A Multi-Task Learning Algorithm for Non-personalized Recommendations

Anonymous authors

Paper under double-blind review

ABSTRACT

In this paper, we introduce a multi-task learning (MTL) algorithm for recommending non-personalized videos to watch next on industrial video sharing platforms. Personalized recommendations have been studied for decades, while researches on non-personalized solutions are very rare to be seen, which still remain a huge portion in industry. As an indispensable part in recommender system, non-personalized video recommender system also faces several real-world challenges, including maintaining high relevance between source item and target items, as well as achieving multiple competing ranking objectives. To solve these, we largely extended model-based collaborative filtering algorithm by adding related candidate generation stage, Two-tower DNN structure and a multi-task learning mechanism. Compared with typical baseline solutions, our proposed algorithm can capture both linear and non-linear relationships from user-item interactions, and live experiments demonstrate that it can significantly advance the state of the art on recommendation quality.

KEYWORDS: Recommendation and Ranking, Non-personalized Recommendations, Multitask Learning, collaborative filtering, Two-tower DNN

1 INTRODUCTION

In 2018, a bomb soon became a household word that personalized recommendations in Facebook manipulated 2016 president election of United States by distributing biased articles to different people. Nowadays, users care more about personal data privacy, and more and more countries imposed laws and regulations to protect user information, such as GDPR from Europe, and in Aug in 2021, China decided that every Chinese internet company should provide users the option that they can turn off personalized recommendations. Therefore, generating non-personalized results become more important, and on video sharing platforms, which has the contents of movies, dramas, documentaries, anime and so on, the most common scenario is generating more related videos below the video that a user is currently watching. Our algorithm is designed for it and can generate high quality related results.

To generate those next videos without personalized implementation, there are some typical solutions in industry. Firstly, the most conventional one is manually looking for appropriate items by domain knowledge or expertise. Its limitations are huge human cost, poor recalled results, and unguaranteed quality. Secondly, a popular technique is using one stage of CF [1], such as ICF (item-based collaborative filtering) [2] or model-based CF [3]. CF analyzes the user-item matrix, with the value of the matrix indicates the user action on the item, such as whether $user_A$ clicks $movie_i$ or not. Then either ICF or model-based CF can produce a set of vectors representing item features. Next we can calculate the similarities between different items. Finally given a $movie_i$, we can recommend other movies by similarity score in descending order. This technique completely depends on the interactions between users and items, so for new items with few user actions, the recommended results are often very bad. Besides, sometimes it generates completely irrelevant items especially in the scenario of a video sharing platform. For example, if the platform simultaneously puts two wonderful and extremely popular movies online, which are highly attractive so that most visiting users watch them, and CF will probably get extremely high similarity score between them, because their user actions are highly overlapped. However, one video could be a comedy, the other one could be a horror film, and they have nothing in common. Such cases are very common to be seen on video platforms. Therefore, even two items that have similar user behaviors, recommending B given A in such case will cause users' complaints or other negative feedbacks. In recent years, several revised versions of CF were introduced, like CB2CF [4], it combines CB and CF to solve cold item recommendations. And like Deep Social Collaborative Filtering [5], it uses social networks to provide relevant information. These have some improvement for optimizing the single target problem.

However, on a real-word video sharing platform, the problem is much more complicated and challenging. Firstly, given a source item, the recommended target items are considered to maintain high correlation. Secondly, there are multiple competing and even conflicting objectives which we want to optimize for. For example, in addition to clicking and watching, we would like to recommend videos that people rate highly, willing to click Thumb-up on, and share with their friends, as well as have fewer number of dislikes.

Specifically, we group our multiple objectives into three categories according to industrial convention [6]: 1) engagement metrics, such as user clicks, watched time, and degree of engagement with recommended videos; 2) satisfaction metrics, such as user liking or sharing a video; 3) dissatisfaction metrics, such as disliking, low rating, negative commenting, etc.

To address these challenges, our algorithm has been introduced and it has the advantages of keeping high relevance and achieving multiple objectives at the same time. It can be divided into three stages: Candidate Generation Stage, Multi-objective Learning Stage and Merging stage, as shown in Figure 1. Given a source video, Candidates Generation Stage aims to make sure that the recalled candidates are related to the source video to some degree. We use various attributes as video profile, like video types, styles, actors, staff, tags, etc. Then we use Cosine Function to calculate similarities between items. Next we set a threshold *e* to filter outliers which are considered as irrelevant item pairs. For Multi-objective Learning Stage, there are multiple model-based CFs and two-tower DNNs executing parallel training tasks, and the Coordinating Networks (CN). When a training sample comes, it doesn't go to each parallel training module, and CN determines its right direction, such as what are these CFs and DNNs can use that sample for training. This technique can reduce noises for the entire MTL model. Each of CFs is optimizing one single objective by using alternating least squares (ALS) algorithm. Then for each objective, we extract the matrix of similarity scores between videos from trained model, which captures linear user-item relationship. Also we construct a Two-tower DNN [7, 8, 9, 10] for each objective. After training, we extract the output vector from the item-tower which can represent the input video to calculate similarity scores between videos, which have non-linear user-item relationship. After that, the merging stage generates the final result. Given a source video and a target video for objective 1, we use the maximum of their linear and non-linear similarity score. And there is a module using outputs of CN and learning the weights of all objectives. The final ranking score is a weighted sum, but we also introduce α, which indicates the manually set weight by domain expertise because different scenarios have different importance on these objectives, like some company consider time spent more important than others. Repeat this step and we get the final video-video matrix containing all ranking scores. After all, given a source video, we generate its related videos with Top *N* ranking scores.

