Generation. We follow the setting used by PETER [25] strictly to measure M6-Rec's performance on generating natural language explanations for justifying the decisions made by the recommender. Table 1 shows that M6-Rec performs far better than the baselines in terms of both explainability and text quality.

4.1.4 Personalized Content Creation. We explore two use cases here: (i) generating search queries for recommending to a user, and (ii) generating new product titles based on user behaviors which tell us what types of products may be popular among a user group and help us improve the product supply.

Table 4 proves M6-Rec's ability to mine useful search queries, following the metrics used by KOBE [4]. Figure 6 then showcases some products of the clothing category designed by M6-Rec. Personalization makes it possible to discover contents catering to users' niche interests and enriches the content supply for recommendation.

4.1.5 Conversational Recommendation. We experiment with dialog-based recommendation following the settings of KBRD [3] on dataset ReDial . Table 5 shows that M6-Rec makes responses closer to the ground-truth with higher lexical diversity than the baselines.

4.2 Zero-Shot Recommendation

The unique advantage of using a pretrained language model as the foundation is that it can judge the likelihood of any event by expressing the event in natural language. To corroborate this statement, in Figure 7 we verify M6-Rec's ability to perform zero-shot ranking on three datasets of different domains, using the zero-shot method described in Subsection 3.2. Moreover, after fitting the language loss on a few samples, M6-Rec can match the performance of a traditional ID-based ranker trained on a million samples.

4.3 Effects of Late Interaction

Minimizing the latency of real-time inference is critical for some tasks such CTR prediction. In Table 6 we report the predictive performance and latency of M6-Rec after applying multi-segment late-interaction. By caching the immediate results pre-computed by the first $L' = 21$ layers, it needs to only compute the last $L - L' = 3$ layers when a request arrives. It thus enjoys a low latency similar to a distilled 3-layer student model, while incurs much less loss in terms of predictive performance than knowledge distillation.

4.4 Effects of Option-Adapter Tuning

Parameter-efficient tuning is vital to mobile phone deployment, where even one extra MB in the size can harm the user experience and serving many tasks using a shared model is preferred.

In Figure 8, we show that adding soft options to prompt tuning brings significantly better convergence. And Table 8 shows that the proposed option-adapter tuning is capable of outperforming even full-model fine-tuning, despite tuning only 1% parameters.

4.5 Effects of Compression for Mobile Devices

Beside the "cloud" rankers on our servers, our systems have additional "edge" rankers deployed on the users' mobile phones. We distill a series of tiny language models named M6-Edge to serve as the foundation for the edge rankers, as described in Subsection 3.5.

In Table 7, we show that M6-Edge outperforms public tiny language models [3] of similar scales on the Chinese benchmark CLUE [54]. An edge ranker built upon M6-Edge has been successfully deployed in Alipay, which brings an around 0.4% increase in user clicks.

5 CONCLUSION

We have proposed M6-Rec, a framework that unifies various tasks in an industrial recommender system, generalizes to open-ended

[3] https://github.com/brightmart/albert_zh

domains, and is capable of performing zero-shot learning. One future direction is to extend our framework to multimodal settings.

REFERENCES

[1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, et al. 2021. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language Models are Few-Shot Learners. In *NeurIPS 2020*.

[3] Qibin Chen, Junyang Lin, Yichang Zhang, et al. 2019. Towards Knowledge-Based Recommender Dialog System. In *EMNLP 2019*.

[4] Qibin Chen, Junyang Lin, Yichang Zhang, et al. 2019. Towards knowledge-based personalized product description generation in e-commerce. In *KDD 2019*.

[5] Tianlong Chen, Jonathan Frankle, Shiyu Chang, et al. 2020. The Lottery Ticket Hypothesis for Pre-trained BERT Networks. In *NeurIPS 2020*.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys 2016*.

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

[8] Ming Ding, Zhuoyi Yang, Wenyi Hong, et al. 2021. CogView: Mastering Text-to-Image Generation via Transformers. In *NeurIPS 2021*.

[9] Li Dong, Shaohan Huang, Furu Wei, et al. 2017. Learning to Generate Product Reviews from Attributes. In *EACL 2017*.

[10] Li Dong, Nan Yang, Wenhui Wang, et al. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS 2019*.

[11] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity.

[12] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. In *ICLR 2020*.

[13] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *ICLR 2022*.

[14] Lu Hou, Zhiqi Huang, Lifeng Shang, et al. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *NeurIPS 2020*.

[15] Edward J Hu, yelong shen, Phillip Wallis, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR 2022*.

[16] Gao Huang, Danlu Chen, Tianhong Li, et al. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *ICLR 2018*.

