MSE | MASTER OF SCIENCE IN ENGINEERING

Zurich University of Applied Sciences
zh aw School of Engineering

VT2: RECOMMENDATIONS ON MULTIMEDIA DATA

Claude Lehmann

Zurich University of Applied Sciences
Winterthur, Switzerland
lehl@zhaw.ch

ABSTRACT

The availability of large computational resources and a boom in recommending multimedia content on streaming services leads to new opportunities for research at the intersection between recommender systems and the use of multimedia features. While the recommender system problem is well understood, we believe the question on how to make use of all the available multimedia data is still largely unanswered.

In this paper, we present an end-to-end framework that utilizes multimodal features — in particular video trailers — for generating rating predictions on the popular MovieLens benchmark [1]. The framework learns generalized embedding vectors for each individual movie and uses the similarity between trailers in combination with historic user ratings. In our experiments, we are unable to outperform the current state-of-the-art collaborative filtering method libFM [2] with our content-based filtering approach. Our solution based on video embeddings measures an RMSE of 0.96, compared to state-of-the-art performance of 0.76 RMSE. In addition, we present a reasoning why the prediction performance could not be met. Furthermore, we reveal problems when working with videos from online streaming platforms and present strategies to clean the dataset and further mitigate these issues.

Index Terms— Embeddings, MovieLens20M, Multimodal Features, Neural Networks, Recommender Systems

1. INTRODUCTION

Recommender systems are not an inherently new topic, early attempts were made in the 1970's with the Grundy system [3], which used the idea of stereotypes to generate individual models for their users. However, the field came to life in the 1990's with the meteoric rise of the internet. In academia, the "algebra for recommendation" [4] presented the mathematical goals of a recommender system. Practical systems emerged shortly after, GroupLens [5] collected ratings on Usenet news articles and generated predictions for users with a similar taste. Video Recommender [6] allowed a virtual community of movie buffs to find new movies with the same idea as the GroupLens system. The Ringo [7] system fulfilled the same role in the world of music and finally, PHOAKS [8] applies a similar idea like the PageRank [9] algorithm later used by Google, using URL interconnectivity to gauge relevance for recommending information on the internet.

More recently, neural networks started to surpass human performance on different visual tasks. In the ImageNet challenge, the top-5 classification error for humans was measured at 5.1% [10]. Meanwhile, neural networks are able to achieve a top-5 classification error of as low as 1.2% [11, 12].

Just as the attention mechanism heralded the start of a new era for natural language processing [13, 14, 15], these advances in visual understanding open up new research directions. This paper in particular is concerned with the use of multimodal features for recommender systems. Recommender systems traditionally use historic ratings and metadata in predicting new ratings, as the famous challenges and benchmark datasets show [1, 16, 17]. Adding auxiliary features that go beyond the age or gender of a user, or the category of an item, is only just starting to be explored [18, 19]. We believe there is information in multimodal features, and as a first step videos, which should improve the ability to find similarities between items. The main questions are:

- *How can we use auxiliary, multimodal features in a content-based recommender systems?*
- *How can the content of a video be condensed, such that a generalized representation is formed?*

In this paper, we investigate how multimodal features — and in particular videos — can be used to condense the essence of an item into an embedding vector that can be further used for augmenting recommender systems. Choosing

videos as the first multimodal feature stems from the large impact of movie trailers in driving the sales for upcoming movies [20]. Trailers act as a short sample, a summary, of a movie. We hypothesize there exists a connection between a movie and its trailer — a relationship that likely resembles the relationship between a viewer's sentiment towards the movie and towards the trailer. Our goal is to utilize this relationship, modifying the way recommender systems work traditionally. Furthermore, the existence of a trailer suggests confidence in a movie and exudes a level of quality, as enough scenes were shot to assemble a full trailer [20]. However, in our dataset we only look at movies where both the trailer exists and the movie ended up in cinemas, to be watched by viewers.

We propose a novel end-to-end framework to extract keyframes from a movie trailer, use said keyframes to generate a movie embedding, and generate rating predictions by comparing the similarity between trailers. We evaluate our architecture against a well-known benchmark dataset in order to measure the effect on the predicted ratings. Additionally, we compare the predicted ratings against a state-of-the-art method to see, whether a fusion might be beneficial. Initial results are inferior to the state-of-the-art results by 26% and show no benefit as a linear combination of both methods.

The remainder of this paper is structured as follows: Section 2 describes approaches for the general recommender problems on the example of movie recommendations. In addition, we outline our end-to-end system architecture for a hybrid, multimodal recommendation system. Section 3 describes the experimental setup, including how the dataset deviates from pre-existing benchmarks and shows our results. Finally, in Section 4 we present our conclusion, giving an overview of ongoing issues and future work opportunities.

2. A MULTIMODAL APPROACH TO RECOMMENDER SYSTEMS

2.1. Related Work

This section contains an overview of the literature related to this paper. Since various aspects are touched, the section is split into metric learning and recommendation in general.

2.1.1. Deep Metric Learning

In typical machine learning tasks, experts had to manually extract handcrafted features. Deep neural networks allow us to learn representations automatically by embedding the features into a latent vector space. This approach has been applied successfully to a large variety of tasks, such as cyber security [21], medical image retrieval [22], face recognition [23, 24], speech recognition [25, 26], and bioinformatics [27, 28].

In the realms of video data, there exist success stories such as the retrieval of near-duplicate videos [29] or crowd counting and congestion level detection [30]. For video recommendations using multimodal features in particular, a combination of audio and video features was shown to be effective [19]. However, for the task of video understanding, convolutional neural networks (CNN) are often used in combination with recurrent neural networks (RNN), to capture temporal dependencies [31, 32, 33]. More recently, the transformer architecture is used, replacing previous RNNs with their attention mechanism [34], forgoing the CNN as a feature extractor.