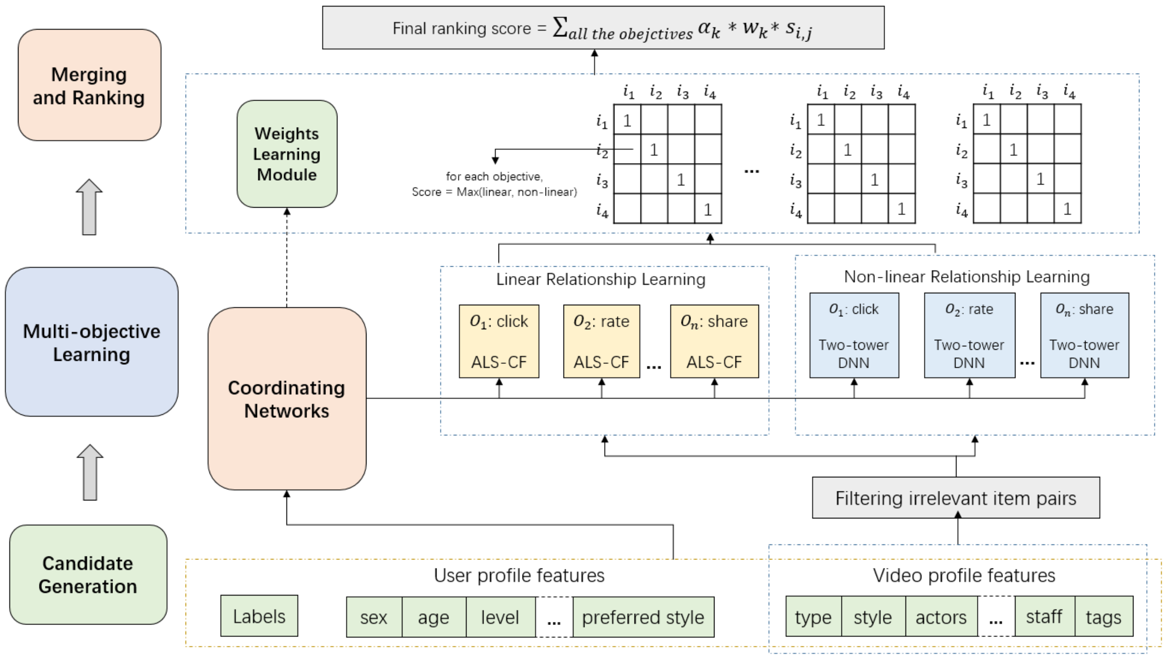
Final ranking score = $\sum_{all\ the\ obejctives} \alpha_k * w_k * s_{i,j}$
Merging and Ranking
Weights Learning Module
for each objective, Score = Max(linear, non-linear)
Multi-objective Learning
Coordinating Networks
Linear Relationship Learning
O_1: click ALS-CF
O_2: rate ALS-CF
O_n: share ALS-CF
Non-linear Relationship Learning
O_1: click Two-tower DNN
O_2: rate Two-tower DNN
O_n: share Two-tower DNN
Filtering irrelevant item pairs
Candidate Generation
User profile features
Labels
sex
age
level
preferred style
Video profile features
type
style
actors
staff
tags

Figure 1: Architecture of our multi-task learning algorithm. It has three stages. In Candidate Generation Stage, it makes sure the candidates are related to their source videos. In Multi-objective Learning Stage, it uses CFs with ALS to learn linear relationship, and uses Two-tower DNNs to learn non-linear relationship. And the Coordinating Networks control the training flow. In Merging and Ranking Stage, firstly for each objective, it uses the maximum of non-linear and linear score, and let a module learn weights of multiple objectives, then get the final ranking score by weighted sum.

To evaluate it, we design and conduct both offline and live experiments to verify the performances of: 1) maintaining high correlation, and 2) multitask learning. Comparing with baseline methods, our algorithm can highly advance the state of the art on related recommendation quality. More specifically, we see significant improvements on both engagement and satisfaction metrics, as well as a measurable reduction on dissatisfaction metric. We use BiliBili, one of the largest video sharing platforms with more than 0.2 billion monthly active users to conduct our experiments.

In summary, this have the following academic and industrial contributions:

- An end-to-end ranking system for recommending more videos for non-personalized scenarios according to the source video.
- Making use of content based attributes to generate high relevant candidates.

- Combination of model-based CF and two-tower DNN to capture both linear and non-linear relationships to enhance recommendation quality.
- Parallel learning mechanism and high performance on all objectives.
- Evaluating on a real-world large-scale video recommendation system and demonstrating significant improvements.

The rest of this paper is organized as follows: Section 2 are previous studies which are necessary for building real-world ranking systems for recommending next videos. In section 3, we provide problem descriptions for generating state-of-the-art performance. In section 4, we talk about the details of the MTL algorithm. In section 5, we describe how we design offline and live experiments to evaluate our proposed algorithm. In section 6, we conclude with our findings.

2 RELATED WORK

The problem of recommending non-personalized next videos can be formulated as returning a number of other related items given a source item. For example, after a user watches a Jet Li's movie, the system is supposed to return other Jet Li's movies, or maybe Jackie Chan's movies, because their movies have a lot in common, like Chinese actors, Action Films, Kungfu styles, Martial Art, etc. In order to generate excellent results with better user experience, our proposed algorithm contains those techniques: collaborative filtering, DNNs, multitask learning. So in this section, we provide a background to the real-world scenario, and discuss relevant works in these fields.

2.1 Non-personalized Recommendations in Industry

As we mentioned above, personalized recommendations have side effects. Some e-commerce APPs give different people different prices of a same product, some news are delivered to a certain group while others will never see them. Nowadays, many people realize that their personal secrets might be divulged or abused. In our case, nearly 10% users never login to watch videos so that they can avoid giving info to the platform. Fortunately, many laws or regulations for protecting privacy such as GDPR are imposed in recent years.

Non-personalized recommendations are indispensable in real-world recommender systems in industry. Many parts in most APPs are without personalized implementation. For example, new or non-login users whom we can't extract user features from only see the same results. If a module is going to present top K most watched videos, they are non-personalized. When a video is playing, below it we can show more related videos, and they can be non-personalized as well, and this kind of module is everywhere in Internet products. The last case is the target that this paper is talking about.

In order to generate these related videos, there are some typical solutions in industry. Firstly, the most conventional one is manually looking for appropriate items by domain knowledge or expertise. Usually each module needs several operators with very strong domain knowledge. For example, given a movie *A*, they should know its related movie list according to their judgement or quick research. We find this way does is not effective enough because of huge human cost, limited recalled results, and unguaranteed quality. Industry typically uses ICF (item based collaborative filtering) or model-based CF algorithms instead. For ICF, there are two stages: Item Similarity Computation and Prediction Computation. The first stage can be used for generating related videos. For model-based CF, such as ALS, after training, we get latent factors describing users and items respectively, so we can calculate similarity between two items by using their latent factors. These techniques work fine in some cases, but is not able to solve main issues in real-world Related Recommendations, a more functional and robust solution is needed to come into being.