[17] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, et al. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In *Findings of EMNLP 2020*.

[18] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. 2017. Visually-aware fashion recommendation and design with generative image models. In *ICDM 2017*.

[19] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR 2020*.

[20] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, et al. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR 2020*.

[21] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, et al. 2021. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. In *ICLR*.

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *EMNLP 2021*.

[23] Mike Lewis, Yinhan Liu, Naman Goyal, et al. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL 2020*.

[24] Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate Neural Template Explanations for Recommendation. In *CIKM 2020*.

[25] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *ACL 2021*.

[26] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural Rating Regression with Abstractive Tips Generation for Recommendation. In *SIGIR 2017*.

[27] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL 2021*.

[28] Junyang Lin, Rui Men, An Yang, et al. 2021. M6: A Chinese Multimodal Pretrainer. arXiv:2103.00823

[29] Junyang Lin, Rui Men, An Yang, et al. 2021. M6: Multi-Modality-to-Multi-Modality Multitask Mega-transformer for Unified Pretraining. In *KDD 2021*.

[30] Junyang Lin, An Yang, Jinze Bai, et al. 2021. M6-10T: A Sharing-Delinking Paradigm for Efficient Multi-Trillion Parameter Pretraining. arXiv:2110.03888

[31] Yiding Liu, Guan Huang, Jiaxiang Liu, et al. 2021. Pre-trained Language Model for Web-scale Retrieval in Baidu Search. In *KDD 2021*.

[32] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *NeurIPS 2019*.

[33] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured BERT Models for Efficient Retrieval. arXiv:2002.06275

[34] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, et al. 2021. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332

[35] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP 2019*.

[36] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP 2019*.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML 2021*.

[38] Alec Radford, Jeff Wu, Rewon Child, et al. 2019. Language Models are Unsupervised Multitask Learners.

[39] Colin Raffel, Noam Shazeer, Adam Roberts, et al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* (2020).

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, et al. 2021. Zero-Shot Text-to-Image Generation. In *ICML 2021*.

[41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108

[42] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, et al. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053

[43] Damien Sileo, Wout Vossen, and Robbe Raymaekers. 2022. Zero-Shot Recommendation as Language Modeling. In *ECIR 2022*.

[44] Weijie Su, Xizhou Zhu, Yue Cao, et al. 2019. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. In *ICLR 2020*.

[45] YuSheng Su, Xiaozhi Wang, Yujia Qin, et al. 2021. On Transferability of Prompt Tuning for Natural Language Understanding. arXiv:2111.06719

[46] Yu Sun, Shuohuan Wang, Shikun Feng, et al. 2021. ERNIE 3.0: Large-scale Knowledge Enhanced Pre-training for Language Understanding and Generation. arXiv:2107.02137

[47] Zhiqing Sun, Hongkun Yu, Xiaodan Song, et al. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In *ACL 2020*.

[48] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *ICPR 2016*.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 2017. Attention is All you Need. In *NeurIPS 2017*.

[50] Wenhui Wang, Furu Wei, Li Dong, et al. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *NeurIPS 2020*.

[51] Xuezhi Wang, Nithum Thain, Anu Aradhana Sinha, et al. 2021. Practical Compositional Fairness: Understanding Fairness in Multi-Component Recommender Systems. In *WSDM 2021*.

[52] Chuhan Wu, Fangzhao Wu, Tao Qi, et al. 2020. PTUM: Pre-training User Model from Unlabeled User Behaviors via Self-supervision. In *Findings of EMNLP 2020*.

[53] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic Early Exiting for Accelerating BERT Inference. In *ACL 2020*.

[54] Liang Xu, Hai Hu, Xuanwei Zhang, et al. 2020. CLUE: A Chinese Language Understanding Evaluation Benchmark. In *COLING 2020*.

[55] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. *SIGIR 2020*.

[56] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, et al. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *SIGIR 2021*.

[57] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8BERT: Quantized 8Bit BERT. arXiv:1910.06188

[58] Ofir Zafrir, Ariel Larey, Guy Boudoukh, Haihao Shen, and Moshe Wasserblat. 2021. Prune Once for All: Sparse Pre-Trained Language Models. arXiv:2111.05754

[59] Wei Zeng, Xiaozhe Ren, Teng Su, et al. 2021. PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation. arXiv:2104.12369

[60] Zheni Zeng, Chaojun Xiao, Yuan Yao, et al. 2021. Knowledge Transfer via Pre-training for Recommendation: A Review and Prospect. *Frontiers in Big Data* (2021).

[61] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends in Information Retrieval* (2020).