2.1.2. Recommendation in General

Recommender systems in general are based upon the collaborative filtering idea, where user similarity plays an important role. There exists a substantial amount of literature for recommender systems — and rating prediction in particular — around the MovieLens[1] [1] benchmark, and to a lesser degree for the Netflix prize[2] [16]. Among the most prominent methods are matrix factorization [2, 35, 36] and autoencoders [37, 38], with new approaches joining in from other research fields, such as BERT4Rec [39] following the success for the transformer architecture in natural language processing. All these methods primarily use the rating history and only partially use user or item features, such as the user's age or the category of an item. However, progress has been made to add auxiliary features, such as text [40], images [41, 42] and audio [43, 44], utlizing the multimodal information.

2.2. Recommender Systems

2.2.1. General Recommender Problem

Datasets for recommender systems generally consist of the past interaction history of all users as a list of triplets in the form of $(user, item, rating)$. The triplets can be transformed into the user-item matrix, where each triplet corresponds to the $rating$ value at position $(user, item)$ and every other position is zero. The majority of user-item combinations will not have interacted, making this matrix highly sparse. An item refers to the object for which a user gave a rating, e.g. a movie, an e-commerce product or a travel destination.

Triplets

(U1, I1, 5.0)
(U1, I4, 3.0)
(U3, I2, 3.5)
(U4, I1, 4.0)
(U4, I2, 2.5)

User-Item Matrix

	Item 1	Item 2	Item 3	Item 4
User 1	5.0			3.0
User 2				
User 3		3.5		
User 4	4.0	2.5		

Fig. 1. Comparison of typical interaction triplets and the sparse user-item matrix. Colors indicate corresponding fields.

A visualization of both the interaction triplets and the corresponding user-item matrix is presented in Figure 1. Fur-

[1] https://grouplens.org/datasets/movielens/
[2] https://www.netflixprize.com/

thermore, the literature distinguishes between so called *implicit* and *explicit* feedback. Figure 1 shows explicit feedback, where ratings are given as floating point numbers in a fixed range. For explicit feedback, regression approaches are used to predict ratings. Implicit feedback however is binary, and indicates solely whether a user *has interacted with an item* or not. This type of feedback is generally used in online interactions, such as click-through-rates, where users do not typically give detailed feedback. Explicit feedback can be turned into implicit feedback, by considering only interactions above a certain threshold and omitting the remainder [38, 45, 46].

In order to give a simple example, we will consider a recommender system that suggests movies based on the previous ratings of a user. Many different approaches exist in the literature, the majority of which can be categorized as follows:

Collaborative Recommender System These recommender systems are based on the collaborative filtering technique, the idea that if *you* liked similar movies than *other users*, then suggesting movies they like has a high probability of success. Intuitively, this approach resembles recommendations made by a friend or family members that knows you and has a similar taste.

Content-Based Recommender System Instead of comparing *your* preferences against *other users'* preferences, content-based filtering aims at finding movies that are *similar to the movies you liked*. Intuitively, if you enjoyed the *Lord of the Rings* movies, then you are likely to enjoy the *Hobbit* movies from the same universe.

Session-Based Recommender System Sessions are in the *here and now* — they are modelled after the sequence of your recent interactions. A session could be the last five movies that you watched, all clicks since you opened a video streaming service, or which products you visited on an e-commerce platform. Such a session, unlike the previously discussed approaches, contains up-to-date information about a user's mood, which can influence the ideal recommendation substantially. Additionally, it allows models to focus on more recent signals, while older ones can be omitted entirely.

Hybrid Recommender System All approaches have their own benefits and limitations. Combining multiple approaches in a hybrid recommender system allows to take advantage of multiple benefits, while ideally overcoming, or at least mitigating, the limitations entirely.

2.2.2. Cold Start

According to Schafer et al. [47], there are three main cold start issues: (a) *new community*, (b) *new item* and (c) *new user*. Setting up a recommender system for a *new community* is by far the largest challenge, as ratings are missing for both users and items. They suggest mitigating this challenge by utilizing a heuristic for recommendations instead or applying otherwise available ratings — for example form a different community. However, this issue only arises once for the initial launch of a recommender system and it not typically an on-going issue. Adding a *new item* to a recommender system instead poses a more frequent risk. New items are often added during the lifetime of a recommender system, but they lack initial ratings and are not recommended until a critical mass of ratings is available. Additionally, items with very few ratings are further disadvantaged. For those reasons, it is difficult to recommend newly added items. Finally, *new users* are challenging, as generating personalized ratings without user specific data is impossible. Schafer et al. [47] propose different strategies here, such as asking new users for initial preferences or start by recommending popular items across the whole system. The nature of both users and items determines whether the *new user* or *new item* cold start problems deserve more attention, relative to their frequency of occurrence.

The various recommender system approaches described in the previous Section 2.2.1 all have their own distinct advantages in regards to the cold start problem. In particular, content-based filtering allows to recommend niche items due to the similarity to previously rated items and needs no information about other users. Furthermore, with content-based filtering it is easy to recommend new items, as long as they are similar *enough* to any other item in the system. However, it is impossible to present diverse recommendations, which are not similar to any of the user's rated items. Collaborative filtering on the other hand easily allows to explore new interests, as long as at least one other user interacted with the item. However, the cold start problem both for items and users makes it nigh impossible to generate recommendations. Our hypothesis is, that a hybrid combination could be able to gap the disadvantages of both individual filtering approaches.

2.3. System Architecture

The goal of this paper is to combine the building blocks described in previous sections as an end-to-end framework for multi-modal recommendations, using a combination of collaborative and content-based approaches. To achieve this, we employ a hybrid recommender system that combines a collaborative filtering approach with a content-based approach. For the collaborative half, we use one of the state-of-the-art methods called libFM [2], which is a matrix factorization method. For our content-based recommender, we generate predictions through a weighing scheme of the distance between movie embeddings. Both recommenders generate their individual ratings, which are then fused as a linear combination.

