Dynamic Popularity-Aware Contrastive Learning for Recommendation

Fangquan Lin* FANGQUAN.LINFQ@ALIBABA-INC.COM
Wei Jiang* ALICE.JW@ALIBABA-INC.COM
Jihai Zhang JIHAI.ZJH@ALIBABA-INC.COM
Cheng Yang CHARIS.YANGC@ALIBABA-INC.COM
Alibaba Group

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

With the development of deep learning techniques, contrastive representation learning has been increasingly employed in large-scale recommender systems. For instance, deep user-item matching models can be trained by contrasting positive and negative examples and learning discriminative user and item representations. Despite their success, the distinguishable properties of the recommender system are often ignored in existing modelling. Standard methods approximate maximum likelihood estimation on user behavior data in a manner similar to language models. Specifically, the way of model optimization corresponds to approximating the user-item pointwise mutual information, which can be regarded as eliminating the influence of global item popularity on user behavior to capture intrinsic user preference. In addition, unlike the situation in language models where word frequency is relatively stable, item popularity is constantly evolving. To address these issues, we propose a novel dynamic popularity-aware (DPA) contrastive learning method for recommendation, which consists of two key components: *i)* a dynamic negative sampling strategy is involved to enhance the user representation, *ii)* a dynamic prediction recovery is adopted by the real-time item popularity. The proposed strategy can be naturally overlaid on any contrastive learning-based matching model to more accurately capture user interest and system dynamics. Finally, the effectiveness of the proposed strategy is demonstrated through comprehensive experiments on an e-commerce scenario of Alibaba Group.

Keywords: dynamic recommendation; neural networks; contrastive learning; user interest modelling

1. Introduction

Contrastive representation learning (CRL) was first derived from the field of computer vision (Hadsell et al., 2006), and since the last decade, has also achieved remarkable results in natural language processing (He et al., 2020; Chen et al., 2020; Oord et al., 2018). Language models learn to estimate the probability of a word given a context, but computing the normalization term in the softmax output layer leads to a scalability issue. To tackle the problem of computational complexity, the learning framework based on *noise contrastive estimation* (NCE)(Gutmann and Hyvärinen, 2012; Mikolov et al., 2013) has been developed. More precisely, negative examples are drawn from some noise distribution, such as uniform

* These two authors contributed equally.

sampling, as well as importance sampling based on word frequency, and an objective function is then derived based on binary classification. When importance sampling is performed, the way of optimizing the objective function can be regarded as factorizing a word-context *pointwise mutual information* (PMI) matrix (Levy and Goldberg, 2014; Melamud et al., 2017; Ma and Collins, 2018). Intuitively, PMI indicates how frequently two events co-occur more than if they independently occur, and PMI is therefore a common mathematical proxy for assessing semantic similarity at the word level (Li et al., 2020).

When it comes to the large-scale recommender system, CRL has been employed more and more widely in recent research (Lv et al., 2019; Zhou et al., 2020; Xie et al., 2021). For instance, to recommend top items to a user from a library of millions of items, deep matching models (Covington et al., 2016; Lv et al., 2019) can be trained by contrasting positive and negative examples and learning the discriminative item representation. Nevertheless, compared to other fields of study, the recommender system retains its own properties which are often ignored in existing modelling. *i)* Dynamic: Items with a different data distribution arrive in the pool every day and the online environment varies, driven by certain underlying trends or occasional patterns. Unlike the situation in NLP where the word frequency is relatively stable in documents, in recommender systems, the items popularity can vary considerably from day to day, *e.g.*, frequent replacement of headlines in news recommendation. *ii)* Popularity-aware: user behavior is affected not only by his/her intrinsic interest, but also by attention of the general public; *e.g.*, in the e-commerce recommendation, users will be influenced by the events on the platform to purchase items on sale. In this case, when importance sampling is adopted in CRL framework, PMI can be interpreted as a user's intrinsic preference score, by eliminating the influence of the global item popularity on the user's interaction behavior, and meanwhile, the global popularity continues changing.