2.2 Linear and Non-linear Relationship Learning

For learning linear relationship from user-item interactions, Collaborative filtering (CF) is the most popular algorithm. CF is a successful approach widely applied by lots of recommender systems [45], whose basic assumption is that people who share similar behaviors in the past tend to have similar choices in the future. Conventional CF-based methods use the ratings given to items by users for learning to make recommendations. In real-world industrial scenario, the ratings are often very sparse in many applications, which causes CF-based methods to degrade sharply in their recommendation performance. To enhance its performance, CF algorithms have been studied for decades. Some of them look into different techniques for computing item-item similarities [11]. In Recommender Systems research, algorithms are often characterized as either CF or CB (Content Based). CF algorithms are trained using a dataset of user preferences while CB algorithms are typically based on item profiles. The resulting recommended items are very different. So some researchers work on the combination of CF and CB [12]. Compared with UCF, ICF have seen verified to have better performance and be widely adopted in recommender systems in industry. However, previous versions of ICF have only considered linear and shallow relationship between user-item interactions.

In recent years, deep neural networks have been widely studied and explored on many conventional machine learning applications for representation learning. Therefore, some studies about adding DNN mechanism to CF have been introduced to enable nonlinear and higher-order relationship among items [13, 14, 15, 16]. Nevertheless, these algorithms are CF based, even adding some DNN structure. We believe directly using a pure DNN has superior effects, which can get stronger non-linear relationships from raw logs.

2.3 Multi-objective Learning for Recommendation Systems

For video recommendation, improving user experience is not a single-target problem. In social media with rich information, user preferences and item characteristics reflect in many aspects besides clicking [17], such as playing, sharing, rating, commenting and disliking etc. And each one is insufficient to reflect true user utility. For example, a user can click and play a movie but quit no longer than 1 minute; a user can finish watching a movie entirely but end up not liking it; a user can watch one movie over and over again but rate lowly. Therefore, real-world recommendation system should work well with all key objectives, achieving only one certain target but ignoring others is not acceptable.

Nowadays, more and more multitask learning algorithms come out. However, existing studies on behavior aware and multi-objective recommendation mostly focus on the module of personalized implementation [18 - 23]. Few researches and methods have been proposed about non-personalized recommendations.

3 PROBLEM DESCRIPTION

In this section, we describe the problems of recommending next videos according to the source video.

For real-world videos recommendation problems, we need to consider the following factors:

Firstly, given a source item, the recommended target items should maintain high correlation. All recommender system should improve user utility, so we should let people feel representing these results makes sense. That means, the recommended videos should be similar to the source video at some point, such as the videos are telling the same story, their main actor is the same person, or they belong to a same IP, etc. This is a very important topic in Recommender System, which is recommended results should be explainable. If the user can't understand why we recommend those movies to them, they would feel confused, end up not watching, criticizing or even never visit this platform again. Typical CF algorithms have limitations for guaranteeing this quality. Especially for new videos with fewer user interactions, known as cold-start problem, CF's performance is really bad. The first stage of our algorithm aims to solve this challenge.

Secondly, improving user utility is not about optimizing just one single target. You can't find one target to determine whether the user experience is good or bad. There are multiple competing and sometimes conflicting objectives which real-world recommender system aims to optimize for. We categorize all objectives into three groups: engagement, satisfaction and dissatisfaction objectives. And industry pays attention to all three metrics. Our model is considered as a useful framework specialized for multitask learning.

4 MODEL ARCHITECTURE

In this section, we describe three stages of the algorithm respectively in detail.

4.1 Candidate Generation Stage

Given a source video, Candidates Generation Stage aims to make sure that the recalled candidates are related to the source video to some degree. The technique of candidate generation is straightforward but crucial. This stage determines how many negative feedbacks we will receive.

We extract features from a video profile, such as video types, styles, actors, staff, tags, meta-data, content signals etc. Most of them are categorical features, so we use One-hot Encoding to transform it to a binarized sparse vector. For each original feature, we only keep Top *K* most frequent values, in order to control the size of transformed vector. Now each video has a vector with fixed size *N*, then we use Cosine Function to calculate similarities between videos, as shown in Figure 2. The calculation of cosine similarity is in Equation 1,

$$similarity = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}, \qquad (1)$$

where A_i and B_i are the *i* th elements of vector *A* and *B* respectively.

The calculated score is among [0, 1]. And we set a threshold *e* to filter outliers, which means these two videos are irrelevant and will not be calculated in subsequent stages. We can define the following generative process:

1. Extract features from the video profile,
2. One-hot Encoding, then each video has a sparse vector,
3. Use Cosine Function to calculate similarities,
4. Set a threshold *e* to filter outliers.

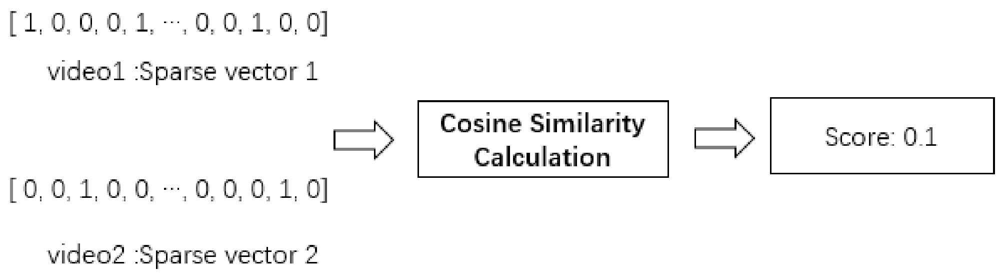

Figure 2: calculating the similarity score between two videos

4.2 Multitask Learning Stage

For multiple objectives learning stage, there are multiple CFs with ALS algorithm and 2-tower DNNs executing parallel training tasks, each of them is optimizing one single objective. Before they start to train, there is a module called Coordinating Networks, which control a sample goes to which CF or 2-tower DNN. The structure of CN is shown in Figure 3. The CN is a multiple classifier with 2 hidden layers. And the number of output neuron equals the number of ranking objectives. Each output neuron represents the probability that the input sample belongs to this class. A sample only with a probability higher than *t* to a certain objective can go its subsequent CF or 2-tower DNN. CN uses data from all classes for training, so it can reduce noises for multi-task learning. For example, when we want to optimize clicks, we have some bad cases like a user clicks a video but quits immediately, and a single task model can’t predict this case as negative, but CN is more robust and can give it a low score for objective click. So samples the subsequent CF or 2-tower DNN used have higher quality that can reflect better user experience.