Figure 2 shows the overall end-to-end structure. Blue boxes represent the input data of both ML20M and ML20M-YT datasets to our system. Yellow boxes transform the data, such as turning individual movie trailer files into a collection of keyframes. The green boxes on the right side show the pre-

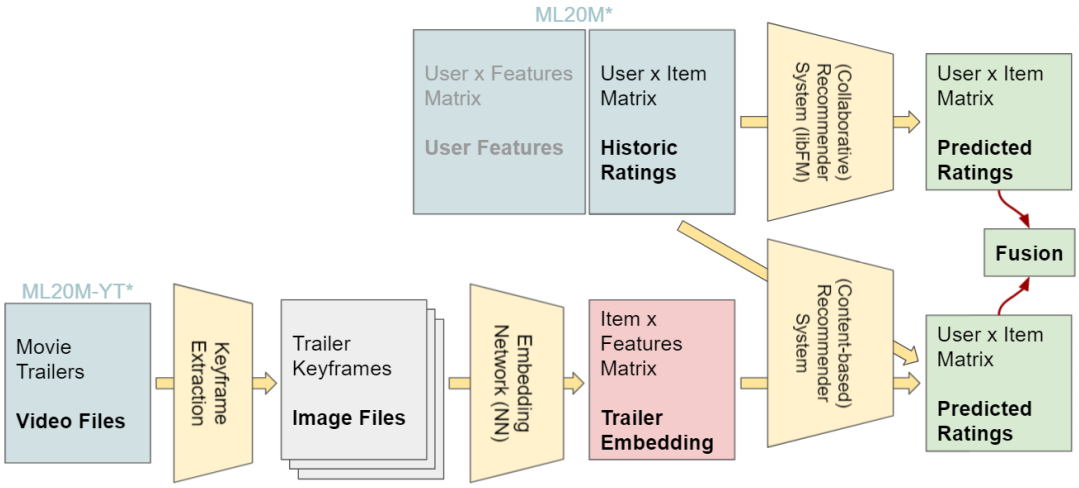

Fig. 2. Overview of the system architecture, showing data flow and individual components.
* Only a subset of ML20M and ML20M-YT is used, see Subsection 3.1.1 *Incomplete Dataset* for more details.

dicted ratings, which are fused to generate the final ratings.

Let us start with the upper half of Figure 2: All video trailers are downscaled to match a resolution commonly used by many pre-trained neural networks for visual tasks. For the *keyframe extraction*, we select every 25th frame of the video — corresponding to one frame per second. This is done to reduce the number of frames per video (averaging at around 5,000 frames) and reducing the overall size of the handled data. We do not show whether this is the best approach to take, but it could quickly and easily be implemented as a minimal working example and a closer investigation is out of scope for this paper. The extracted keyframes are then given to the *embedding network* as short sequences of keyframes. This network uses a pre-trained MobileNetV2 [48] as part of its backbone for visual feature extraction, followed by recurrent neural network (RNN) and fully connected layers. Various network heads are layered on top, such that during the training multiple tasks have to be solved simultaneously. By doing so, the intermediate fully connected layers should form a low-level, generalized representation of the movie trailers — the *trailer embedding*. These embeddings are extracted at inference time by removing the task heads from the network and exposing only the backbone (see Figure 3, the embedding network is discussed in more detail in the following paragraphs). Because the trailer is not given in full length to the network, but in short sequences, we generate a number of embeddings for each sequence. These sequence embeddings can be further processed into a single trailer embedding or used as a point cloud of embeddings. The *content-based recommender system* takes these embeddings and generates rating predictions by measuring the similarity of ML20M's movies according to the similarity of the trailer embeddings. For comparison, we also generate ratings from a *collaborative recommender system*, a state-of-the-art factorization machine called libFM [2]. Finally, the generated rating predictions from both recommenders are fused in a linear combination.

The main focus of this paper lies on the generation of a representative video embedding from the trailer video, which takes place in the embedding network. A more detailed overview of the network architecture can be seen in Figure 3. We use the MobileNetV2 [48] network with pre-trained weights on the ImageNet dataset [49], which enables the network to already start with decent low level visual features for use in the subsequent layers. We use sequences of images of size $S \in \{10, 20\}$, with subsequent sequence shifted over by 5 frames. These sequences are fed through the MobileNetV2 network, followed by two bi-directional LSTM layers with $2 * 256$ neurons. Two fully connected layers with 512 and 256 neurons complete the embedding network backbone.

During training, three network heads are used to perform simultaneous multi-task learning as a surrogate task. While our aim for the network is to construct a representative embedding for each trailer video, the network heads optimize three dissimilar tasks: *mean rating prediction*, *genre prediction* and *trailer ID prediction*. These tasks force the network to focus on different characteristics of the trailer, in order to succeed. The first head solves mean rating prediction, a regression task optimized by the mean squared error loss function. This head consists of three fully connected layers with 64, 32 and 1 neurons, respectively. The second head predicts the movie genre across 20 categories, such as action, animation, or romance. This task is both a multilabel and multiclass classification optimized by the binary crossentropy loss function, meaning a movie can have multiple genres at the same time. For this head, three fully connected layers are available with 128, 64 and 20 neurons, respectively. The third and last head learns to distinguish all trailers individually. This head takes up the majority of neurons, as the final layer has T neu-

