Master's Thesis of Data Science

Content Representation Learning for Cold-start Video Recommendations

콜드 스타트 비디오 추천시스템을 위한 컨텐츠 표현 학습

February 2023

Graduate School of Data Science
Seoul National University
Data Science Major

Jooeun Kim

Content Representation Learning for Cold-start Video Recommendations

Advisor Joonseok Lee

Submitting a master's thesis of
Data Science

December 2022

Graduate School of Data Science
Seoul National University
Data Science Major

Jooeun Kim

Confirming the master's thesis written by
Jooeun Kim
January 2023

Chair Taesup Kim (Seal)
Vice Chair Joonseok Lee (Seal)
Examiner Jay-Yoon Lee (Seal)

Abstract

Cold-start item recommendation is a long-standing challenge in recommendation systems. A common approach to tackle cold-start problem is using content-based approach, but in movie recommendations, rich information available in raw video contents or textual descriptions has not been fully utilized. In this paper, we propose a general cold-start recommendation framework that learns multimodal content representations from the rich information in raw videos and text, directly optimized over user-item interactions, instead of using embeddings pretrained on proxy pretext task. In addition, we further exploit multimodal alignment of the item contents in a self-supervised manner, revealing great potential in content representation learning. From extensive experiments on public benchmarks, we verify the effectiveness of our method, achieving state-of-the-art performance on cold-start movie recommendation.

Keyword : video representation learning, multi-modal learning, cold-start recommendation, content-based recommendation, transformers
Student Number : 2021-22861

Table of Contents

Chapter 1. Introduction

Recommendation systems are widely adopted for a variety of real-world applications, *e.g.*, online retails, video sharing platforms, and more, as the scale of items that people should choose from has been rapidly growing. Collaborative filtering (CF) [3, 15], recognizing preference patterns observed in user-item interactions, has been successfully applied to personalized recommendation systems to provide potentially preferred items in a personalized manner.

Despite its success, CF approaches suffer from several challenges, one of which is the cold-start problem. Since CF relies only on user and item interaction, it is not capable of generating personalized recommendations for a new user without any records. Likewise, a brand-new item with no user feedback cannot be recommended to the right customers who are most likely to prefer that item.

Indeed, cold-start is actually common and important in modern recommendation systems. In YouTube, for example, 500 hours of contents are being uploaded every minute [21]. With a standard CF recommendation system, those fresh contents can only be recommended to some random users until sufficient interaction data is collected. Another example is Netflix, where new movies or TV series often compete for the main advertisement space. Since this space is a limited resource, it is extremely important for the supplier to select users who will most likely enjoy the new contents to maximize its revenue. Again, cold-start item recommendation would play a key role in selecting the right set of users for each fresh content that has not gathered any user feedback.

A common approach to tackle the cold-start problem is using side information of the users or items to get prior knowledge of them. Unlike collaborative filtering, which relies only on user and item interactions, content-based (CB) approaches utilize attributes or properties of the users and items, *e.g.*, demographic information of the users or metadata of the items. Since content information becomes available at the time of release, it is possible to retrieve a set of neighboring items that are of similar content, and it may be recommended to users who like this kind of items. Traditionally, metadata like genre or artist was mainly used. [29, 33, 41, 50]

With recent advances in deep learning, there have been attempts to learn more powerful item representations directly from raw contents, *e.g.*, music or movie. CDML [27] and GCML [26] propose to learn video embeddings from raw contents (*e.g.*, pixels), trained on co-watch statistics among videos aggregated over multiple users. These models turn out to be strong on video retrieval and classification tasks, while performance on personalized recommendation is reported marginal. This is probably because the item representation is trained on a signal aggregated over multiple users, instead of individual user feedback, and thus this representation may not provide sufficiently fine-grained details needed for a personalized recommendation.

CLCRec [46] is another recent work that is relevant to our approach. Adopting a hybrid approach, CLCRec is equipped with both a CF and a CB module, combining their predictions to work on both cold and warm items. According to their experiments, this model achieves the current state-of-the-art on the cold-start recommendation problem on MovieLens. From their ablation study, however, the best performance is actually achieved with significantly unbalanced weights towards the CF part. This indicates there may be a large headroom to further improve cold-start performance with stronger content features.

Here, we pose the key question: have we been using the rich content information in raw videos sufficiently and properly for video recommendations? For this question, we find at least two areas with high potential for improvement.

First, the content representations that have been pervasive in literature may not be suitable for recommendation tasks. Including the aforementioned methods, most content-based recommendation models have used features extracted from models pretrained on tasks other than recommendation. To be specific, most video recommendation models that utilize visual content as side information rely on features trained on image classification, *e.g.*, ImageNet [8], where the model learns to classify images into 1,000 classes. A few fully-connected layers are usually added and trained on recommendation data, expecting them to transform the embeddings optimized for image classification into some useful representation for content-based recommendation. Are we confident that embeddings learned to distinguish only 1,000 general classes are aware of fine-grained visual details subtle enough for recommendation?

For a good movie recommendation, we may expect the features to contain information like story or mood, not just which objects exist in the video or high-level genres. Since the highest bottleneck layer, which is most frequently used to represent the video in previous papers, is trained to compactly represent each example with the most essential information for the target task, *i.e.*, classification, we may not expect such fine-details about the movie to be present in this content embedding. Even if additional layers are trained on top of the image classification embeddings, they may not be able to learn any details if the content embedding is already too general, without preserving any details. For this reason, it is hard to expect the features trained for a significantly different objective to convey good representation for recommendation.

Second, previous content-based movie recommendation models have been ignorant of the multimodality of the items. Movies contain rich side-information of diverse modalities, such as videos, synopsis, or other metadata, that represent the characteristic of the contents. Recently, multimodal video representation learning has been notably advanced and applied to video or clip retrieval from a text query [31, 35, 40, 48], taking advantage of powerful contextualization capability of Transformers [43]. Most content-based recommendation models, however, have been using a simple late-fusion, concatenating embeddings from different modalities [13, 46].

It is time for us to reconsider the decision of naively reusing embeddings pretrained on a proxy task which requires content understanding at a much coarse level, separately from each modality. In this paper, we propose a general cold-start recommendation framework to learn item representations purely from raw content, directly optimized to estimate user-item interactions without relying on weakly relevant pretraining tasks. In other words, our model maps the raw content to an embedding space where users and items with similar taste are clustered, and this representation is generalizable to unseen users and items. Our model is end-to-end trainable, without requiring a pretrained model. We also take an initial step towards elaborating multimodal learning for content-based recommendation by adding a loss that exploits multimodal correspondence of the content feature pairs for the same item.

Specifically, our model takes multimodal content signals as input. Each modality is represented as a sequence of its atomic unit; *e.g.*, a word for text or a small image patch for visual modality. Each sequence is aggregated to represent the entire video using Transformers [43]. Additionally, video-level representations for multiple modalities are aligned. Then we put a rating ranking head predicting how much a user will like an item by collaborative filtering. This prediction is compared to the known preference in the training data, and the loss arisen from this backpropagates all the way back to the lowest level of the model, treating pixels or word tokens.