[62] Zhengyan Zhang, Yuxian Gu, Xu Han, et al. 2021. CPM-2: Large-scale Cost-effective Pre-trained Language Models. arXiv:2106.10715

[63] Zhu Zhang, Jianxin Ma, Chang Zhou, et al. 2021. M6-UFC: Unifying Multi-Modal Controls for Conditional Image Synthesis. In *NeurIPS 2021*.

[64] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. In *KDD 2021*.

[65] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, et al. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD 2018*.

[66] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, et al. 2020. Improving Conversational Recommender Systems via Knowledge Graph based Semantic Fusion. In *KDD*.

[67] Wangchunshu Zhou, Canwen Xu, Tao Ge, et al. 2020. BERT Loses Patience: Fast and Robust Inference with Early Exit. In *NeurIPS 2020*.

[68] Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv:1710.01878
[69] Lixin Zou, Shengqiang Zhang, Hengyi Cai, et al. 2021. Pre-trained Language Model based Ranking in Baidu Search. In *KDD 2021*.

Table 9: Statistics of the datasets used for ranking and retrieval. M6-Rec uses only a small subset of the training data when experimenting with TaoProduct, AlipayQuery, and AlipayMiniApp due to the limited hardware resources available for training, even though the baselines such as DIN and YouTubeDNN do use all training data. Please refer to the related works for the information on the other tasks' datasets.

Dataset	Train	Valid	Test	#Items
TaoProduct	≈500M	≈200k	≈200k	≈30M
AlipayQuery	≈200M	≈200k	≈200k	≈5M
AlipayMiniApp	≈300M	≈200k	≈200k	≈80M
Amazon-Movie	411,946	55,168	44,362	5,419
Amazon-Cloth	93,912	11,866	11,738	17,254

A APPENDIX

A.1 Datasets

We provide here the information on the datasets collected in our real-world systems, whose statistics are listed in Table 9. Please refer to the related works for the information on the other datasets.

TaoProduct. It is collected from the ranking stage of one of the large-scale product recommender systems of the mobile application Taobao, which serves hundreds of millions of users per day. To avoid long training time, the negative samples, i.e., history records not clicked by users, are down-sampled by a factor of ten. The training data and the validation data are collected from day one to day seven, while the test set is constructed from the data of day eight. Infrequent items are filtered out, otherwise ID-based methods such as DIN will perform terribly on these cold items.

AlipayQuery. It is collected from the ranking stage of the search query recommender system of the mobile application Alipay. The negative samples are down-sampled by a factor of ten. The training data and the validation data are collected from day one to day seven, while the test set is constructed from the data of day eight. The user behaviors include the search queries and mini-apps that the users have previously interacted with, where infrequent queries and mini-apps are filtered out.

AliapyMiniApp. It is collected from a retrieval stage responsible for retrieving mini-apps in the mobile application Alipay. Mini-apps, also known as mini-programs, are lightweight apps with limited features that exist in a bigger main app that is Alipay in this case, and are developed by third-party developers instead of Alipay. The training set and the validation set are constructed from data sampled from the past thirty days, and the test set is constructed from data sampled from the following seven days. The number of items listed in Table 9 counts both search queries and mini-apps, after filtering out rare or noisy items. Due to the time gap between the training set and the test set, the test set contains a significant number of items not appeared in the training set. For example, there are many novel search queries in the test set.

A.2 Hyper-parameters

The maximum sequence length is set to 1,024. The low-rank adapters used by option-adapter tuning set the rank to $r = \frac{d}{H}$, where d is the hidden size and H is the number of attention heads. The number of soft prompts is decided such that the total number of task-specific parameters account for roughly 1% of the whole model's parameters unless otherwise specified, typically around 128. The global batch size is 128. We observe that fine-tuning and prompt-like tuning require different learning rates to achieve optimal performance. In particular, prompt-like tuning necessitates a larger learning rate, which is at least the case for M6. We thus use a learning rate of 0.00005 for fine-tuning, and use a learning rate of 0.0001 for prompt-like tuning including our option tuning and option-adapter tuning.

We train M6-Rec in a distributed manner on four machines, where each machine is equipped with eight Nvidia Tesla V100 GPUs. Each training step of a dense deep model such as M6-Rec takes far more time than that of a sparse shallow model such as DIN and YouTubeDNN. Therefore, M6-Rec uses only a small subset of the training data when running on TaoProduct, AlipayQuery, and AlipayMiniApp, even though the baselines such as DIN and YouTubeDNN do use all training data. Specifically, we sample the training data such that a training session of M6-Rec lasts no more than eighteen hours. The measurement of inference latency is also conducted on a V100 GPU, and the measurement includes the time spent on some miscellaneous overheads such as text preprocessing and embedding lookup.