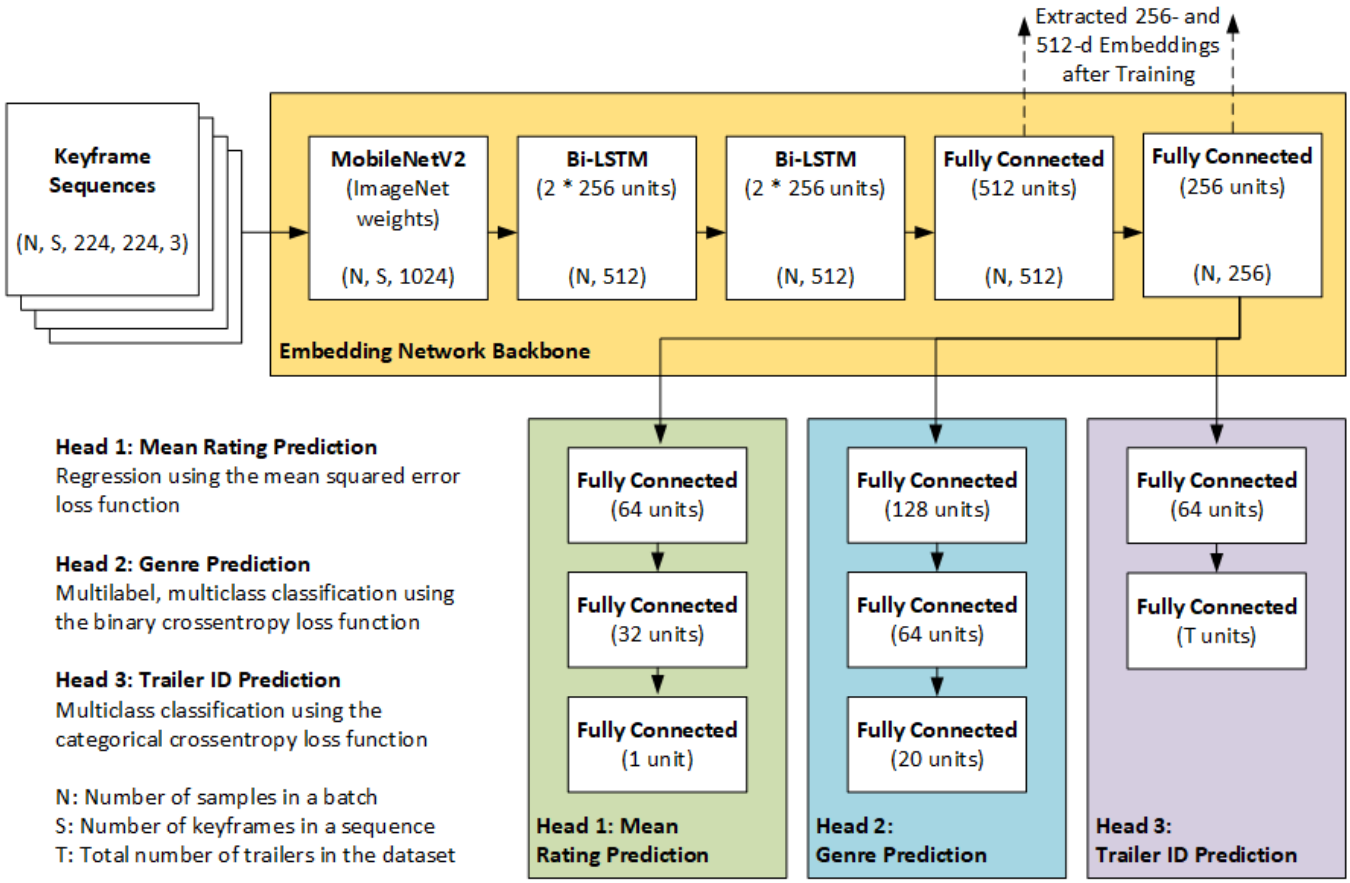

Fig. 3. Overview of the embedding network architecture and its training-only network heads.

rons, as many as there are different trailers in the dataset — on the order of 13,600 trailers. For this head, two fully connected layers are available with 64 and T neurons each. This head is optimized with the categorical crossentropy loss function.

All network heads are removed at inference time. Every sequence in the dataset is run through the backbone network, outputting the intermediate representations in the fully connected layers. The now extracted 256- and 512-dimensional embedding vectors however represent a sequence and not yet the full trailer. In order to compare different trailers and measure their similarity, we must first turn the set of sequence embeddings into a singular trailer embedding. For that purpose, we take the average of all sequences of every trailer.

2.3.1. Content-Based Rating Prediction

Our learned movie trailer embeddings alone only allow us to compare different movies and apply a pair-wise similarity or distance metric. To produce a numerical rating for an $(user, item)$ pair, we need to incorporate all previous ratings of a user. The trailer embedding then allows us to measure the similarity of all previously rated movies with the new movie.

Let us consider the situation shown in Figure 4: User 1 has rated movies A and B with 5.0 and 2.0, respectively. Now we want to predict his rating for Item D. We see that items A and D have a similarity of 0.92 (shaded in red), while items B and D are much less similar at 0.34 (shaded in blue). Our prediction is now a weighted average of user A's previous ratings, scaled by the similarity between movies. Because the similarities range between zero (indicating complete dissim-

HISTORY	Item A	Item B	Item C	Item D
User 1	5.0	2.0	-	?

SIMILARITY	Item A	Item B	Item C	Item D
Item A		0.05	0.37	**0.92**
Item B	0.05		0.15	**0.34**
Item C	0.37	0.15		0.62
Item D	**0.92**	**0.34**	0.62	

Fig. 4. Example data for the content-based rating prediction. The upper table depicts the rating history of User 1, the lower table the similarities between items A, B, C and D.

ilarity) and one (indicating that both items are identical), the similarity weights have to be normalized. The predicted rating for user 1 and item D of 4.19 is calculated as:

$$\frac{0.92}{0.92+0.34} \times 5.0 + \frac{0.34}{0.92+0.34} \times 2.0 = 4.19 \quad (1)$$

3. EXPERIMENTS AND RESULTS

3.1. Dataset

For the experiments, we use a publicly available benchmark dataset, MovieLens [1]. It is based around the ratings gathered on the equally named website[3]. Multiple variants of said dataset exists, varying in the total number of ratings used. In this paper we use the MovieLens 20M[4] (ML20M) variant, containing a total of 20 million user ratings on roughly 27,000 movies (or items), generated by 138,000 users. In terms of video data, a mapping exists from movie IDs in the ML20M dataset to YouTube IDs, the MovieLens 20M YouTube Trailers dataset[5] (ML20M-YT). Note that this mapping does not contain all 27,000 movies from the ML20M dataset.

3.1.1. Incomplete Dataset

While the ML20M dataset provides both user ratings, as well as features for users and items, the ML20M-YT dataset only provides a mapping from movie IDs to videos on YouTube. All trailers must be manually gathered from any of the available sources and were downloaded at 720p resolution.