From extensive experiments, we verify that content representations directly trained on the recommendation domain generalize significantly better than the ones transferred from weakly relevant proxy domains. Thereby, our model achieves state-of-the-art performance on cold-start video recommendation, demonstrating strong adaptation capability to another movie dataset.

Our main contributions are summarized as follows:

- To the best of our knowledge, our model is the first attempt to learn video content representations directly on individual user-item interactions from the raw content.
- Our model exploits the multimodality of the item contents in a self-supervised manner, significantly improving cold-start recommendation performance.
- From extensive experiments, we demonstrate that our proposed method achieves state-of-the-art performance on cold-start movie recommendation task.

Chapter 2. Related Work

2.1. Cold-start Recommendations

CF has contributed a lot to enhance the performance of recommender systems in the existence of plentiful historical data [19, 28, 36, 37, 39, 49], but the cold-start problem is its long-standing challenge, where no historical interaction record of user or item exists. To tackle this problem, diverse approaches have been proposed.

MWUF [53] proposes to warm up cold item embeddings with meta-scaling and shifting networks. DropoutNet [44] randomly dropouts item or user embeddings to make the model better adapt to the cold-start condition. Heater [54] tackles the problem with a randomized training mechanism and mixture-of-experts transformation. Recently, various meta-learning approaches tackle cold-start recommendation, *i.e.*, MAMO [10], Meta-E [34], MetaHIN [32], MeLU [25], and PAML [47].

2.2. Content-based Recommendations

Auxiliary information like content features integrated in CF models has been beneficial for alleviating cold-start problem [6, 7]. CLCRec [46] maximizes the mutual dependencies between item content and collaborative signals using contrastive learning. CLCRec shares a common theme with our model in that both utilize content features to tackle cold-start recommendation. However, CLCRec trains embeddings on image classification data and transfers them to recommendation, while our model learns the video representations directly on the recommendation task. Also, CLCRec uses an image model ignoring temporal dependencies, while we use a spatio-temporal video model to extract item features. Although CLCRec claims that it combines CF and CB, in reality it relies heavily on CF, according to their experiment.

CDML [27] is another model that uses content features (audio-visual) and proves to be useful in cold-start scenario. Its refined model, GCML [26], learns video embeddings from a relational graph and shows better performance than CDML. However, both models are not personalized in that they learn item-item co-watch similarity aggregated over all users, not at individual user level. On the other hand, our model explicitly uses individual user feedback to learn the item representations.

Recently, CVAR [50] proposes model-agnostic framework to generate enhanced warmed-up item embeddings for cold items using content information. DUIF [14] is a feature learning approach for image recommendation. MTPR [12] and CC-CC [38] use item content features to leverage collaborative signals. Although there are many content-based approaches to tackle the cold-start problem, our model is distinctive in that it is the first vision transformer specialized for content-based recommendation to fully optimize the feature extractor for the recommendation task.

2.3. Contrastive Learning

Contrastive learning is a self-supervised task, learning to discriminate which pairs of data points are similar and different from the dataset, widely used in computer vision and NLP [4, 5. 16. 18, 20, 22]. Recent works employ contrastive learning in recommender systems to optimize the representations of users and items. For instance, Liu et al. [30] proposes a graph contrastive learning for a general recommender system, introducing debiased contrastive module to alleviate the sample bias. CLRec [51] employs contrastive learning to improve DCG in recommendation. CLCRec [46] adopts contrastive learning to preserve collaborative signals in the content representations. Our method also employs contrastive loss for rating prediction and multi-modal alignment. Unlike other methods [46, 51] that use fixed negative samples or adopt additional module for negative sampling, we use other samples in the mini-batch, shuffled each time. Seeing more diverse negative samples without extra cost, our method demonstrates a superior generalization capability on cold-start recommendations.

Chapter 3. Problem Formulation and Notations

In this paper, we assume implicit feedback from the items so there are only two types of ratings: *preferred* or *unknown*. Given a binary preference matrix $R \in \{0,1\}^{M \times N}$ with M users and N items, an element $R_{ij} = 1$ indicates that the item j is preferred by the user i, while $R_{ij} = 0$ means unknown. The matrix R can be split into two parts: R_w with warm items and R_c with cold items, where all entries within R_c are zeros. The cold-start recommendation task is predicting preferable items within R_c; in other words, retrieving a list of items that each user i may prefer among the cold items.

Each item is provided with a set of content attributes. Our framework is general enough to treat arbitrary number of sequence attributes, but for the ease of explanation, we will use two concrete types, visual (*e.g.*, raw frames) and text (*e.g.*, synopsis) modalities throughout this section. The visual side information for item j is denoted by $X^{(j)} \in R^{T_j \times h \times w \times 3}$, where T_j is the number of frames of the item j and $\mathrm{h} \times w$ is the frame size. Each frame at timestamp t is denoted by $x_t^{(j)}$. Similarly, the text side information for item j is denoted by $w^{(j)} \in \{1, \dots, |V|\}^{L_j}$, where V is the vocabulary set and L_j is the length of the text for item j.

We target only cold-start items with no interaction records with any user, and we assume no cold-start user. Although cold-start users can be modeled in a similar way, we do not tackle this problem because no public dataset provides meaningful user side information due to privacy issues. Cold-start item recommendation performance is evaluated by ranking all unseen items for each user and comparing the top K items from the ranked list with the items that the user actually gave positive feedback to.

Chapter 4. Preliminary

We briefly review Transformers [43], a powerful model that achieves state-of-the-art performance on sequence-to-sequence (Seq2Seq) tasks like machine translation. It applies a self-attention mechanism in an encoder-decoder structure to learn context by tracking relationships in sequential data. We first describe the Transformer encoder in detail, followed by how it is utilized for language and visual modalities by representative models. We do not cover the decoder since it is not used in our model.

4.1. Transformer Encoder

Given an input sequence of tokens (*e.g.*, words or frames) of length T, they are first embedded to vectors, $Z \equiv \{z_1, \ldots, z_T\}$, where $\mathrm{z_t} \in R^d$. Then, Z is fed to a series of encoder blocks, where each block is composed of a self-attention layer and a feed-forward network, that enrich token representations with contextual information from other tokens in the sequence.

First, the token embeddings $\mathrm{Z} \in \mathrm{R}^{\mathrm{T} \times \mathrm{d}}$ are transformed to three special representations, namely, query ($\mathrm{Q} \in \mathrm{R}^{\mathrm{T} \times \mathrm{d}'}$), key ($K \in R^{T \times d'}$), and value ($V \in R^{T \times d'}$), by linear transformation, where d' is not necessarily same as d. Then, the self-attention is defined by

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d'}}\right)V. \tag{1}$$

Intuitively, attention of each token is represented as a weighted average of other token embeddings (using V) in the same sequence, where the weight is proportional to the relevance score (computed using Q and V) between them. The learnable parameters are linear mappers from token embeddings to Q, K, and V. Since each token may have more than one semantics depending on the neighboring tokens, Transformer adopts multiple heads to allow the token to represent different semantics depending on the context.