Given the properties above, we propose **dynamic popularity-aware (DPA)** contrastive learning for recommender system. The main contributions of this paper are as follows:

- A new perspective on the contrastive learning for dynamic recommendation is provided, by illustrating the connection between PMI and user's intrinsic interest, which can be decoupled from the global item popularity.
- System dynamics in both training and testing phases have been exploited. More precisely, we propose a dynamic negative sampling strategy to accurately capture user interest; meanwhile a dynamic prediction recovery is performed by the real-time item popularity to suppress the probability of recommending obsolete items.
- The proposed strategy can be overlaid on any state-of-the-art contrastive learning-based matching model to achieve better performance.
- Offline experimental results demonstrate the effectiveness of the proposed approach over the baseline method of deep matching on an e-commerce scenario in Alibaba.

2. Related work

2.1. Contrastive representation learning

CRL targets on learning expressive representation by comparing different examples. Since the last decade, it has been widely developed in the field of computer vision and natural language processing (Hjelm et al., 2018; Oord et al., 2018; Chen et al., 2020; He et al., 2020). For instance, the classical Word2vec model (Mikolov et al., 2013) contrasts co-occurring words with negative examples to learn the word embedding. In computer vision, the image representation can be learned in a self-supervised manner by minimizing the distance between two views of the same image (Wu et al., 2018; He et al., 2020). The contrastive loss that we investigate in this paper is the NCE loss (Gutmann and Hyvärinen, 2010, 2012), which approximates the maximum likelihood estimation by a binary classification problem. Previous work has revealed the relationship between the NCE and the PMI (Levy and Goldberg, 2014; Melamud et al., 2017). An essential element of contrastive learning lies in the sampling strategy. For example, in computer vision, augmented data can be obtained by randomly cropping and flipping (Oord et al., 2018). Despite many studies related to positive pairs (Chen et al., 2020; Logeswaran and Lee, 2018), the role of negative examples has been overlooked. Existing work (Robinson et al., 2020) observes that difficult negative examples are useful for learning expressive embedding. Importance sampling is the most common technique for constructing hard negative examples, such as using batch noise to approximate the overall frequency.

2.2. Matching models

In general, the large-scale recommender system can be coarsely divided into two stages: matching and ranking. However, most recent academic research has focused on the second stage for limited-size data, while the role of matching is highly underestimated, especially for a large e-commerce platform such as Amazon[1] and Alibaba[2]. Early works are generally based on collaborative filtering (CF) algorithms (Linden et al., 2003). Later, representation-based methods with deep learning techniques (Barkan and Koenigstein, 2016; Covington et al., 2016) have revolutionized recommendation systems dramatically. Notably, Covington *et al.* (Covington et al., 2016) propose the deep matching method for YouTube recommendation to learn both the user and item embedding in real-time. More recently, progress has been made on the graph embedding methods (Grover and Leskovec, 2016; Wu et al., 2019) and the sequential recommendation (Sun et al., 2019; Lv et al., 2019). However, these models cannot well capture the system dynamics in the recommendation as we will discuss in the following.

2.3. Dynamic recommendation

The standard recommendation algorithms consider a static system for modelling, however, the recommender system is constantly evolving as the fashion trends and popularity patterns are changing with time going by. Previous work (He et al., 2014) has verified the influence of the delay of model updating on the recommendation performance. In recent

1. https://www.amazon.com/
2. https://www.alibaba.com/

years, time-aware recommender systems have been received increasing attention (Campos et al., 2014; Zhang et al., 2017). Related studies (Xiang et al., 2010) focus on the idea that the attraction of items to users will decay with time. In addition, real-time updates of model have been tackled with online updating and stream processing (Chang et al., 2017). To model the change of a user's interest over time, Bayesian framework has been adopted in news recommendation (Liu et al., 2010) and the a time-aware mixture model has been proposed for social media system (Yin et al., 2015). Another topic of dynamic recommendation is sequential modelling (Kang and McAuley, 2018; Dong et al., 2018; Sachdeva et al., 2019), which considers the information provided by the order and position in the user behavior sequence. The sequential modelling simplifies the temporal aspects by the sequential pattern, while we emphasize more the importance of time-related modelling to provide additional information. Furthermore, we differ from the existing time-aware methods by incorporating the dynamic properties into the contrastive learning framework to provide an expressive representation of user preferences.