ML20M-YT contains the ID mapping for 94% (25,141 out of 26,744) of the whole ML20M dataset. Our analysis showed, that more than 25% of all YouTube IDs (6,511 out of 25,623) are no longer available because they are either deleted or set to private. Due to the size of ML20M, we decided to continue with an as-large-as-possible subset of ML20M and ignore all missing trailers. In addition, the remaining trailer titles were examined for obvious errors, for example containing key phrases such as "part N of M" or "full movie"[6].

As shown in Figure 5, the majority of movie trailers lasts around two to three minutes. Note that the label "<3min" is relative to the previous, lower label "<2min" and only displays how many trailers last between two and three minutes. It is clear, that a trailer should capture the audience's attention quickly and act as a short summary of a movie. We decided to ignore any trailer with a duration of more than 10 minutes, as a duration above this threshold is unlikely to be a trailer[7].

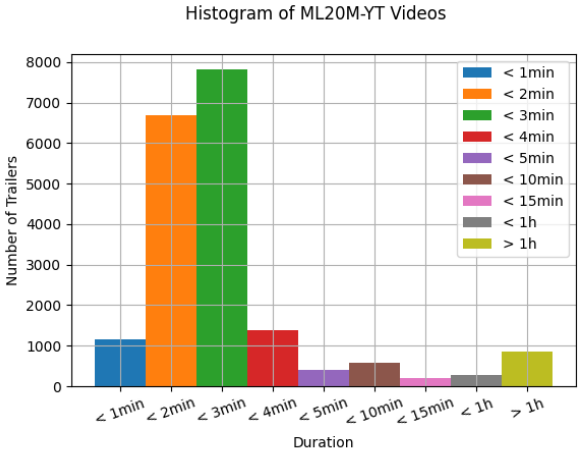

Fig. 5. Video duration histogram, showing the length of the publicly available trailers of the ML20M-YT dataset.

3.1.2. Rating Preprocessing

As discussed in previous section, a sizeable chunk of video trailers had to be removed from the dataset. As such, the ratings of the ML20M dataset are also filtered, removing any rating for discarded movies. Any user with fewer than $N = 5$ interactions after this step is also discarded, which is common practice in the literature [19, 38, 50, 51], potentially with even larger values of N. This step had no impact on the data.

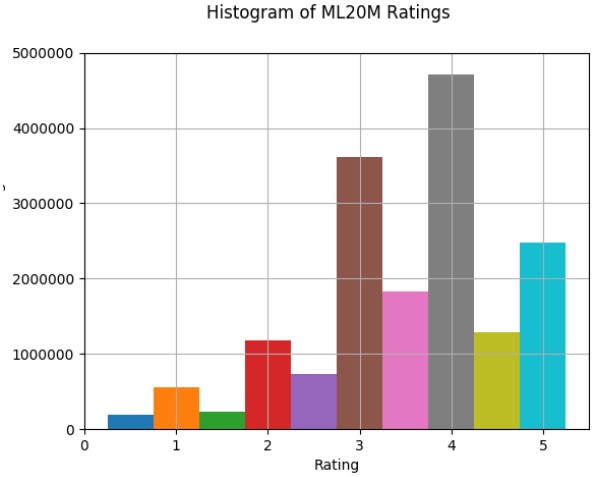

Fig. 6. Rating histogram, showing number of ratings for each distinct value in the ML20M dataset.

The ratings contain triplets of the form (user, item, rating), where *user* denotes the ID of a user, *item* corresponds

[3] https://movielens.org

[4] https://grouplens.org/datasets/movielens/20m/

[5] https://grouplens.org/datasets/movielens/20m-youtube/

[6] The necessary information about availability, duration and other relevant meta data of the ML20M-YT dataset can be found in our repository on GitHub: https://github.com/edualc/vt2_multimedia_recommendation/blob/master/youtube_extracted.csv

[7] The longest *trailer* in the ML20M-YT mapping lasts just shy of four hours, containing the full movie: https://www.youtube.com/watch?v=eJ3RzGoQC4s

Description	Sequence Length	Recommender Type	RMSE				MAE			
			Min	Max	Mean	Median	Min	Max	Mean	Median
libFM	—	collaborative	**0.7602**	1.1957	**0.8044**	**0.7608**	**0.5787**	0.9539	**0.6166**	**0.5790**
256-dim. embeddings	10 frames	content-based	0.9604	0.9624	0.9612	0.9612	0.7490	0.7504	0.7496	0.7495
512-dim. embeddings	10 frames	content-based	0.9598	**0.9618**	0.9606	0.9605	0.7485	**0.7499**	0.7491	0.7490
256-dim. embeddings	20 frames	content-based	0.9616	0.9636	0.9624	0.9624	0.7499	0.7513	0.7505	0.7505
512-dim. embeddings	20 frames	content-based	0.9617	0.9637	0.9625	0.9624	0.7500	0.7514	0.7506	0.7505
Movie meta data	—	content-based	0.9617	0.9637	0.9626	0.9625	0.7499	0.7514	0.7506	0.7505

Table 1. Results of the different recommender systems. Showing the performance of libFM, as well as our content-based approach utilizing the learned movie trailer embeddings for rating prediction on the filtered ML20M-YT ratings.

	ML20M	ML20M-YT	
		Original	Filtered
No. of Movies	26,744	25,141	13,571
No. of Users	138,493	138,493	138,493
No. of Ratings	20M	19,97M	16,77M
Mean Rating	3.526	3.525	3.538
Median Rating	3.5	3.5	4.0

Table 2. Statistics of the MovieLens 20M dataset variations.

to the MovieLens ID of a particular movie and the *rating* is a floating point number in the range of 0.5 to 5.0 in steps of 0.5. As Figure 6 shows, there is a strong bias towards whole-numbered ratings. This might be an artefact from combining different rating schemes, where only whole numbers were available (for example one to five stars). Additionally, the majority of ratings lies between 3 and 5, with a mean rating of 3.538 and a median rating of 4.0. A numerical comparison between the original ML20M dataset, the original ML20M-YT and the filtered ML20M-YT datasets is shown in Table 2. Missing movies in the original ML20M-YT dataset barely have an impact on the overall number of ratings compared to ML20M, even though 6% of movies were removed. However, using the filtered ML20M-YT variant where only available movie trailers are considered, the number of ratings drops by about one sixth from 20 to 16,77 million ratings. Even though half the movies are missing, all users remain. Naturally, this reduction in ratings has an impact on the mean and median ratings — with the median rating jumping from 3.5 to 4.0.