After the multi-head self-attention, the embeddings are fed into a position-wise feed-forward network, allowing further transformation. These steps are repeated by stacking L blocks. The output of the last encoder block is the final embedding of each token. Optionally, we may put an additional classification token (`[CLS]`) to learn aggregated representation of the entire sequence. Without having specific meaning, `[CLS]` aggregates tokens without being biased towards itself as other regular tokens do. The Transformer is often trained by losses arisen from a downstream task like classification, performed based on this aggregated embedding from `[CLS]` token.

4.2. BERT

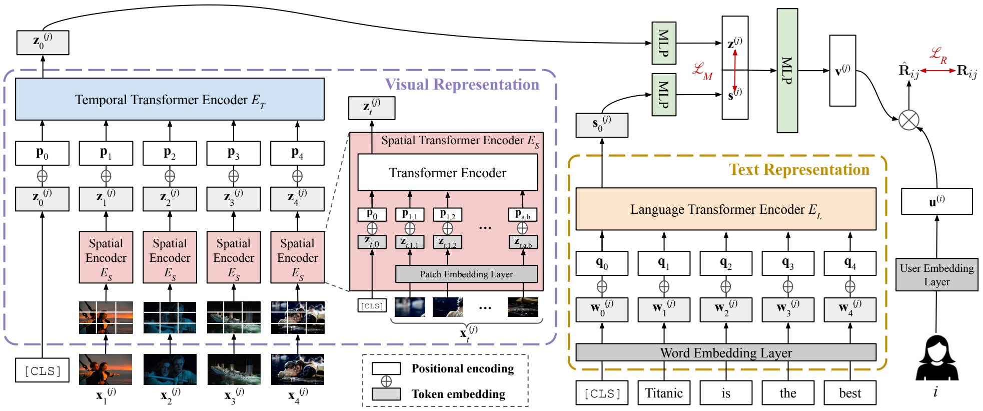
Bidirectional Encoder Representations from Transformers (BERT) [9] is a language model that learns representations from unlabeled text by self-supervised learning. BERT uses only the encoder of the Transformer. The main training objectives are to predict masked tokens in sentences (Masked language modeling; MLM) and to predict whether two input sentences are consecutive (Next Sentence Prediction; NSP). With MLM, the randomly masked tokens are classified based on context (remaining tokens). For NSP, the embedding corresponding to the **[CLS]** token is fed to a classifier determining if the two input sentences are consecutive. For both, a classification loss (*e.g.*, cross entropy) is used to train the model. BERT is powerful in precisely learning semantics of words when trained on large-scale corpus, achieving state-of-the-art performance on various NLP tasks.

4.3. Vision Transformers

Vision Transformer (ViT) [11] is a Transformer-based object recognition or image classification model. While text Transformers like BERT use words in a sentence as input tokens, ViT employs a Transformer over fixed-size (*e.g.*, 16 × 16) patches split from the input image. Each image patch is linearly transformed to a patch embedding, added with a positional encoding and fed into the Transformer encoder. Optionally, multiple blocks of Transformers may be stacked. ViT adds a special learnable classification token (**[CLS]**) to the sequence. At the end of the last block, this **[CLS]** token encodes the learned representation of the entire image by contextualizing over all patches. It is fed into an MLP head performing the downstream task, *e.g.*, image classification. We adopt this architecture to embed video frames.

Chapter 5. The Proposed Method

For a user i and an item j, the goal of our model is estimating the preference score R_{ij}. The user representation $u^{(i)} \in R^d$ is simply learned with an embedding layer, similarly to the traditional collaborative filtering models. In order to treat cold-start items, however, our model learns to represent the items from their content signals (side information). Fig. 1 illustrates an example of using visual and text features to represent an item j, denoted by $z^{(j)} \in R^{d_z}$ and $s^{(j)} \in R^{d_s}$, respectively, where d_z is the embedding dimensionality of the visual modality and d_s is that of the textual modality. More details on how to represent each modality will be described subsequently. Note that our model is general enough to take arbitrary number of features of any type.



$\mathbf{z}_0^{(j)}$
Visual Representation
Temporal Transformer Encoder E_T
$\mathbf{p}_0$ $\mathbf{p}_1$ $\mathbf{p}_2$ $\mathbf{p}_3$ $\mathbf{p}_4$
$\mathbf{z}_0^{(j)}$ $\mathbf{z}_1^{(j)}$ $\mathbf{z}_2^{(j)}$ $\mathbf{z}_3^{(j)}$ $\mathbf{z}_4^{(j)}$
Spatial Encoder E_S
[CLS]
$\mathbf{x}_1^{(j)}$ $\mathbf{x}_2^{(j)}$ $\mathbf{x}_3^{(j)}$ $\mathbf{x}_4^{(j)}$
$\mathbf{z}_t^{(j)}$
Spatial Transformer Encoder E_S
Transformer Encoder
$\mathbf{p}_0$ $\mathbf{p}_{1,1}$ $\mathbf{p}_{1,2}$ … $\mathbf{p}_{a,b}$
$\mathbf{z}_{t,0}$ $\mathbf{z}_{t,1,1}$ $\mathbf{z}_{t,1,2}$ $\mathbf{z}_{t,a,b}$
Patch Embedding Layer
[CLS]
$\mathbf{x}_t^{(j)}$
Positional encoding
Token embedding
MLP
MLP
$\mathcal{L}_M$
$\mathbf{z}^{(j)}$
$\mathbf{s}^{(j)}$
MLP
$\mathbf{v}^{(j)}$
$\mathcal{L}_R$
$\hat{\mathbf{R}}_{ij}$ $\mathbf{R}_{ij}$
$\mathbf{s}_0^{(j)}$
Text Representation
Language Transformer Encoder E_L
$\mathbf{q}_0$ $\mathbf{q}_1$ $\mathbf{q}_2$ $\mathbf{q}_3$ $\mathbf{q}_4$
$\mathbf{w}_0^{(j)}$ $\mathbf{w}_1^{(j)}$ $\mathbf{w}_2^{(j)}$ $\mathbf{w}_3^{(j)}$ $\mathbf{w}_4^{(j)}$
Word Embedding Layer
[CLS] Titanic is the best
$\mathbf{u}^{(i)}$
User Embedding Layer
i

Figure 1: Overall architecture of the proposed model. Given a raw video and its text description, the model extracts features using the visual (left) and text (right) representation modules. Overall item representation is computed by an MLP, and the final rating is predicted by dot product with the target user embedding, learned in the manner of collaborative filtering.