3. Methodology

3.1. Notations and problem statement

Let $\mathcal{U}$ and $\mathcal{V}$ denote a set of users and items respectively, where $|\mathcal{U}|$ and $|\mathcal{V}|$ are the number of users and items. Considering the user-to-item matching problem, we aim at modelling whether a user would interact with an item, given historical interaction pairs $(u, v) \in \mathcal{D}$, where the user $u \in \mathcal{U}$, the item $v \in \mathcal{V}$ and $\mathcal{D}$ is the data space. According the historical user behavior, the target of the network turns to learning the user representation matrix $E^{\mathcal{U}} = (e^u)_{u\in\mathcal{U}} \in \mathbb{R}^{d\times|\mathcal{U}|}$ and the item representation matrix $E^{\mathcal{V}} = (e^v)_{v\in\mathcal{V}} \in \mathbb{R}^{d\times|\mathcal{V}|}$ simultaneously, where d is the dimension of the embedding space. In order to retrieve top N items for a user u, the relevant scores can be calculated based on the inner product between the user embedding vector e^u and each column e^v in $E^{\mathcal{V}}$.

During the training stage, assuming that the set of parameters is denoted by θ and the relevant scores are $s_\theta(u, v)$, we aim at fitting the data to the model by minimizing full softmax loss:

$$\arg\min_{\theta} \sum_{(u,v)\in\mathcal{D}} -\log p_\theta(v|u), \quad \text{where} \quad p_\theta(v|u) = \frac{\exp s_\theta(u,v)}{\sum_{v'\in\mathcal{V}} \exp s_\theta(u,v')} = \frac{\exp(e^u \cdot e^v)}{\sum_{v'\in\mathcal{V}} \exp(e^u \cdot e^{v'})}.$$

The denominator of $p_\theta(v|u)$ sums over all possible items, which is infeasible in practice. An approximation can be achieved by sampling K negative examples according to a noise distribution $q(v)$ in the framework of contrastive learning:

$$p(y=0|u,v) = \frac{Kq(v)}{\exp s_\theta(u,v) + Kq(v)}, \quad p(y=1|u,v) = \frac{\exp s_\theta(u,v)}{\exp s_\theta(u,v) + Kq(v)},$$

where $y = 0$ indicates that v is a negative example of u while $y = 1$ for the positive case. And the NCE loss is defined as follows:

$$\mathcal{L}_{\text{NCE}} = -\sum_{(u,v)\in\mathcal{D}} \left(\log p(y=1|u,v) + K\mathbb{E}_{v'\sim q} \log p(y=0|u,v')\right). \tag{1}$$

Following the standard procedure as for language models, the maximum likelihood estimate on the user behavior data is approximated by minimizing eq. (1). Existing methods generally assume an identical pattern of user behavior between training and testing, which is however not the case in a dynamic recommender system. Compared with the volatile drifts of item popularity, user's intrinsic interest is relatively stable. Therefore, to address the problem caused by the system dynamics, we alternate the assumption of an identical behavior pattern with the more realistic one of an identical intrinsic interest. Based on that, we focus on learning and predicting the user preferences dynamically in the framework of contrastive learning.

3.2. Understanding contrastive learning: PMI and user interest

Point-wise mutual information Following the discussion in (Yang et al., 2017; Stratos, 2019), in a well-trained model where the objective function (1) achieves its minimum value, the optimal relevant score can be approximated as follows:

$$s_\theta(u, v) = \log \frac{p(v|u)}{q(v)} - \log K\,, \quad \forall u \in \mathcal{U}, \forall v \in \mathcal{V}\,. \tag{2}$$

Note that if the noise distribution is the item popularity, *i.e.*, $q(v) = p(v)$, then the first term in eq. (2) corresponds exactly to the user-item point-wise mutual information (PMI):

$$\mathrm{PMI}(u, v) = \log \frac{p(u, v)}{p(u)p(v)} = \log \frac{p(v|u)}{p(v)}\,. \tag{3}$$

Indeed, reaching the minimum value of eq. (1) is generally infeasible, since the embedding dimension is intentionally limited. Therefore, using static recommendation models as an intuitive example, learning the user and item embedding can be viewed as finding a low-rank approximation to the user-item PMI matrix (Melamud et al., 2017).

User interest PMI generally reflects the frequency with which two events co-occur more than if they occur independently (Li et al., 2020). In the user-to-item recommendation, PMI can be adopted to measure the user's intrinsic preference for an item.