3.2. Training Details

In our rating experiments, we use a 10-fold cross-validation on the filtered subset of the ML20M ratings. For the training of the embedding networks, we use a batch size of 32 in conjunction with the Adam optimizer [52], using default parameters with a learning rate of 3e-4. The models were trained for a maximum of 16 epochs, with early stopping terminating after a lack of improvement over 2 epochs. Most models converged between 8 and 9 epochs. One epoch took on average 20 hours, with the 10-frame sequence models training up to 10% faster than the 20-frame sequence models. Grid search was employed to find more suitable hyperparameters, however, no deviations from the initial parameters resulted in a statistically significant performance increase. The code for all our experiments can be found in the GitHub repository[8].

3.3. Recommender Performance

Table 1 shows the performance of the different recommender setups used in this paper. The libFM [2] system performed mostly as expected and we were able to match libFM's fine-tuned performance on MovieLens 10M of 0.749 RMSE, which is in the region of our unoptimized 0.761 RMSE on a similar — but larger — dataset. Note however, that our filtered version of ML20M and ML10M are different. Furthermore, we were able to see libFM fail in one of the cross-validation folds with a 1.196 RMSE, performing vastly inferior to the otherwise very stable and robust measurements around the median. Re-running this fold yielded a similar outlier performance and we could not find an easy explanation as to why libFM was unable to fit this particular fold.

Taking a look at the content-based results using our embeddings in Table 1. Since we extracted embeddings both from the last and second to last layer of the backbone embedding network (see Figure 3), there are two embedding sizes of 512 (second to last layer) and 256 (last layer). The sequence lengths are set to 10 and 20 keyframes, corresponding to 10 and 20 seconds, respectively. As Table 1 clearly shows, there is no statistically significant performance difference between both embedding sizes, nor when comparing the sequence lengths. With these embeddings alone, we are unable to outperform libFM. Ignoring the outlier of libFM, the content-based recommender using the extracted embeddings performs 0.2 RMSE worse and have a 0.17 higher MAE.

The performance of the linaer combination of the predicted ratings is shown in Figures 7 and 8 in terms of RMSE and MAE, respectively. The visualized data is from a 10-fold cross-validation. The orange line depicts the mean, the red line the median across the ten folds. Shaded areas indicate the percentile ranges from the 10th to the 90th percentile in

[8] https://github.com/edualc/vt2_multimedia_recommendation

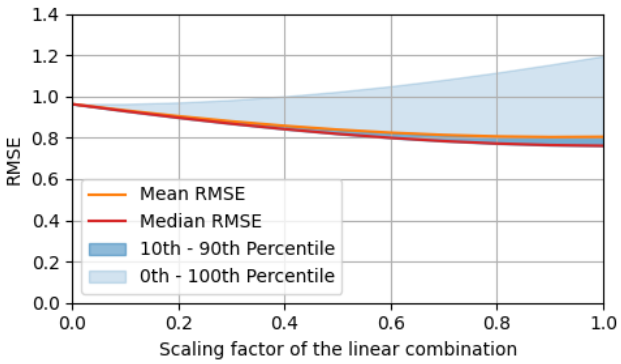

Fig. 7. RMSE performance of the linear rating combination.

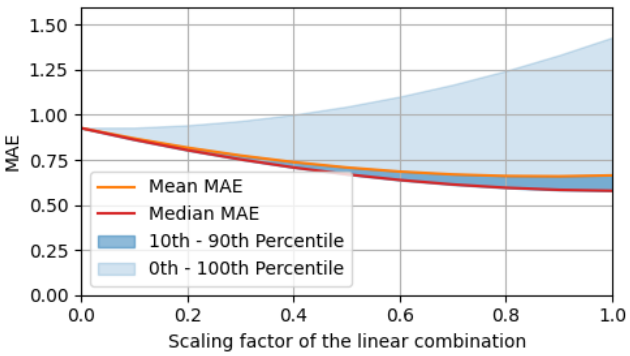

Fig. 8. MAE performance of the linear rating combination.

dark blue, and from the 0th to the 100th percentile in light blue. The x-axis indicates how the ratings are generated, with a scaling factor $k \in [0, 1]$. For every $(user, item)$-pair in the test set, the predicted rating r_{new} is calculated as follows:

$$r_{new} = k \times r_{libFM} + (1 - k) \times r_{Ours} \quad (2)$$

Due to the outlier behavior of libFM on one out of ten cross-validation folds, the average seems to point towards an optimal combination of around $0.8 \times r_{libFM}$ and $0.2 \times r_{Ours}$. However, this holds true only for the particular fold where libFM failed. In all other cases, a linear combination of the predicted ratings is performing strictly inferior.

However, one might argue this is an unfair comparison: Firstly, libFM is a collaborative filtering approach taking into account the similarity between *users and all their ratings*, while we perform a content-based filtering that only uses the similarity of *movie trailers* and for every user only *his own ratings*. Secondly, there is a disconnect between a trailer and the actual movie. A trailer is — after all — a marketing tool to motivate the audience. If or how people enjoy the movie can be influenced by a large number of factors unrelated to the enjoyment of the trailer. Thirdly, if two things are *too similar*, we humans can start to perceive them in a negative way. This idea is called the "uncanny valley" — the more similar two things become, the more familiar those objects seem. However, there comes a point where being *too similar* can become something negative [53]. This drop is called the uncanny valley and might explain partially why a user does not rate movies highly that are similar to his preferred movies.