5.1. Visual Representation Learning

The left box marked as Visual Representation in Fig. 1 illustrates our visual representation learning module. From T_j video frames of an item j, we first randomly sample a clip consisting of F consecutive frames, $\{x_1^{(j)}, x_2^{(j)}, \dots, x_F^{(j)}\}$. Next, we adopt a two-step architecture where we compute frame-level embeddings for each frame using Spatial Encoder (E_S) and then obtain the entire clip representation ($z^{(j)}$) by Temporal Encoder (E_T) from the frame embeddings. We choose this two-step architecture in order to effectively capture the spatio-temporal semantics of the video, including both details at frame level and overall information flow through temporal dimension. The architecture is similar to the model 2 of the Video Vision Transformer (ViViT) [2], reported as most efficient and cost-effective. In order to learn complex underlying spatio-temporal dynamics from videos, choosing a computationally efficient architecture is critically important.

Spatial Encoder (E_S). Each frame $x \in \{x_t^{(j)}: t = 1, \dots, F\}$ is divided into $P \times P$ image patches, forming a set $\{x_{a,b}: a = 1, \dots, h/P, b = 1, \dots, w/P\}$ to be fed into the spatial encoder E_S. Adopting the ViT architecture [11] introduced in Sec. 4, E_S first embeds the input image patches $\{x_{a,b}\}$ with a linear layer, denoted by $\{z_{a,b}\}$. Then, it adds learnable positional encodings $\{p_{a,b}\}$ to the patch embeddings, depending on the location of each patch within the image. Together with the `[CLS]` token, patch embeddings are fed into L_S Transformer encoder blocks. During this process, each patch embedding is updated to capture diverse semantics in the image. The output embedding corresponding to the `[CLS]` token, denoted by $z_t^{(j)}$, encodes semantics of the entire image $x_t^{(j)}$.

Temporal Encoder (E_T). Now, the clip sampled from item j is represented as a sequence of frame-level embeddings, $\{\boldsymbol{z}_{\boldsymbol{1}}^{(j)}, \ldots, \boldsymbol{z}_{F}^{(j)}\}$. In order to aggregate them into a single clip-level embedding, we feed this sequence to another Transformer blocks, Temporal Encoder (E_T). Similar to E_S, each frame embedding is added with a learnable temporal positional encoding, $\{p_1, \ldots, p_F\}$, and a classification token (`[CLS]`) is concatenated to the sequence. While E_S captures the spatial semantics of each frame, the temporal transformer encoder E_T is in charge of capturing temporal semantics in the clip. We take the final `[CLS]` token output $z_0^{(j)}$ from E_T, followed by an MLP that outputs our final visual content representation, $z^{(j)}$.

5.2. Text Representation Learning

Similar to the visual modality, we use a Transformer-based architecture to encode content signals in text modality. Unlike the visual encoder, which adopts a two-step architecture for spatial and temporal aggregation, we choose a single-stage encoder architecture for the text. Specifically, we adopt the BERT [9] described in Sec. 4.

Language Encoder (E_L). Given a sequence of L_j words, the word embedding layer encodes them into a sequence of word embeddings, $\{w_1^{(j)}, \dots, w_{L_j}^{(j)}\}$. A **[CLS]** token is inserted and also passes through the word embedding layer, denoted by $w_0^{(j)}$. Then, they are added with the positional encoding $\{q_0, q_1, \dots, q_{L_j}\}$. Unlike the learnable positional encodings used for visual modality, we follow the fixed positional encodings in BERT [9], as it is more suitable for text. The position-aware word embeddings pass through the language Transformer encoder E_L which contextualizes the word embeddings throughout the entire text and produces another sequence of transformed word representations, denoted by $\{s_0, s_1, \dots, s_{L_j}\}$. The final embedding of the **[CLS]** token, $s_0^{(j)}$, from E_L passes through an MLP and becomes our text representation embedding, $s^{(j)}$.

5.3. Training Objectives

Once computed, each modality representation is concatenated and passed through MLP layers, producing the final item embedding $v^{(j)} \in R^d$. The final preference R_{ij} is the dot-product of user and item embeddings; that is, $\widehat{R_{ij}} = {u^{(i)}}^{\top v^{(j)}}$. Given the ground truth R_{ij}, we train the model by maximizing $\{\widehat{R_{ij}}: R_{ij} = 1\}$ and minimizing $\{\widehat{R_{ij}}: R_{ij} = 0\}$ using contrastive loss.

Rating Ranking Loss. For Top-K recommendations, we are interested only in the relative relevance among the items to the target user's taste. Thus, we train the model to generalize well to predict higher scores for preferred items and lower scores for the others, rather than regressing to a fixed scale.

For this reason, we choose contrastive learning, which is recently widely adopted for representation learning [18, 24, 45]. Unlike other supervised learning where we explicitly fit to a fixed label, contrastive learning trains the model to distinguish positive and negative examples based on relative relevance. Particularly, SimCLR [4] applies contrastive learning to a self-supervised setting, where the positive example is created by data augmentation while all other examples in the same mini-batch are considered as negatives. In our recommendation setting, the contrastive loss can be applied similarly to predict the ratings from user and item embeddings. Specifically, for each user, the item paired in the same example (which means that this user actually likes the item) is used as the positive, while all other items belonging to different pairs in the mini-batch are considered as negatives. With contrastive loss, the encoder is trained to maximize the dot product between the user and item embeddings in the same pair, while minimizing that of the different pairs in the mini-batch. Rating ranking loss $\mathcal{L}_R$ for each mini-batch consisting of B pairs of user and item is formulated by

$$\mathcal{L}_R = -\sum_{i=1}^{B} \frac{u^{(i^*)^\top v^{(i^*)}}}{\sum_{j=1}^{B} u^{(i^*)^\top v^{(j^*)}}} - \sum_{i=1}^{B} \frac{u^{(i^*)^\top v^{(i^*)}}}{\sum_{j=1}^{B} u^{(j^*)^\top v^{(i^*)}}}, \qquad (2)$$

where $\boldsymbol{i}^*$ denotes the actual index of a user or an item in the training dataset for the i-th example in minibatch.

Here, from $\mathcal{L}_R$, false negative issue arises when applying contrastive loss to pair-wise recommendation training batches. Unlike other contrastive models like SimCLR where other examples in the mini-batch are negatives for sure, a user has multiple positive items and an item also has multiple positive users. Thus, an item j_1 paired with a user i_1 might be actually positive for another user i_2, although i_2 is paired with another item j_2 in the particular batch. We may optionally remove these false negatives from the denominator of Eq. (2) for more precise training. We report empirical performance with or without false negative filtering in Sec. 7.

Multi-modality Loss. With multiple content features from more than one modalities, we may take further advantage of self-supervision by training the model to learn multi-modal correspondence. In a simple setting, item content embeddings from each modality for the same item are concatenated and passed through another MLP, projected to the same embedding space. This embedding is our final item content representation, which contains latent representation of multi-modal item content that relates the item characteristics and general patterns of user preference observed in the data.