To explain this, we first assume that the user behavior can be decomposed into a term decided by his/her intrinsic interest and another one influenced by the item popularity. Then, if we take a closer look at the PMI equation (3), we find that: *i)* the numerator $p(v|u)$ represents the probability that user u will interact with item v based on user's historical behavior; *ii)* and the denominator $p(v)$ represents the overall item popularity. Consequently, PMI can be interpreted as an intrinsic user preference score, by removing the influence of the global item popularity on user interaction behavior.

Recovering the prediction After learning the user and item embedding, the objective is to make a prediction of the user-item interaction $p(v|u)$. Based on eq. (2), the underlying conditional probability can be recovered as:

$$p(v|u) = \exp\left(s_{\hat{\theta}}(u, v) + \log q(v) + \log K\right)\,, \tag{4}$$

where $\hat{\theta}$ is the optimal set of parameters obtained in the training stage.

In all, in the NCE-based framework, we train a model to estimate the PMI between each user-item pair; then, in the testing phase, we derive the conditional probabilities of user-to-item matching from PMI by recovering the global popularity term.

3.3. Dynamic training & testing strategy

Time-aware modelling of item popularity Notably, the global item popularity is constantly evolving given the dynamic property. Therefore, item popularity can be modeled by a mixture distribution with respect to time. A natural and straightforward idea is to use a daily-variant modelling, which mimics the general behavior of the online e-commerce platform, *e.g.*, promotional events are usually assigned in the unit of one day.

The time-aware item popularity is captured by the conditional distribution $q_i(v) = p(v|t=i)$ for day i, where $i = 1, 2, \cdots, T$, and T is the total number of days (*i.e.*, the number of mixture components) in the training dataset. However, if the same learning and testing procedure introduced in Section 3.2 is adopted, the global item popularity in eq. (2) and eq. (4) corresponds to the marginal distribution of items: $p(v) = \sum_{i=1}^{T} p(v|t=i)p(t=i)$, where the discrete distribution $p(t=i)$ is the mixture weights which represent the proportion of items belongs to day i.

Dynamic training & testing stages To capture the dynamic property in recommender system more accurately, we propose an adaptive training and testing procedure as illustrated in Algorithm 1.

Algorithm 1: Dynamic popularity-aware contrastive learning for recommendation

---

// Dynamic training stage.
Input: user ids, the corresponding interacted items ids and the date of the interaction as a triple: $\{(u_1, v_1, t_1), (u_2, v_2, t_2) \cdots, (u_n, v_n, t_n)\}$.
Output: Estimated model parameters $\hat{\theta}$.
while $i = 1, 2, \cdots, n$ **do**
  while $j = 1, 2, \cdots, K$ **do**
    *Sampling the negative example v_j^- from the item popularity $p(v|t=t_i)$ in day t_i;
  end
  Update NCE loss:

$$\mathcal{L}_{\text{NCE}} \leftarrow \mathcal{L}_{\text{NCE}} - \left(\log p(y=1|u_i, v_i) + \sum_{j=1,2,\cdots,K} \log p(y=0|u_i, v_j^-) \right).$$

end
$\hat{\theta} = \arg\min_\theta \mathcal{L}_{\text{NCE}}$.

// Dynamic testing stage.
Input: a new triple $(u_{n+1}, v_{n+1}, t_{n+1})$.
Output: the matching probability $p(v_{n+1}|u_{n+1})$.
**Update the item popularity distribution: $p(v|t=t_{n+1})$;
Calculate the relevant score: $s_{\hat{\theta}}(u_{n+1}, v_{n+1})$;
Recovering the prediction:

$$p(v_{n+1}|u_{n+1}) = \exp\left(s_{\hat{\theta}}(u_{n+1}, v_{n+1}) + \log p(v_{n+1}|t=t_{n+1}) + \log(K)\right).$$