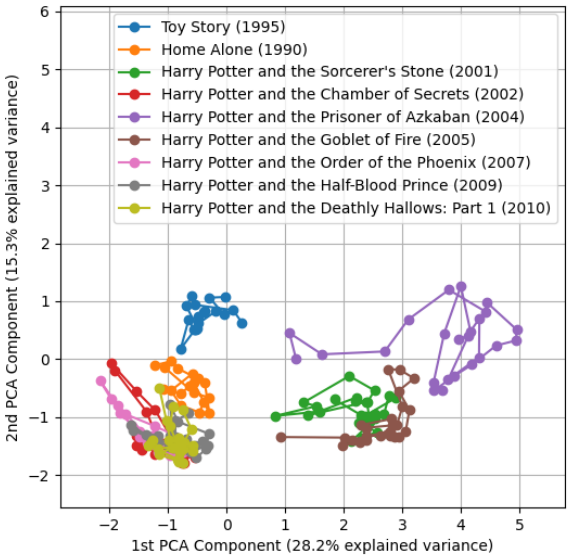

Fig. 9. PCA visualization of extracted sequence embeddings, showing the different movies of the "Harry Potter" saga.

3.4. Embedding Quality

Figure 9 shows the extracted sequence embeddings, decomposed into their first two principal components, which are able to explain 28.2% and 15.3% of the variance in the data, respectively. The movie trailers picked in this Figure are from the movies "Toy Story", "Home Alone" and all seven parts of the "Harry Potter" saga. Each color corresponds to one trailer and every individual dot is the extracted embedding vector from a sequence of 20 keyframes. Connecting lines indicate how the embedding changes throughout the trailer, as sequences were generated in a sliding window approach. If two points are connected through a line, the sequences are overlapping and share the majority of their keyframes. Additional PCA visualizations can be found in Appendix A.1 for different groups of movies, with "Toy Story" and "Home Alone" added as a constant reference across all PCA figures.

Clusters in Figure 9 tend to be clumped together, indicating that taking two different sequences from the same trailer results in two sequence embeddings in vicinity. However, these trailers are downloaded from the YouTube platform and there exist cases, where an additional end screen was added by

the person who uploaded the movie trailer. An example of this is the trailer for "Harry Potter and the Prisoner of Azkaban"[9], where the Movieclips channel added an advertisment for their own website (see the leftmost five dots of the purple cluster in Figure 9). Furthermore, we could not find an obvious reason why four of the Harry Potter movies cluster within proximity on the bottom left corner in Figure 9, while the remaining three movies cluster on the bottom right side. Knowing the contents of the movies, an argument could be made that these movies *grow up* over time: harmless wizardry in the first few movies slowly building up to a more horror-inspired action franchise. However, the second movie "... and the Chamber of Secrets" (in red) is located close to the final few movies (in grey and olive), outright disproving the argument. The same grouping into the bottom-left and bottom-right "superclusters" can be observed for other groups of movies as well (e.g. "Lord of the Rings" and "Star Wars" in the Appendix A.1), and even for random selections of movie trailers.

3.5. Correlation Analysis

Assuming we take the mean rating for every movie in the dataset, are we able to reasonably estimate it just from the information in our generated embeddings? Since every user has his own bias, our hypothesis is that the mean rating averages out these biases and is a good proxy for a user-agnostic rating. Our embeddings are also free of any direct user information, as they are solely trained on the information contained in the movie trailers. The question now becomes, is there enough information in the data on a movie trailer content level, to reasonably estimate the mean rating of a movie?

Independent Variable X	R	R^2
Trailer Embeddings	0.405	0.164
Views, Likes & Dislikes (log.)	0.306	0.094
Like/Dislike-Ratio (log.)	0.290	0.084

Table 3. R and R^2 values for mean movie rating prediction.

To answer this question, we use a linear regression where the independent variable X is the matrix containing all embeddings for every movie in the dataset and the dependent variable is y a vector containing all mean ratings. The linear regression model predicts the mean rating $\hat{y}$ with an R^2 value of 0.164, indicating a very weak — if any — correlation. Similarly, if we instead replace the independent variable X with the logarithm of the number of views, likes and dislikes on YouTube, we end up with an R^2 value of 0.093. Finally, the ratio between likes and dislikes measures R^2 if 0.084. Table 3 shows the R and R^2 values in relation to the chosen independent variable. For reference, figures showing the fitted prediction and the residuals are added in the Appendix A.2.

Depending on the field of research, different thresholds are chosen for the interpretation of the R^2 value. Hair et al. [54] recommend *"R^2 values of 0.75, 0.50, or 0.25 for endogenous latent variables in the structural model can be described as substantial, moderate, or weak, respectively"*. Applying their rule of thumb in our situation, we measure well below the threshold for a weak correlation for all choices of X.

This experiment indicates that there is a lack of relevant information to learn even just the mean rating for movies from neither the trailer embeddings nor likes, dislikes and views. In addition, we argue that the lack of correlation explains partially why generating predictions with the content-based recommender from the movie meta data alone yields a similar performance compared to using the trailer embeddings.

4. CONCLUSION

In this paper, we introduced an end-to-end framework for generating rating predictions using multimodal features — the movie trailers. We applied this framework to the ML20M dataset, but were unable to outperform current state-of-the-art. We outlined three arguments, why the performance of the content-based recommender was unable to match the collaborative filtering approach. Furthermore, we showed how incomplete and error-prone the ML20M-YT dataset is, and which steps can be taken to mitigate many of its flaws.

4.1. Future Work

While we were unable to produce ground-breaking new results, we believe this area of research is not yet well explored and many questions remain unanswered. Our framework serves as a proof of concept, but many of the individual sections were left at the stage of a minimal working example.

Firstly, we downscale the HD resolution videos to a 224 by 224 square image. Through this scaling, we might be losing a large chunk of the information embedded in the trailers. Secondly, the keyframe extraction takes frames at uni-distant intervals. While easy to implement, this approach does not intuitively seem great. Instead, it would be interesting to see, if a small fraction of all frames can be selected that represents the full movie trailer. Thirdly, the embedding network could be replaced with the modern transformer architecture, utilizing the power of the attention mechanism. Transformers have been applied in more general research problems, far from its original emergence in NLP. We believe this generalized architecture could be able to better model the unique representations. Furthermore, the learned attention might contain valuable information about which parts of a trailer are more or less relevant. Fourthly, the surrogate task of learning intermediate representations does not fully align with the goal of learning intermediate, generalized representations. A training paradigm using a triplet loss function might better suited to this task, as the training would directly optimize the embeddings. Fifthly, collaborative filtering approaches are dominating the current research trends for recommender systems.