Although this simple design reasonably fuses multimodal signals, we further exploit multimodal correspondence. Specifically, we apply contrastive loss to all item embeddings within each mini-batch, maximizing the similarity between embeddings from the same item with different modalities, while minimizing the similarity between all other combinations. Multimodality loss $\mathcal{L}_M$ for a mini-batch with B pairs of visual and text representation is

$$\mathcal{L}_M = -\sum_{i=1}^{B} \frac{z^{(i^*)}{}^{\top} s^{(i^*)}}{\sum_{j=1}^{B} z^{(i^*)}{}^{\top} s^{(j^*)}} - \sum_{i=1}^{B} \frac{z^{(i^*)}{}^{\top} s^{(i^*)}}{\sum_{j=1}^{B} z^{(j^*)}{}^{\top} s^{(i^*)}}, \tag{3}$$

where $z^{(i^*)}$ and $s^{(i^*)}$ denote the visual and text representation of the i-th item in the mini-batch, respectively.

Note that the two approaches above are the simplest ones that we can easily try, while more complicated multimodal losses are also possible, *e.g.*, cross-modal attention [31].

Overall Objective. We linearly combine the two loss functions above to form the overall objective $\mathcal{L}$,

$$\mathcal{L} = \mathcal{L}_R + \lambda \mathcal{L}_M, \tag{4}$$

where λ is a hyperparameter that controls relative importance of the two loss components.

5.4. Inference

Recall that we randomly sample a clip with F frames from the target video during training, as it helps the model to observe various aspects of each movie, serving as data augmentation. At inference, we sample $C > 1$ clips and predict movie preference scores with each clip. Then, we aggregate those scores by taking the max:

$$\widehat{R_{ij}} = \max_{c=1,\dots,C} u^{(i)\top} v^{(j_c)}, \tag{5}$$

where j_c indicates our embedding for the clip c randomly sampled from item j. In this way, we cover wider range of the video and compute the score based on a clip that the user most likely prefers.

Chapter 6. Experimental Settings

We conduct extensive experiments to verify the effectiveness of our method by comparing it with competing models on cold-start recommendation tasks. In this section, we describe our evaluation protocol, including the datasets, baselines and metrics.

Table 1: Overview of Our Datasets

Dataset		Before filtering	After filtering		
			Train	Validation	Test
MovieLens	Users	162,541	28,542	28,527	28,522
	Items	62,423	4,198	378	277
	Ratings	25,000,095	2,024,323	496,130	659,783
Yahoo Movies	Users	7,642	4,506	2,542	2,898
	Items	11,915	1,802	402	400
	Ratings	211,231	41,572	8,357	10,163

Datasets. We use two widely-used standard recommendation benchmarks: MovieLens 25M[1][17] and Yahoo Movies[2]. Both datasets provide explicit user ratings from 1 (least preferred) to 5 (most preferred), so we convert them to implicit ones by taking only ratings 3.5 or above as positives following [27, 52]. We also exclude items with any missing content information for both datasets. Also, we filter out the users with less than 20 ratings from MovieLens. However, as Yahoo Movies has a small number of items and sparse ratings, we do

[1] https://grouplens.org/datasets/movielens/

[2] https://webscope.sandbox.yahoo.com/

not filter by the number of ratings. After filtering, we randomly split the items into three parts--warm, cold validation, and cold test--with the ratio of 85:7.5:7.5 for MovieLens and 70:15:15 for Yahoo Movie. The warm partition is used for training, and the cold validation is used to tune hyper-parameters. The cold test set is used to evaluate the final performance of each model. The scale of datasets after filtering and split is summarized in Table 1. The number of overlapping movies between the two datasets is 891.

Content Features. Our proposed model is feature-agnostic, meaning that the content features can be of any modality or format, depending on the item type or the choice of feature extractor model. In our experiments, we choose to use video and text features.

For visual content, we use movie trailers provided by MovieLens [1] and MovieNet[3], since the full videos are publicly unavailable for most movies due to copyright. From each video, frames are sampled at 2 fps. We drop the first and last 10% of the sampled frames, since they often include age rating screen or ending credits. The average length of the trailers is 137 seconds for MovieLens dataset and 140 seconds for Yahoo Movies, so we get around 220 frames per video on average for both datasets. For each mini-batch, we randomly sample $B = 48$ trailers, and a single sub-clip of length $F = 32$ is randomly sampled within each trailer. This allows the model to see various parts

[3] https://movienet.github.io/

of the video uniformly throughout the whole training process, after sufficient number of epochs.

For text content, we use movie synopsis collected from IMDB[4] for MovieLens. Yahoo Movies dataset self-contains synopsis. These synopses are two to three sentences that summarize the movie overview. The sentences are first tokenized into word tokens with a maximum length of 512, using uncased BERT large tokenizer [9]. Average number of tokens in text contents is 54.7 and 83.0 for MovieLens and Yahoo Movies, respectively.

Evaluation Metrics. Following CLCRec [46], we treat all users with varied number of ratings equally by averaging the score for each user. We use three widely-used metrics for ranking tasks: {Precision, Recall, NDCG}@K with $K = \{1, 5, 10, 20\}$.

Competing Models. We compare three recent cold-start item recommendation models using content information: CLCRec [46], DropoutNet [44], and CVAR [50]. For a fair comparison, we use the public code released by the authors to train the competing models on our dataset. For the side-information of the baselines, we use ViT [11] embeddings pretrained on ImageNet [8] and BERT [9] embeddings for the text features. The weights for the pretrained models are kept frozen during training. We empirically choose hyperparameters by

[4] https://www.imdb.com/

cross-validation.

Model Hyperparameters. We experiment with $d = \{32, 64, 128, 256\}$ for the dimensionality of users ($u^{(i)}$) and items ($v^{(j)}$), while fixing d_z and d_s to 128, respectively. For visual features, we spatially split each frame to $P \times P$, where $P = 16$. Within the Spatial Transformer Encoder E_S , we stack $L_S = 4$ blocks of Vision Transformers. For the Temporal Encoder E_T, we again stack $L_T = 4$ blocks. For both E_S and E_T, positional encodings p are learned from data, similarly to ViT [11]. For text features, we follow the architecture of BERT [9]. For the MLPs after visual and text representation modules, we try several single-layer or two-layer fully-connected networks, compared in Sec. 7. The MLP after concatenation of modality-specific features consists of a single fully-connected layer. We perform grid search for λ within the range $[0,1]$. We randomly sample $C = 10$ clips for inference.

Training Hyperparameters. We use Adam optimizer [23] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We warm-up the learning rate by linearly increasing during the first 3 epochs, and train up to 200 epochs. After 70% of training, we decay the learning rate to 20% of the initial one, which is found by grid search among $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. We use mini-batch size $B = 48$.

Chapter 7. Results and Discussion

In this section, we describe detailed experimental results and provide insightful discussions from our observation.