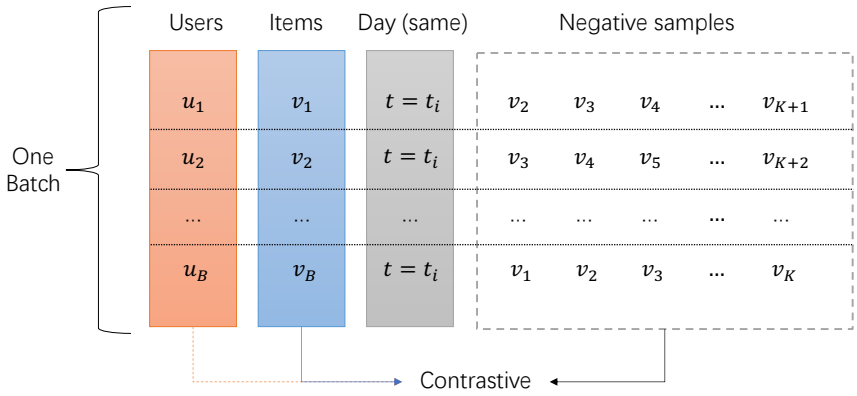

Figure 1: Intra-Batch sampling. Negative examples are constructed with positive examples of other instances in the current mini-batch.

*: To approximate the item popularity $p(v|t = t_i)$ on day t_i, we apply an Intra-Batch sampling strategy (Zhou et al., 2020), for which positive examples of other instances in the current mini-batch are used as negative examples as illustrated in Figure 1. Note that each mini-batch is composed of instances from the same day.
**: In practice, we often use the training instances of T days to predict the matching score for day $T+1$. Therefore, the item popularity distribution for day $T+1$ is usually unknown. In this situation, we use historical data to approximate the actual distribution, *e.g.*, item popularity in the past 24h is used in our experiments as described in Section 4.

In general, during the training stage, a time-dependent negative sampling strategy is adopted to capture the user's intrinsic interest; then during the testing stage, the real-time item distribution is updated and applied to recover the prediction. Figure 2 illustrates the framework of DPA strategy.

4. Experiments

In this section, we will describe in details how the experiments are conducted. We aim at answering the following research questions:
RQ1. How does the proposed DPA strategy perform compared to the baseline in the matching task?
RQ2. How does different components of DPA benefit its performance, *i.e.*, the effectiveness of dynamic negative sampling in the training stage and the dynamic prediction recovery in the testing stage?
RQ3. Does the DPA strategy really help to learn a better intrinsic user interest?

4.1. Datasets and preprocessing

Alibaba is the largest online marketplace in China. The training and evaluation framework is developed with data from the Alibaba's mobile commerce platform. To construct the offline dataset, we collect the user behavior data in one month from the 18$^{\text{th}}$ November