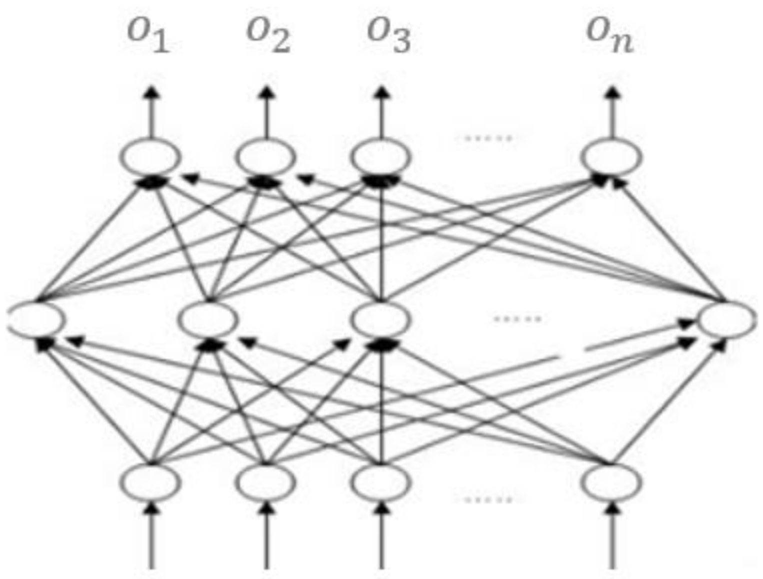

Figure 3: Structure of Coordinating Networks

Next are CFs and 2-tower DNNs. CF looks into the set of items uses have rated, but unlike UCF and ICF, as a matrix factorization technique, ALS learns the latent factors which can represent both users and items. After training, we can compute how similar the rest items are to the target item *i* and select *k* most similar items $\{i_1, i_2, i_3, \ldots, i_k\}$.

The basic part of ALS-CF is the construction of user-video matrix. Let's take Click interaction for example. We define it as,

$$C = \{c_{ij}\}_{n_u \times n_v}, \qquad (2)$$

Where each c_{ij} represents the number of clicks of video *j* by user *i* with the value being either a real number or missing, and n_u designates the number of users and n_v designates the number of videos.

Then ALS models both videos and users by giving them coordinates in a low dimensional feature space. Both the users and videos have their own feature vectors where the clicking of a video by a user is modeled as the inner product of the desired user and video feature vectors. Let *U* represent the user feature matrix and *V* represent the video feature matrix having both user and video feature vectors respectively, which is shown in Figure 4, where *d* is the dimension size of the feature space.

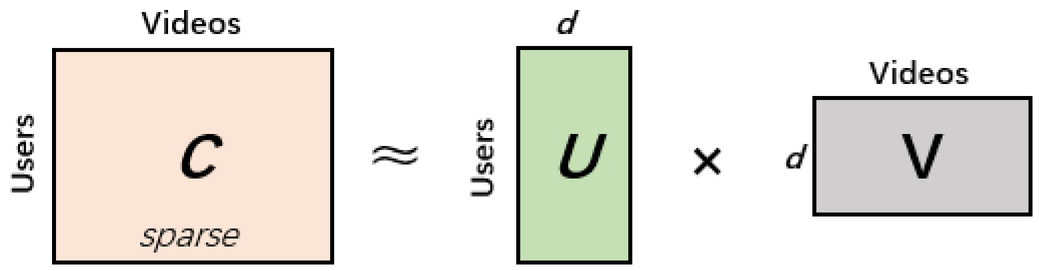

Figure 4: ALS Collaborative Filtering Computation

Next we need to define a loss function of *U* and *V*, which is the root mean square error (RSME), and minimize it to get the optimized matrices. Therefore,

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,v}|(o_{u,v} - p_{u,v})^2|}\,, \quad (3)$$

where $o_{u,v}$ and $p_{u,v}$ are observed and predicted clicks for user *u* and video *v* respectively [24]. And predicted value can be computed by:

$$p_{u,v} = < u_i, v_j >\,, \qquad (4)$$

Therefore in order to minimize RMSE, we formulate this problem to obtain the optimized latent factor vectors (u_i, v_j):

$$\left(u_i, v_j\right) = \min_{u,v} \sum_{(i,j)\in K} (o_{i,j} - p_{i,j})^2 + \lambda(\|u_i\|^2 + \left\|v_j\right\|^2)\,, \quad (5)$$

where the lambda part is a Tikhonov regularization to solve overfitting.

Then ALS fixes one of the unknowns u_i or v_j, and compute the other one to solve the least-squares problem by using gradient descent. This technique actually turns the non-convex problem into a quadratic function which can be solved by typical solutions.

After ALS training, we obtain the optimized matrices *U* and *V*, which contain the latent factors of users and videos. Then we use *V* to compute similarities between videos for Objective Clicking. Same steps for other objectives.

In summary, we can define the following generative process:

1. Construct user-video matrix for each objective, for example, objective *1* is clicking, then the value of the matrix indicates whether the user click the item or not.