7.1. Comparison to the Baselines

In Table 2, we report the performance on cold-start recommendation evaluated by NDCG@K, Prec@K, and Recall@K with $K = \{1,5,10,20\}$.

Table 2: Comparison with the Baselines on MovieLens & Yahoo Movies (%)

Dataset	Method	NDCG (↑)				Precision (↑)				Recall (↑)			
		@1	@5	@10	@20	@1	@5	@10	@20	@1	@5	@10	@20
MovieLens	DropoutNet [44]	7.33	4.99	4.72	5.29	7.33	4.36	4.27	4.79	7.33	4.38	4.34	5.54
	CLCRec [46]	9.09	6.59	6.77	8.62	9.09	6.02	6.31	7.19	9.09	6.06	6.71	10.27
	CVAR [50]	8.89	9.11	9.09	9.49	8.89	9.12	9.10	**9.56**	8.89	9.13	9.12	9.71
	Ours	**14.05**	**11.41**	**10.16**	**10.40**	**14.05**	**10.77**	**9.13**	8.14	**14.05**	**10.80**	**9.39**	**11.33**
Yahoo Movies	DropoutNet [44]	1.10	1.68	2.40	3.29	1.10	1.12	1.10	1.01	1.10	2.16	4.05	6.40
	CLCRec [46]	0.75	6.50	6.47	6.65	0.75	3.87	2.24	1.24	0.75	7.95	8.34	8.97
	CVAR [50]	1.07	1.74	2.31	3.09	1.07	1.34	1.17	1.05	1.07	2.67	4.13	6.49
	Ours	**6.79**	**8.53**	**11.66**	**12.48**	**5.39**	**5.87**	**6.04**	**6.27**	**6.79**	**8.86**	**14.54**	**16.24**

According to Table 2, our model outperforms all baselines under all metrics except for Precision@20 on MovieLens. Especially, our model significantly outperforms other models on Yahoo Movies, where observations are far sparser than MovieLens. This indicates that our content features directly trained on user-item interaction data turn out to be even stronger on a sparser condition, when cold-start recommendation is more important.

Another noticeable observation is the relationship between the models' performance and the value of K for metrics@K on MovieLens, where the average number of positive items in the test set is $23.1(\pm 14.0)$. For instance, our approach tends to be stronger with a smaller K, so it will be more suitable for cases like watch next, where only the top one item is important. Baseline models like CVAR, on the other hand, tend to be stronger with a larger K, so they will be more suitable for homepage recommendations, where multiple items are presented at the same time. On Yahoo Movies, the average number of positive items is $3.5(\pm 6.1)$, much lower than 20. Thus, all methods tend to show higher scores with larger K.

7.2. Ablation Study

Modality Ablation. To explore the effectiveness of multimodal features and alignment loss ($\mathcal{L}_R$), we compare the performance of our model with visual content only, visual and text features with and without applying $\mathcal{L}_M$ on MovieLens. With $\lambda = 0$ in Eq. 3, the final embedding is simply generated from concatenated visual and text representation, $[z^{(j)}; s^{(j)}]$, without applying $\mathcal{L}_M$.

As reported in Table 3, using multimodal features and alignment loss between them improve the overall recommendation performance. One notable thing in the result is that the performance is even lower than that of the model trained with a single modality, if the multimodal features are not aligned by $\mathcal{L}_M$. In other words, just concatenating multimodal embeddings and stacking MLPs on top is insufficient to fully utilize the multimodality of the item contents. However, the model trained with $\lambda > 0$ (we use $\lambda = 0.5$) outperforms both the visual-only model and the concatenated visual-text model. This result implies that exploiting multimodality has a great potential for cold-start recommendation, since what we demonstrate in this experiment is just a simple loss that encourages higher similarity between multimodal representations for the same content. More advanced form of multimodal correspondence learning (*e.g.*, [31, 48]) may further improve cold-start recommendation performance, and we leave this as a promising future work.

Table 3: Modality Ablation Study (MovieLens)

Modality	NDCG@10	Prec@10	Recall@10
Visual	9.41	8.68	9.06
Visual + Text	8.14	7.37	7.64
Visual + Text + $\mathcal{L}_M$	**10.16**	**9.13**	**9.39**

Model Architecture Search. We explore specific architecture of our model from two perspectives: the dimensionality $\boldsymbol{d}$ of the final content and user embeddings, and the depth and width of MLP layers on top of the learned content representations, mapping $\boldsymbol{z}_{\mathbf{0}}^{(j)}$ and $\boldsymbol{s}_{\mathbf{0}}^{(j)}$ to $\boldsymbol{z}^{(j)}$ and $\boldsymbol{s}^{(j)}$, respectively.

Table 4: Embedding Dimensionality Exploration (Yahoo Movies)

d	NDCG@10	Prec@10	Recall@10
32	6.87	3.81	10.17
64	8.91	4.41	10.58
128	10.20	5.52	12.82
256	**11.66**	**6.04**	**14.54**

First, Table 4 summarizes embedding size (d) search result trained on Yahoo Movies. As expected, larger embedding size leads to better performance in general, although the gain per additional dimension diminishes as d gets larger. For other experiments, we use $d = 128$ for computational efficiency.

Table 5: Model Architecture Search (MovieLens; visual only)

Model	NDCG@10	Prec@10	Recall@10
[1024-128]	9.41	8.68	9.06
[1024-512-128]	9.50	**8.80**	**9.09**
[1024-1024-128]	**9.98**	8.78	9.06
[2048-128]	9.82	7.94	8.26
[2048-1024-128]	9.17	6.33	8.45

For the MLP structure, we experiment on MovieLens with visual contents only, with different number of layers and dimensionality. We try one to two layers of MLPs, with different widths among $\{128, 512, 1024, 2048\}$ as summarized in Table 5. Stacking more layers shows marginal performance gain, while expanding the dimensionality of the layers does not always improve the performance. From this point, we conclude that the complexity of content representations is learned well enough at the feature extraction modules, so the MLP layers can be concise.

Table 6: False Negative Filtering (MovieLens)

	NDCG@10	Prec@10	Recall@10
With Filtering	9.85	**9.14**	9.21
Without Filtering	**10.16**	9.13	**9.39**

False Negatives Filtering. To quantify the effect of false negatives in pair-wise recommendation training, we compare two models learned with and without false negative filtering in our rating ranking loss $\mathcal{L}_R$, on MovieLens. Table 6 shows marginal impact of applying false negative filtering. We conjecture that this is partly because false negatives are less likely to be included in a small number of mini-batch as the scale of the dataset gets larger. Therefore, considering the computational overhead coming from false negative filtering, we conduct all other experiments without false negative filtering unless noted otherwise.

7.3. Content Representation Evaluation

To verify if our video embeddings are properly trained to capture users' watch behavior in general, we conduct a couple of additional studies of transfer learning.