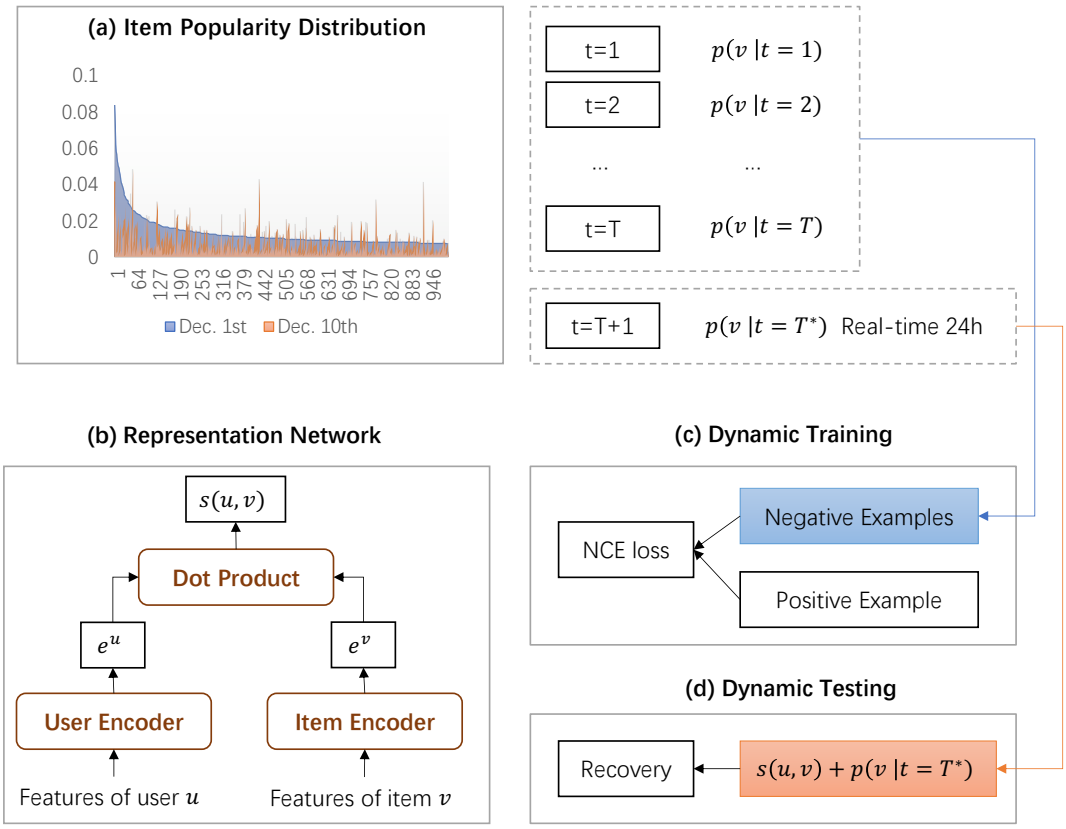

Figure 2: Framework of DPA strategy: (a) The item popularity distribution of two separated days from the Alibaba e-commerce dataset as introduced in Section 4. Their gap demonstrates the system dynamics. The x-axis represents the item id and the y-axis is the corresponding proportion in that day. Note that we only keep the most popular 1000 item ids on the 1st December due to the limit of space. (b) The architecture of representation network. (c) The dynamic training stage with time-dependant negative sampling. (d) The dynamic testing stage with prediction recovery.

to the 17th December 2014 as training set, and the data-logs on the 18th December as the testing set. The dataset has been released on the Tianchi competition website[3]. Apart from the information of user behavior, the dataset also contains the time of the behavior which facilitates the implementation of the proposed methodology. For data preprocessing, we only keep the items with more than 30 clicks and remove the cold-start users/items from the test set. The statistics of the processed offline datasets are summarized in Table 1.

We have chosen Alibaba E-commerce dataset due to its specific properties, which fit well the assumptions of DPA. *i)* The user behavior can be continuously captured by the platform and is rarely interrupted by third-party activity; *ii)* The records of user behavior are session-based, and user interest is relatively stable in one session; *iii)* User behavior is strongly influenced by the environment. Furthermore, we have demonstrated the system dynamics by comparing the item popularity of two separated days as shown in subplot (a) in Figure 2.

Table 1: Statistics of the offline Alibaba dataset.

Dataset	# users	# items	# interactions
Alibaba	19 835	82 557	5 278 193

4.2. Evaluation metrics

Following the previous work related to matching (Covington et al., 2016; Lv et al., 2019), we use the top N recall as the evaluation metrics. It is widely used to measure the predictive precision in session-based recommendation, representing the ability of coverage in the user's ground truth. For user u, it is defined as:

$$\text{Recall@}N(u) = \frac{|P_{u,N} \cap G_u|}{|G_u|},$$

where $P_{u,N}$ denotes the set of N recommended items, and G_u is the user's ground truth. For method comparison, the Recall@N for each user will be summarized by the average on the test set.

4.3. Baseline and implementation

Baseline—DeepMatch We implement DeepMatch—a neural network that embeds the user id and the item id separately, and then performs the inner product to calculate the user-item relevant score. The users and items are represented by a lookup in the embedding matrices. Subplot (b) in Figure 2 shows the model architecture. To learn the model parameters, negative examples are generated by Intra-Batch sampling and NCE loss is targeted.

In addition, we also consider POP—the simplest baseline that always recommends the most popular items.

3. https://tianchi.aliyun.com/competition/entrance/231522/introduction

Implementation of DPA strategy The DPA strategy has been employed based on the DeepMatch model. Note that, since the deep matching model and the dynamic popularity-aware procedure are independent of each other, here we use a simple but powerful baseline to verify the effect of the proposed strategy. Apart from that, the proposed method can be easily superimposed on other existing models.

Hyperparameter settings We set the embedding dimension $d = 128$ for all approaches. Adam is used as the optimizing algorithm and the learning rate is set as 0.001. The mini-batch size is set to 1024. The number of negative examples generated for each instance is 99. And 60 training epochs have been performed. All the models are implemented by TensorFlow on Machine Learning Platform for AI (PAI)[4] provided by Alibaba Cloud. All experiments are conducted on a server with GeForce GTX 1080 Ti and GPU memory 2794 MB. For more details, the code is available at https://github.com/alibaba/Dynamic-popularity-aware-recommendation/.