2. Parallel ALS training for each objective.

2.1 Initialize configuration for matrix *U* by assigning average or small random numbers.

2.2 Fix *U*, solve *V* by minimizing the loss function.

2.3 Rotate, fix *V*, solve *U*.

2.4 Repeat 2.2 and 2.3 until convergence.

3. Extract *V* (latent factors of videos) after model trained.

4. Calculate s_{ij}, which indicates the similarity score with linear relationship between $item_i$ and $item_j$.

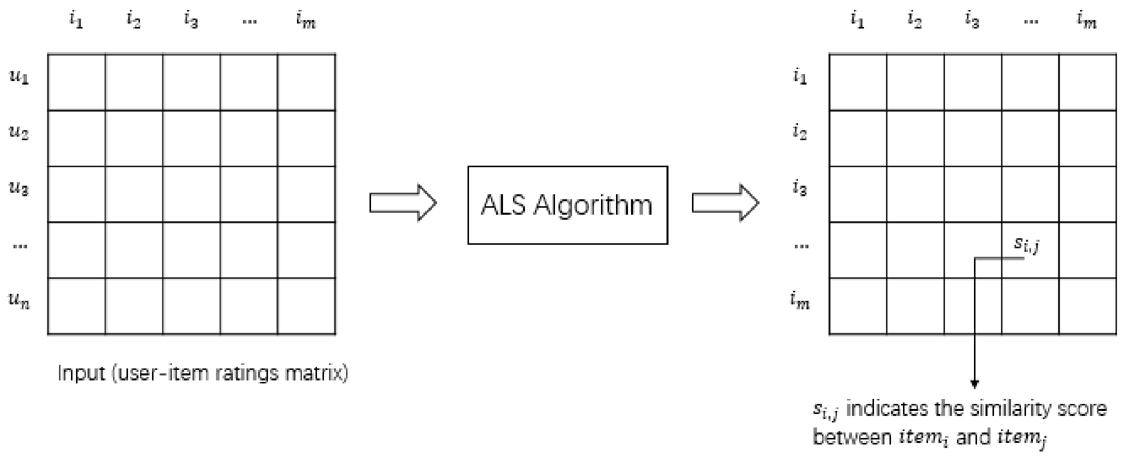
i_1 i_2 i_3 ... i_m
u_1 u_2 u_3 ... u_n
Input (user-item ratings matrix)
ALS Algorithm
i_1 i_2 i_3 ... i_m
$s_{i,j}$
$s_{i,j}$ indicates the similarity score between $item_i$ and $item_j$

Figure 5: get latent factors of items by ALS training

Meanwhile, we construct a two-tower DNN for each objective as shown in Figure 6. We revised the Youtube version to make it more suitable for our scenario [8]. After building each tower, we use Cosine function to compute the similarity between User Tower and Candidate Tower:

$$similarity = cos(\theta) = \frac{U \cdot C}{\|U\|\|C\|} = \frac{\sum_{i=1}^{n} U_i \times C_i}{\sqrt{\sum_{i=1}^{n}(U_i)^2} \times \sqrt{\sum_{i=1}^{n}(C_i)^2}}, \qquad (6)$$

AND we use the typical Log Loss function for minimizing.

We can define the following generative process:

1. Extract info from user profile and video profile to generate input features for two towers.

2. Use One-Hot for dense features and Embedding for sparse features. Dense feature: a feature has a numerical value which is measurable, and most values are not null, like ages, user level, tec. Sparse feature: a feature whose raw content is not numerical, or a feature with a wide range of categories, so when using One-Hot the most values of this feature are null, such as IDs, tags, actors.

3. Put two ReLU layers in the DNN part.

4. Do L2 normalization for avoiding overfitting.

5. For each objective, feed Two-tower DNN with particular historical user logs, for example, for objective sharing, use sharing logs.

6. Use the output of item tower as vector for calculating similarities between items.

7. Get the w_{ij}, which indicates the similarity score with non-linear relationship between $item_i$ and $item_j$.

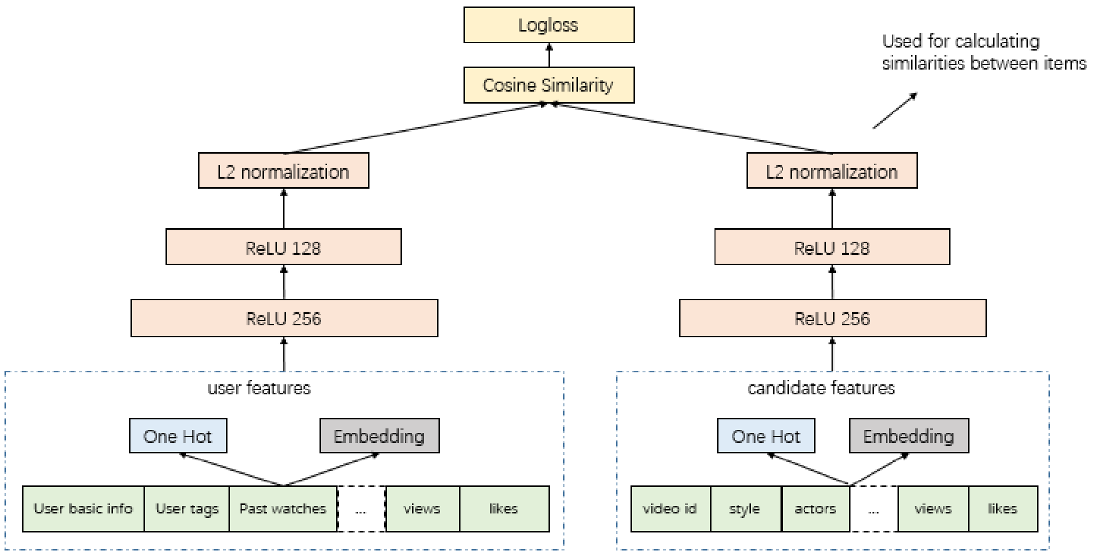

Figure 6: Revised Two-tower DNN

4.3 Merging and Ranking Stage

The merging stage is to generate the final result. Firstly, we use the maximum of their linear and non-linear similarity score because CF and DNN are two different solutions, and either of them with a high score is a strong relationship indicator. Secondly, there is a weights learning module. The detail is a Logistic Regression, using multi-classes outputs from CN, and use users' one week retention as label. In other words, we use whether a user stay or not as label to train and get weights of different objectives. Thirdly, we also keep an α_k which indicates the manually set weight by domain expertise. For example, if you consider the Likes more important, you can give it a higher weight. After all, given a source $video_i$, the final ranking score with $video_j$ can be formulated as:

$$Ranking\ score_{i,j} = \sum_{all\ the\ obejctives} \alpha_k * w_k * s_{i,j}\ , \qquad (7)$$

We can define the following generative process:

1. Given a source video and a target video for objective 1, get maximum of their linear and non-linear similarity score, as shown in Figure 7.