First, we compare our full model against the same model where the feature extractor is replaced with ViT [11] features pretrained on ImageNet, average-pooled over temporal axis. Table 7 reports the experimental results. Comparing the first four rows, we observe that the performance of our model trained from scratch outperforms the same model using ViT pre-trained embeddings on both MovieLens and Yahoo Movies. From this, we confirm the importance of direct training on recommendation signals, instead of using a proxy pretraining task.

Next, we evaluate transferability of our learned content representation from one dataset to another. If so, we verify the learned content model is general enough to be competent not just on unseen items in the same dataset but also on different set of users.

Table 7: Experimental Result on Content Representations

Pretraining	Target	NDCG@10	Prec@10	Recall@10
From scratch	MovieLens	**9.50**	**8.80**	**9.09**
ViT (ImageNet)		5.32	5.31	5.50
From scratch	Yahoo Movies	**7.25**	**3.63**	**9.24**
ViT (ImageNet)		1.59	0.79	2.66
Ours (MovieLens)		**5.22**	**2.63**	**6.90**

The last row in Table 7 shows the performance of cold-start recommendation on Yahoo Movies, using content embeddings trained on MovieLens. The result indicates that the MovieLens embeddings significantly outperform the ViT embeddings, proving again that directly training on the recommendation signals is far more effective than using commonly-used classification models. In other words, our content representation is well-generalized to collect useful information from the raw input contents, explaining why our model strongly outperforms all other baselines in cold-start recommendation.

7.4. Qualitative Analysis

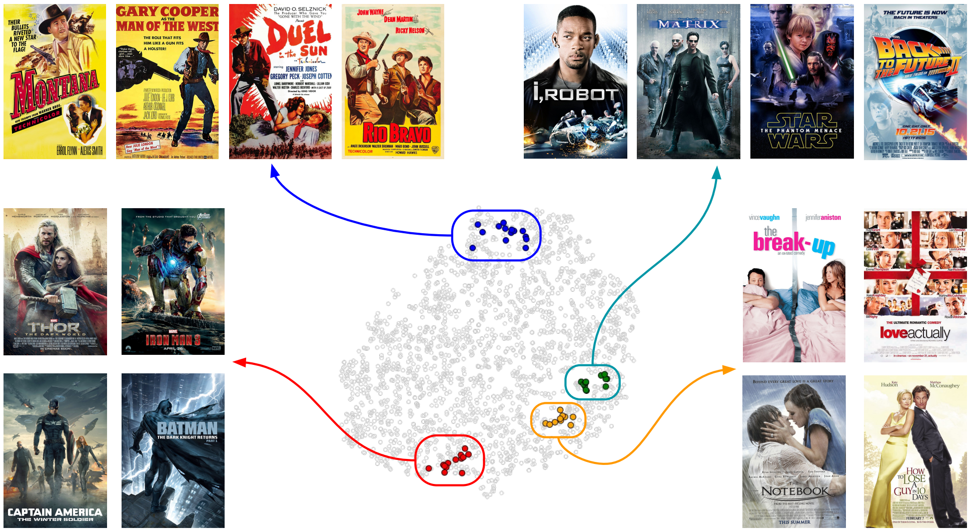


Figure 2: t-SNE Visualization of Learned Video Embeddings

We visualize the learned video embeddings in 2D for qualitative understanding. Fig. 2 presents the t-SNE plot [42] of the video embeddings learned by our best model using both visual and text content on MovieLens.

We observe a few interesting examples where similar movies are positioned nearby each other in the embedding space. For instance, Fig. 2 illustrates four clusters with highly relevant movies in different colors; where the red, orange, green, and blue clusters consist of heroes, romantic comedies from mid-2000s, science fictions, and western movies from mid-1900s, respectively. The full list of colored dots is listed in Table 8.

Table 8: Full list of the movie titles in the colored clusters in Fig. 2

Red Cluster	Orange Cluster	Green Cluster	Blue Cluster
Batman: The Dark Knight Returns, Part 1 (2012) Batman: Year One (2011) Superman Unbound (2013) Captain America: The First Avenger (2011) Captain America: The Winter Soldier (2014) Iron Man 2 (2010) Iron Man 3 (2013) Thor: The Dark World (2013) Guardians of the Galaxy (2014) Fantastic Four (2005) LEGO Batman: The Movie - DC Heroes Unite (2013) Son of Batman (2014) Batman: Assault on Arkham (2014)	Love Actually (2003) Break-Up, The (2006) Notebook, The (2004) How to Lose a Guy in 10 Days (2003) 12 Dates of Christmas (2011) Princess Diaries 2: Royal Engagement, The (2004) P.S. I Love You (2007) Elizabethtown (2005) Bridget Jones: The Edge of Reason (2004) Catch and Release (2006)	I, Robot (2004) Star Trek VI: The Undiscovered Country (1991) Star Wars: Episode I - The Phantom Menace (1999) Star Wars: Episode II - Attack of the Clones (2002) Back to the Future Part II (1989) Back to the Future Part III (1990) Matrix, The (1999) Matrix Revolutions, The (2003) Battlefield Earth (2000) Pitch Black (2000) Spaceballs (1987) Battle of Los Angeles (2011) Highlander II: The Quickening (1991) Time Machine, The (2002)	Duel in the Sun (1946) Ride Lonesome (1959) Montana (1950) Man of the West (1958) Man Who Never Was, The (1956) Unforgiven, The (1960) Bonnie and Clyde (1967) She Wore a Yellow Ribbon (1949) Rio Bravo (1959) Hombre (1967) Cimarron (1960) Bad Day at Black Rock (1955) Spartacus (1960) Easy Rider (1969)

Chapter 8. Summary and Future Work

In this work, we focus on how to tackle the item cold-start problem in recommendation. We propose a general cold-start recommendation framework that first attempts to learn multimodal content representations from the rich information in raw videos and text, directly optimized to estimate user-item interaction. Also, our model further exploits the multimodality of the item contents (visual-text) in a self-supervised manner and attains better recommendation performance under a cold-start condition. We conduct extensive experiments on two public datasets to verify the effectiveness of our proposed method, which outperforms the state-of-the-art baselines on cold-start movie recommendation.

To the best of our knowledge, this is the first attempt to fully optimize the item content representation learner with personalized user feedback. Additionally, our model introduces a great potential in item content representation learning by aligning multimodal signals in a self-supervised manner. Our simple multimodal loss achieves a great performance gain on cold-start recommendation, suggesting the potential of implementing more complicated multimodal architectures for content representation learning.

Bibliography

[1] Sami Abu-El-Haija, Joonseok Lee, Max Harper, and Joseph Konstan. 2018. MovieLens 20M YouTube Trailers Dataset. In MovieLens.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. Vivit: A video vision transformer. In Proc. of the IEEE/CVF Conference on International Conference on Computer Vision (ICCV).

[3] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI).

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In Proc. of the International Conference on Machine Learning (ICML).

[5] Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. (2021).