4.4. Experiment results and empirical analysis

(RQ1) Method comparison We first compare the performance of the proposed DPA strategy with the DeepMatch baseline. Tables 2 illustrates the performance evaluation of the investigated approaches in Recall@N with various $N \in \{1, 5, 10, 50, 100\}$.

Table 2: Performance comparison of the matching methods. Bold scores indicate the best in the method group. The improvement row is the performance of DPA-DeepMatch relative to the DeepMatch with the full softmax.

Methods	Noise distribution	Recall@1	Recall@5	Recall@10	Recall@50	Recall@100
POP	-	0.0001	0.0012	0.0019	0.0064	0.0115
POP-RealTime	-	0.0013	0.0057	0.0096	0.0361	0.0639
DeepMatch-FullSoftmax	-	0.0225	0.0728	0.1049	0.1824	0.2121
DeepMatch	Importance	0.0141	0.0504	0.0792	0.1612	0.2008
DPA-DeepMatch	Dynamic Importance	**0.0489**	**0.1042**	**0.1306**	**0.1948**	**0.2251**
Improv.		*117.30%*	*43.13%*	*24.56%*	*6.81%*	*6.15%*

Note that to recover the prediction, the real-time item popularity in the past 24h for each test instance has been used as the noise distribution. In addition, the improvement row is the performance of DPA-DeepMatch relative to the DeepMatch with full softmax. And the DeepMatch baseline has also been implemented with the standard importance sampling. Meanwhile, we consider recommendation based on both the popularity of the training set (POP) or that of the past 24 hours in real-time (POP-RealTime). Based on the experiment results, we can observe that:

- The proposed DPA outperforms the baseline method significantly in terms of all the recall metrics. For instance, DPA gains 43.13% on Recall@5 and 6.81% on Recall@50 on average against the baseline method DeepMatch with full softmax. The experiment verifies the effectiveness of the proposed DPA method in the matching task. The

4. https://www.alibabacloud.com/product/machine-learning

dynamic learning and prediction recovery in the contrastive framework have enhanced the representation and provided a more accurate recommendation.

- The recall for POP is low even with the real-time popularity, which indicates the weak dependence of the user's behavior on global popularity in this scenario. However, the DPA still leads to an strong enhancement on the recall, which demonstrates that overlaying the influence of dynamic popularity on the user's interest can make a significant effect.

- For the DeepMatch baseline, we have also conducted the full softmax without candidate sampling. Indeed, DeepMatch based on importance sampling is an approximation to the full softmax. Nevertheless, the proposed DPA-DeepMatch not only outperforms the DeepMatch with full softmax in terms of recall metrics, but also shows the computationally efficiency owing to the framework of contrastive learning.

(RQ2) Ablation study Now we analyse how the different components of DPA benefit its performance. Table 3 shows ablation experiments over the two key components in DPA to better understand their impacts. In all listed approaches, the negative examples are generated by importance sampling according to the item popularity. We consider the variant of DPA-DeepMatch with or without dynamic noise for training, and with or without dynamic item popularity recovery for testing.

Table 3: Effectiveness of dynamic negative sampling in the training stage ("Train Dynamic") and recovering the real-time item popularity in the past 24 hours in the testing stage ("Test Dynamic"). The improvement rows are the performance relative to the model M0.

Index	Train Dynamic	Test Dynamic		Recall@1	Recall@5	Recall@10	Recall@50	Recall@100
M0	✗	✗		0.0141	0.0504	0.0792	0.1612	0.2008
M1	✓	✗		0.0224	0.0708	0.1019	0.1834	0.2133
			improv.	*58.84%*	*40.46%*	*28.71%*	*13.79%*	*6.23%*
M2	✗	✓		0.0379	0.0922	0.1232	0.1986	0.2305
			improv.	*168.49%*	*83.00%*	*55.58%*	*23.19%*	*14.78%*
M3	✓	✓		0.0489	0.1042	0.1306	0.1948	0.2251
			improv.	*246.81%*	*106.71%*	*64.96%*	*20.84%*	*12.12%*

Based on the experiment results, we can conclude that:

- **Effect of dynamic sampling in training.** Model M1 outperforms M0, *e.g.*, around 28.71% improvement on Recall@10. This indicates that the dynamic negative sampling during training is useful for enhancing the representation.