2. Weights Learning Module is trained and gets weights.

3. Manually set weight by domain expertise (if necessary).

4. For $item_i$ and $item_j$, get the final ranking score by weighted sum. (For objectives indicating negative feedbacks, like dislike, we do minus operation)

5. After step 4, give an arbitrary item, its related videos are the ones with TOP *N* ranking scores.

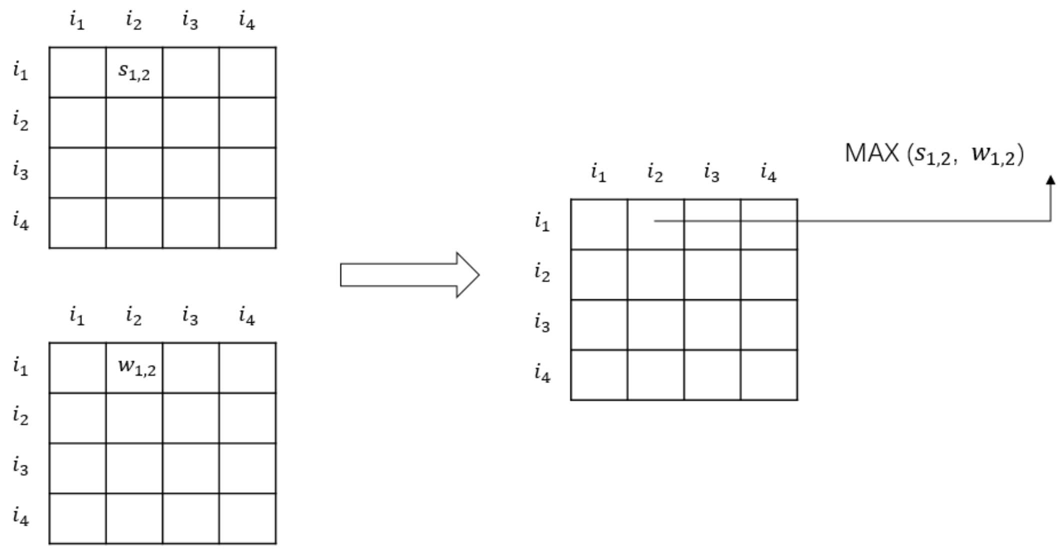

Figure 7: Combine linear and non-linear relationships

5 EXPERIMENT RESULTS

In this section, we introduce the A/B experiments of our proposed multi-objective learning algorithm to recommend related videos on one of the largest video sharing platforms, BiliBili. By using historical user-video interactions including click, played time, rating, comment, dislike and so on, we train our MTL algorithm, and conduct both offline and live experiments.

Bilibili, famous for its bullet screens, which lets viewers post or send each other text messages while watching a film or show, as shown in Figure 8, is one of the largest video sharing platform in the world, with 0.2 billion monthly active users. Its main business is focus on China, and the worldwide version is about to come soon. It's not only a video sharing platform, but also considered as the best online community among Chinese users. Most of the users are Chinese young people, and for every two young people in China, one of them is a BiliBili user. The website or app creates billions of user-video logs every day in the form of user activities interacting with recommended results for them to watch next. A key product of BiliBili provides the functionality of recommending related videos to watch next given a watched video, as shown in Figure 8. Its user interface provides multiple ways for users to interact with recommended videos, such as clicks, watches, likes, downloads, shares, and dismissals.

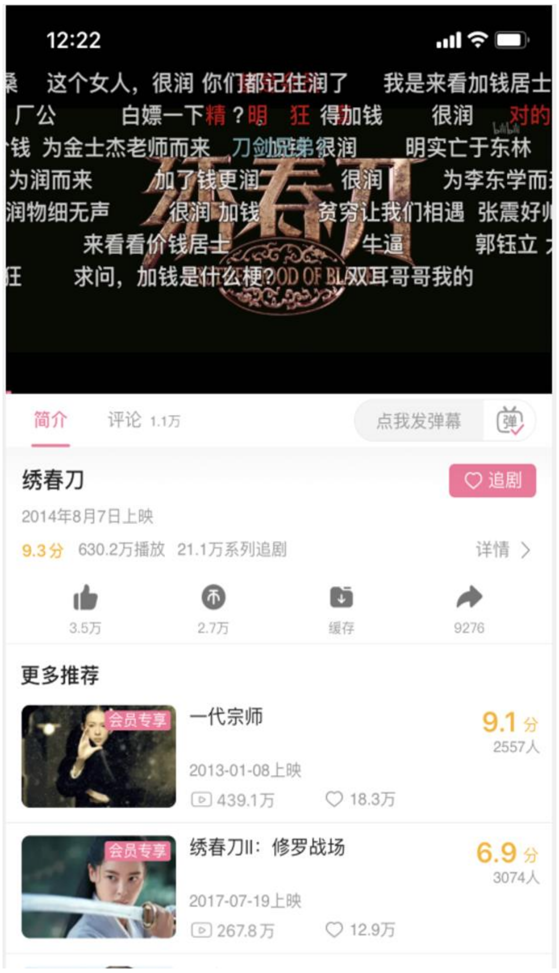

Figure 8: Recommending related videos on BiliBili

5.1 Experiment Setup

We use $TensorFlow^2$ to build the training and predicting of the model. We train both our MTL and baseline models.

For offline experiments, we monitor AUC for classification task and squared error for regression tasks. For live experiments, we conduct A/B testing comparing with production system. We set 9 groups and each has equal volume. We let 10 groups run for 7 days without any strategy to guarantee experimental confidence. And the difference of all metrics such as user visits, page view, click, ctr, and time spent between groups we monitored is within 0.02% for each day. Then we add corresponding strategy to each group, monitor results for 2 weeks, and calculate average value for 14 days. In summary, we examine multiple engagement metrics such as time spent at BiliBili, satisfaction metrics such as rate of liking or sharing, and dissatisfaction metrics such as rate of dismissals, dislikes, user negative comments, etc.