[6] Yashar Deldjoo, Maurizio Ferrari Dacrema, Mihai Gabriel Constantin, Hamid Eghbal-Zadeh, Stefano Cereda, Markus Schedl, Bogdan Ionescu, and Paolo Cremonesi. 2019. Movie genome: alleviating new item cold start in movie recommendation. User Modeling and User-Adapted Interaction 29, 2 (2019), 291–343.

[7] Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. 2020. Recommender systems leveraging multimedia content. ACM Computing Surveys (CSUR) 53, 5 (2020), 1–38.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).

[10] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proc. of the International Conference on Learning Representations (ICLR).

[12] Xiaoyu Du, Xiang Wang, Xiangnan He, Zechao Li, Jinhui Tang, and Tat-Seng Chua. 2020. How to learn item representation for cold-start multimedia recommendation. In Proc. of the ACM International Conference on Multimedia.

[13] Xingzhong Du, Hongzhi Yin, Ling Chen, Yang Wang, Yi Yang, and Xiaofang Zhou. 2018. Personalized video recommendation using rich contents from videos. IEEE Transactions on Knowledge and Data Engineering 32, 3 (2018), 492–505.

[14] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In Proc. of the IEEE/CVF Conference on International Conference on Computer Vision (ICCV).

[15] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. Commun. ACM 35, 12 (1992), 61–70.

[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. Advances in Neural Information Processing Systems (NIPS) 33 (2020).

[17] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (TIIS) 5, 4 (2015), 1–19.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In CVPR.

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proc. of the ACM International Conference on World Wide Web (WWW).

[20] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua

Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In Proc. of the International Conference on Learning Representations (ICLR).

[21] Seong Jae Hwang, Joonseok Lee, Balakrishnan Varadarajan, Ariel Gordon, Zheng Xu, and Apostol Natsev. 2019. Large-scale training framework for video annotation. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[22] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. Advances in Neural Information Processing Systems (NIPS) 33 (2020).

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[24] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. IEEE Access 8 (2020), 193907–193934.

[25] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[26] Hyodong Lee, Joonseok Lee, Joe Yue-Hei Ng, and Paul Natsev. 2020. Large Scale Video Representation Learning via Relational Graph Clustering. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[27] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Natsev. 2018. Collaborative deep metric learning

for video understanding. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[28] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local Low-Rank Matrix Approximation. In Proc. of the International Conference on Machine Learning (ICML).

[29] Tianqiao Liu, Zhiwei Wang, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Recommender systems with heterogeneous side information. In Proc. of the ACM International Conference on World Wide Web (WWW).

[30] Zhuang Liu, Yunpu Ma, Yuanxin Ouyang, and Zhang Xiong. 2021. Contrastive learning for recommender system. arXiv:2101.01317 (2021).

[31] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. VilBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. Advances in Neural Information Processing Systems (NIPS) 32 (2019).

[32] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[33] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In Proc. of the ACM Conference on Recommender Systems (RecSys).

[34] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In Proc. of the International ACM Conference on Research and Development in Information Retrieval (SIGIR).

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In Proc. of the International Conference on Machine Learning (ICML).

[36] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI).

[37] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In The adaptive web. Springer.

[38] Shaoyun Shi, Min Zhang, Xinxing Yu, Yongfeng Zhang, Bin Hao, Yiqun Liu, and Shaoping Ma. 2019. Adaptive feature sampling for recommendation with missing content feature values. In Proc. of the ACM International Conference on Information and Knowledge Management (CIKM).

[39] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. Advances in artificial intelligence (2009).

[40] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. VideoBERT: A joint model for video and language representation learning. In Proc. of the IEEE/CVF Conference on International Conference on Computer Vision (ICCV).

[41] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research

directions. Electronic Commerce Research and Applications 37 (2019), 100879.

[42] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research (JMLR) 9, 11 (2008).

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In Advances in Neural Information Processing Systems (NIPS).

[44] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing cold start in recommender systems. Advances in neural information processing systems 30 (2017).

[45] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In Proc. of the International Conference on Machine Learning (ICML).

[46] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In Proc. of the ACM International Conference on Multimedia.

[47] Runsheng Yu, Yu Gong, Xu He, Yu Zhu, Qingwen Liu, Wenwu Ou, and Bo An. 2021. Personalized adaptive meta learning for cold-start user preference prediction.

[48] Bowen Zhang, Hexiang Hu, Joonseok Lee, Ming Zhao, Sheide Chammas, Vihan Jain, Eugene Ie, and Fei Sha. 2020. A hierarchical multi-modal encoder for moment localization in video corpus. arXiv:2011.09046 (2020).

[49] Ruisheng Zhang, Qi-dong Liu, Jia-Xuan Wei, et al . 2014. Collaborative filtering for recommender systems. In Proc. of

the IEEE International Conference on Advanced Cloud and Big Data.

[50] Xu Zhao, Yi Ren, Ying Du, Shenzheng Zhang, and Nian Wang. 2022. Improving Item Cold-start Recommendation via Model-agnostic Conditional Variational Autoencoder. arXiv:2205.13795 (2022).

[51] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive learning for debiased candidate generation in large-scale recommender systems. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[52] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In Proc. of the ACM SIGKDD International conference on knowledge discovery & data mining.

[53] Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to warm up cold item embeddings for cold-start recommendation with meta scaling and shifting networks. In Proc. of the International ACM Conference on Research and Development in Information Retrieval (SIGIR).

[54] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In Proc. of the International ACM Conference on Research and Development in Information Retrieval (SIGIR).

Abstract in Korean

콜드 스타트 아이템 추천은 추천시스템 연구에서 오래된 문제 중 하나이다. 콜드 스타트 문제를 해결하기 위해 흔히 사용해온 방법은 컨텐츠 기반 접근 방식을 사용하는 것이지만, 영화 추천 시스템 분야에서는 원본 비디오 및 원문 설명 등에 내재된 풍부한 정보를 충분히 활용해오지 못했다. 본 논문에서 제안하는 콜드 스타트 추천 프레임워크에서는 원본 비디오와 텍스트의 풍부한 컨텐츠 정보를 기반으로 멀티모달 컨텐츠 표현을 학습하는 과정에서, 다른 태스크에 사전 학습된 임베딩을 활용하는 대신 유저-아이템 상호작용 정보를 이용하여 직접 임베딩을 최적화하는 방법을 제안한다. 더 나아가, 본 연구는 자기 지도 학습 방법을 통해 여러 모달리티로 표현되어 있는 아이템 컨텐츠를 고려함으로써 컨텐츠 표현 학습의 발전 가능성을 재조명한다. 최종적으로 주요 벤치마크 데이터셋에 대한 다양한 실험을 통해 본 연구에서 제안하는 방법론의 효과를 입증함과 동시에 콜드 스타트 영화 추천 분야에서 해당 분야 최고 성능을 보이는 사실을 확인하였다.

주요어 : 비디오 표현 학습, 멀티모달 학습, 콜드 스타트 추천, 컨텐츠 기반 추천, 트랜스포머 모델

학 번 : 2021-22861