- **Effect of dynamic recovery in testing.** Model M2 also outperforms M0, *e.g.*, around 55.58% improvement on Recall@10. It verifies the effectiveness of dynamic prediction recovery by the real-item item popularity during testing.

(RQ3) Capturing user's intrinsic interest Finally we focus on the case when the prediction recovery is not performed at all, *i.e.*, the prediction only reflects the user's intrinsic interest without recovering the term of global item popularity as analysed in Section 3.2. We have varied whether the dynamic negative sampling is perform, to demonstrate the ability of DPA to capture the intrinsic interest.

Table 4: Effectiveness of DPA to capture the intrinsic interest. The improvement row is the performance of model M5 relative to M4.

Index	Train Dynamic	Test	Recall@1	Recall@5	Recall@10	Recall@50	Recall@100
M4	✗	No recovery	0.0237	0.0676	0.0943	0.1704	0.2043
M5	✓	No recovery	0.0321	0.0862	0.1169	0.1891	0.2141
Improv.			*35.66%*	*27.58%*	*23.99%*	*10.99%*	*4.84%*

According to Table 4, we observe that model M5 outperforms M4, which indicates that the DPA strategy contributes to extract more accurately the user's intrinsic preferences.

5. Conclusion and future research

In this paper, we have proposed a novel strategy called DPA—dynamic popularity-aware contrastive learning for recommender system, to effectively retrieve the top N desired items from a library of items. PMI analysis is connected with user interest modelling, to provide a new perspective on the contrastive learning for dynamic recommendation. Besides, the proposed strategy can be naturally superimposed on any matching model based on contrastive learning to more accurately capture the user's intrinsic interest and system dynamics. The main contribution comes from two parts, namely dynamic noise modelling in the training stage, followed by dynamic prediction recovery in the testing stage. We have empirically demonstrated the effectiveness of the proposed strategy according to the comprehensive experiments on the Alibaba e-commerce dataset.

Concerning the future research, there are several relevant directions worth exploring. From an analytical perspective, the dynamic property can be represented as a continuous evolution of item popularity rather than considering a division per day. In application, the popularity distribution may also be different spatially as well as temporally. Therefore, the DPA strategy can be extended to heterogeneous scenarios to capture the similarities and differences among multiple mixture components.

Acknowledgments

We thank Professor Rong Jin who provided insight and expertise that greatly assisted the research. We thank Hanwei Zhang for assistance with developing the public version of code, and Ziqiang Cui and Wei Wang for comments that greatly improved the manuscript. We would also like to show our gratitude to Jingqiao Zhang for sharing his pearl of wisdom with us during this research. And we are immensely grateful to anonymous reviewers for their detailed comments.

References

Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

Pedro G Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1):67–119, 2014.

Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. Streaming recommender systems. In *Proceedings of the 26th international conference on world wide web*, pages 381–389, 2017.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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.

Disheng Dong, Xiaolin Zheng, Ruixun Zhang, and Yan Wang. Recurrent collaborative filtering for unifying general and sequential recommender. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3350–3356, 2018.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(2), 2012.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9, 2014.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE 18th International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*, 2020.

Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40, 2010.

Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*, 2018.

Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. Sdm: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2635–2643, 2019.

Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, 2018.

Oren Melamud, Ido Dagan, and Jacob Goldberger. A simple language model based on pmi matrix approximations. *arXiv preprint arXiv:1707.05266*, 2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, page 600–608, 2019.

Karl Stratos. 'noise contrastive estimation. *Rutgers Univ., Camden, NJ, USA, Tech. Notes*, 2019.

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.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–732, 2010.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. *arXiv preprint arXiv:2010.14395*, 2021.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)*, 33(3):1–44, 2015.

Fuguo Zhang, Qihua Liu, and An Zeng. Timeliness in recommender systems. *Expert Systems with Applications*, 85:270–278, 2017.

Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. Contrastive learning for debiased candidate generation in large-scale recommender systems. *arXiv preprint arXiv:2005.12964*, 2020.