One big workload is tuning parameters. Here we list our optimized results for work reproducibility, so that other industrial researchers could easily apply it for their real-world scenarios. The threshold *e* to filter irrelevant videos in Stage 1 is 0.65. The threshold *t* of CN is 0.6. ALS-CF for clicking: rank=30, maxIter=20, lambda(regParam)=0.001, alpha=100. ALS-CF for time spent: rank=20, maxIter=17, lambda(regParam)=0.01, alpha=90. ALS-CF for liking: rank=31, maxIter=26, lambda(regParam)=0.015, alpha=30. ALS-CF for sharing: rank=15, maxIter=20, lambda(regParam)=0.02, alpha=50. ALS-CF for disliking: rank=18, maxIter=10, lambda(regParam)=0.005, alpha=10. Two-tower DNN structure: hidden1 = build_layer(item=256, user=256 activation='relu'), hidden2 = build_layer(item=128, user=128, activation='relu'), hidden3 = normalize(item='L2', user='L2'), l4 = trans(func='cos'). Two-tower training params: seed=99, loss='binary_crossentropy', optimizer='sgd', batch_size=512, shuffle=True, verbose=2, epochs=100.

5.2 Multitask Learning

To evaluate the performance of adopting ALS-CFs and Two-tower DNNs for multitask learning, we compare with baseline methods and conduct live experiments on BiliBili.

5.2.1 Baseline and Experimental Methods. Our baseline method is a typical ALS-CF for clicking, which is the production baseline on Bilibili, and is considered as the most popular implementation in industry. And we set various experimental methods to compare the differences. We also set a personalized model for comparing effects.

5.2.2 Live Experiment Results. The live experiment results in BiliBili are shown in Table 1, where No.9 represents our proposed algorithm. We report results on the engagement metric which captures user time spent on watching recommended videos, the satisfaction metric which captures how user likes the related videos after watching, and the dissatisfaction metric which captures the degree of disliking the videos. The red numbers indicate quite better performance, while the green numbers indicate worse performance. Especially for metric of negative feedback, a negative value means a reduction of negative feedbacks, so the lower, the better.

5.2.3 Detailed analysis of Experiment Results.

Compare No.2 to No.1: No.2 is an approach of manually looking for appropriate items by human expertise, which is far from efficient as we describe above. So it gets very bad performance but receive fewer negative feedbacks.

Compare No.3 to No.1: Played Time Metric goes up, while Clicking Metric goes down. So either is better than the other and it's like a tradeoff.

Compare No.4/No.5 to No.1: liking and sharing both serve Satisfaction Metric, so they have similar results with above one, but both Clicking and Playing Metric go down. In addition, they make people less complain. Not good.

Compare No.6 to No.1: for disliking model, given a source video, the video with top similarity score means recommending it will most likely cause users' negative feedbacks. So we use zero minus the score and rank in descending order. Negative feedback Metric goes down but not good for others.

Compare No.7 to No.1: more clicks and time spent than baseline. So it indicates that non-linear relationship learning (2-tower DNN) it better than linear relationship learning (ALS).

Compare No.8 to No.1: No.8 is a personalize recommendation with a MMoE model, so we see more people click, but the time spent does not have relatively equal rise, and especially more people complain. Since it is personalized, the generated videos contain more user interests, so more people click those. After playing for a few seconds, we find it has nothing to do with the previous video but they are looking for that type, so time spent doesn't increase too much. More complaints explain the module is not suitable for personalized solution as we have mentioned above.

Compare No.9 to N0.1: obviously the best for all 9 groups.

Table 1: BiliBili live experiment results for all solutions

No.	Model Architecture	Metric of Multiple Ranking Objectives			
		Clicking	Played Time	Satisfaction	Negative feedback
1	CF for clicking(base)	/	/	/	/
2	knowledge-based	- 10.62%	- 8.79%	- 0.24%	- 4.73%
3	CF for time spent	- 2.67%	+ 2.11%	+ 0.33%	+ 0.35%
4	CF for liking	- 3.61%	- 3.17%	+ 3.81%	- 2.04%
5	CF for sharing	- 3.43%	- 2.97%	+ 3.42%	- 0.83%
6	CF for disliking	-2.26%	-1.07%	+ 0.02%	-3.11%
7	2-tower DNN for clicking	+ 1.07%	+ 1.03%	+ 0.13%	- 0.03%
8	Personalized Model(MMoE)	+ 3.61%	+0.61%	- 2.62%	+ 5.72%
9	Multi-task Learning	+ 3.77%	+ 4.81%	+ 4.53%	- 6.21%

6 CONCLUSION AND FUTURE WORK

In this paper, we started with the description of a few real-world challenges in designing and developing industrial non-personalized recommendation systems to generate next videos for watching. These challenges include maintaining high correlation between source video and target videos, as well as optimizing multiple competing ranking objectives. We designed a multi-objective learning algorithm to solve these challenges. More specifically, we add a candidate generation stage and extract features from the video profile info to maintain relevance between items. We also train ALS-CFs and Two-tower DNNs in parallel to capture both linear and non-linear relationships from user-item interactions, and use coordinating networks to select samples for different training tasks, and merge results of all models to efficiently optimize multiple ranking objectives. Via live experiments on one of the world's largest video sharing platforms, BiliBili, we showed that our proposed algorithm leads to state-of-the-art performance on engagement, satisfaction and dissatisfaction metrics.

Moreover, this model bridges the gap between deep learning techniques and non-personalized Recommender System. Our solution is not only an algorithm, but also a framework which is enable various extensions. The cosine similarity calculation could be replaced by other formulas, such as Pearson Correlation Coefficient calculation. ALS could be replaced by other latent factor models. Learning non-linear relationship could use other DNN structure, not only Two-tower DNN. All in all, the algorithm is designed as a framework and supporting multitask learning. All kinds of revised versions are welcome and we are looking forward to see further performance boost.

REFERENCES

[1] Hu Y , Koren Y , Volinsky C . Collaborative Filtering for Implicit Feedback Datasets[C]// Eighth IEEE International Conference on Data Mining. IEEE, 2009.

[2] Sarwar B . Item-Based Collaborative Filtering Recommendation Algorithms[C]// Proc. the 10th International World Wide Web Conference (WWW10), Hong Kong, May 1-5 (2001). 2001.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, 2009.

[4] Barkan O , Koenigstein N , Yogev E , et al. CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item

recommendations[C]// the 13th ACM Conference. ACM, 2019.

[5] Yang B , Lei Y , Liu D , et al. Social collaborative filtering by trust[C]// IJCAI International Joint Conference on Artificial Intelligence. AAAI Press, 2013. 4 Deep Social Collaborative Filtering

[6] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, Peng Jiang. 2019. A Pareto-Eficient Algorithm for Multiple Objective Optimization in E-Commerce Recommendation. RecSys '19,20–28.

[7] Kotov C A . Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries[J]. ACM, 2012.

[8] Rodriguez P . Comparing Simple Recurrent Networks and n -Grams in a Large Corpus[J]. Applied Intelligence, 2003, 19(1-2):39-50.

[9] Miller L W , Senetar J J . TWO-STAGE QUENCH TOWER FOR USE WITH OXYGENATE CONVERSION PROCESS[J]. US, 2002.

[10] Park M H , Hong J H , Cho S B . Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices[C]// International Conference on Ubiquitous Intelligence and Computing. Springer, Berlin, Heidelberg, 2007.

[11] Sarwar B . Item-Based Collaborative Filtering Recommendation Algorithms[C]// Proc. the 10th International World Wide Web Conference (WWW10), Hong Kong, May 1-5 (2001). 2001.

[12] Barkan O , Koenigstein N , Yogev E , et al. CB2CF: a neural multiview content-to-collaborative filtering model for completely cold item recommendations[C]// the 13th ACM Conference. ACM, 2019.

[13] Rong Z , Wilkinson D , Provost F . Social Network Collaborative Filtering. social science electronic publishing, 2008.

[14] Wang H , Wang N , Yeung D Y . Collaborative Deep Learning for Recommender Systems[J]. ACM, 2015.

[15] Barkan O , Koenigstein N . Item2Vec: Neural Item Embedding for Collaborative Filtering[C]// 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2016.

[16] Xue F , He X , Wang X , et al. Deep Item-based Collaborative Filtering for Top-N Recommendation[J]. 2018.

[17] Joachims T , Granka L , Pan B , et al. Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search[J]. Acm Transactions on Information Systems, 2007, 25(2):7.

[18] Rich Caruana. 1997. Multitask learning. Machine learning 28, 1 (1997), 41–75.

[19] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube Recommendations. In Proceedings of the 10th ACM conference on recommender systems. ACM, 191–198.

[20] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of- experts. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 1930–1939.

[21] Jean-Antoine Désidéri. 2009. Multiple-Gradient Descent Algorithm (MGDA).

[22] Milojkovic N , D Antognini, Bergamin G , et al. Multi-Gradient Descent for Multi-Objective Recommender Systems. 2019.

[23] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In KDD, pages 1235–1244, 2015.

[24] J. Lee, M. Sun, and G. Lebanon. A comparative study of collaborative filtering algorithms. CoRR, abs/1205.3193, 2012.

[25] Hai-Ling X U , Xiao W U , Xiao-Dong L I , et al. Comparison Study of Internet Recommendation System[J]. Journal of Software, 2009, 20(2):350-362.

[26] Walter, Battiston S , Schweitzer F . A model of a trust-based recommendation system on a social network[J]. Autonomous Agents and Multi-Agent Systems, 2008, 16(1):57-74.

[27] Nicholas F C , Carswell I B . Method of Operating a Channel Recommendation System[J]. US, 2007.

[28] Takeuchi Y , Sugimoto M . CityVoyager: An Outdoor Recommendation System Based on User Location History[C]// Ubiquitous Intelligence & Computing, Third International Conference, Uic, Wuhan, China, September. DBLP, 2006.

[29] Park M , Hong J , Cho S . Location-Based Recommendation System Using Bayesian User ' s Preference Model in Mobile Devices. 2007.

[30] Li, Lei, Wang, et al. SCENE: a scalable two-stage personalized news recommendation system.[J]. ACM, 2011.

[31] Zhang Y , Jiao J . An associative classification-based recommendation system for personalization in B2C e-commerce applications[J]. Expert Systems with Applications, 2007, 33(2):357-367.

[32] S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In CIKM, pages 811–820, 2015.

[33] D. Liang, L. Charlin, J. McInerney, and D. M. Blei. Modeling user exposure in recommendation. In WWW, pages 951–961, 2016.

[34] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. Proceedings of the IEEE, 104:11–33, 2016.

[35] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In ICDM, pages 497–506, 2011.

[36] S. Rendle. Factorization machines. In ICDM, pages 995–1000, 2010.

[37] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In UAI, pages 452–461, 2009.

[38] S. Rendle, Z. Gantner, C. Freudenthaler, andL. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In SIGIR, pages 635–644, 2011.

[39] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In NIPS, pages 1–8, 2008. [30] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In ICDM, pages 791–798, 2007.

[40] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, Ed Chi. 2019. Recommending What Video to Watch Next: A Multitask Ranking System. RecSys '19,43–51.

[41] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu. Scalable semi-supervised learning by efficient anchor graph regularization. IEEE Transactions on Knowledge and Data Engineering, 28(7):1864–1877, 2016.

[42] M. Wang, H. Li, D. Tao, K. Lu, and X. Wu. Multimodal graph-based reranking for web image search. IEEE Transactions on Image Processing, 21(11):4649–4661, 2012.

[43] M. Wang, X. Liu, and X. Wu. Visual classification by l1 hypergraph modeling. IEEE Transactions on Knowledge and Data Engineering, 27(9):2564–2574, 2015.

[44] X. Wang, L. Nie, X. Song, D. Zhang, and T.-S. Chua. Unifying virtual and physical worlds: Learning towards local and global consistency. ACM Transactions on Information Systems, 2017.

[45] X. Wang and Y. Wang. Improving content-based and hybrid music recommendation using deep learning. In MM, pages 627–636, 2014.

[46] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In WSDM, pages 153–162, 2016.

[47] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative knowledge base embedding for recommender systems. In KDD, pages 353–362, 2016.