[9] `https://youtu.be/lAxgztbYDbs` (from 1:48 onwards)

Fusing the multimodal features with a collaborative filtering recommender system at a low level should be a next step.

Choosing the MovieLens dataset was correct, as a lot of literature is available for this particular dataset and it is well understood. However, the trailers proved to be a challenge and the quality of the ML20M-YT dataset left much to be desired. While it is reasonable to expect a certain degree of unavailability of videos referenced on YouTube, many of the references point towards obvious non-trailer videos. We believe a rigorous check of these references is necessary if further research should be concluded on the ML20M-YT dataset.

Out of all the multimodalities available, our paper focused on videos. However, there are many different modalities available — even just on MovieLens alone. We believe there is a lot of potential for the individual modalities, but even more so for combinations — similar to the idea of ensembles.

5. REFERENCES

[1] F Maxwell Harper and Joseph A Konstan, "The MovieLens datasets: History and context," *Acm transactions on interactive intelligent systems (TIIS)*, vol. 5, no. 4, pp. 1–19, 2015.

[2] Steffen Rendle, "Factorization machines with libFM," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–22, 2012.

[3] Elaine Rich, "User modeling via stereotypes," *Cognitive science*, vol. 3, no. 4, pp. 329–354, 1979.

[4] Jussi Karlgren, "An algebra for recommendations: Using reader data as a basis for measuring document proximity," 1990.

[5] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994, pp. 175–186.

[6] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas, "Recommending and evaluating choices in a virtual community of use," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 194–201.

[7] Upendra Shardanand and Pattie Maes, "Social information filtering: Algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 210–217.

[8] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter, "PHOAKS: A system for sharing recommendations," *Communications of the ACM*, vol. 40, no. 3, pp. 59–62, 1997.

[9] Sergey Brin and Lawrence Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "ImageNet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[11] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le, "Meta pseudo labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11557–11568.

[12] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le, "Self-training with noisy student improves ImageNet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10687–10698.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

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

[15] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[16] James Bennett, Stan Lanning, et al., "The netflix prize," in *Proceedings of KDD cup and workshop*. New York, NY, USA., 2007, vol. 2007, p. 35.

[17] Steffen Rendle, Li Zhang, and Yehuda Koren, "On the difficulty of evaluating baselines: A study on recommender systems," *arXiv preprint arXiv:1905.01395*, 2019.

[18] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 335–344.

[19] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Natsev, "Collaborative deep metric learning for video understanding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 481–490.

[20] Salma Karray and Lidia Debernitz, "The effectiveness of movie trailer advertising," *International Journal of Advertising*, vol. 36, no. 2, pp. 368–392, 2017.

[21] Giuseppina Andresini, Annalisa Appice, and Donato Malerba, "Autoencoder-based deep metric learning for network intrusion detection," *Information Sciences*, vol. 569, pp. 706–727, 2021.

[22] Aoxiao Zhong, Xiang Li, Dufan Wu, Hui Ren, Kyungsang Kim, Younggon Kim, Varun Buch, Nir Neumark, Bernardo Bizzo, Won Young Tak, et al., "Deep metric learning-based image retrieval system for chest radiograph and its clinical applications in COVID-19," *Medical Image Analysis*, vol. 70, pp. 101993, 2021.

[23] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[24] Dingyi Zhang, Yingming Li, and Zhongfei Zhang, "Deep metric learning with spherical embedding," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[25] Jixuan Wang, Kuan-Chieh Wang, Marc T Law, Frank Rudzicz, and Michael Brudno, "Centroid-based deep metric learning for speaker recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3652–3656.

[26] Joon Son Chung, Jaesung Huh, Seongkyu Mun, Minjae Lee, Hee Soo Heo, Soyeon Choe, Chiheon Ham, Sunghwan Jung, Bong-Jin Lee, and Icksang Han, "In defence of metric learning for speaker recognition," *arXiv preprint arXiv:2003.11982*, 2020.

[27] Mark Lennox, Neil Robertson, and Barry Devereux, "Deep metric learning for proteomics," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 308–313.

[28] Shaoliang Peng, Lei Zhang, Yaning Yang, Wei Liu, Fei Li, Hao Hong, Kenli Li, and Shulin Wang, "A deep metric learning algorithm for similarity measure of the gene expression profile," in *2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM)*. IEEE, 2021, pp. 1–6.

[29] Giorgos Kordopatis-Zilos, Symeon Papadopoulos, Ioannis Patras, and Yiannis Kompatsiaris, "Near-duplicate video retrieval with deep metric learning," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 347–356.

[30] Qi Wang, Jia Wan, and Yuan Yuan, "Deep metric learning for crowdedness regression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2633–2643, 2017.

[31] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International conference on machine learning*. PMLR, 2015, pp. 843–852.

[32] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz, "Multilayer and multimodal fusion of deep neural networks for video classification," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 978–987.

[33] Yi Bin, Yang Yang, Fumin Shen, Ning Xie, Heng Tao Shen, and Xuelong Li, "Describing video with attention-based bidirectional LSTM," *IEEE transactions on cybernetics*, vol. 49, no. 7, pp. 2631–2641, 2018.

[34] Gedas Bertasius, Heng Wang, and Lorenzo Torresani, "Is space-time attention all you need for video understanding?," *arXiv preprint arXiv:2102.05095*, 2021.

[35] Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[36] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen, "Deep matrix factorization models for recommender systems.," in *IJCAI*. Melbourne, Australia, 2017, vol. 17, pp. 3203–3209.

[37] Xiaopeng Li and James She, "Collaborative variational autoencoder for recommender systems," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 305–314.

[38] Ziwei Zhu, Jianling Wang, and James Caverlee, "Improving top-K recommendation via joint collaborative autoencoders," in *The World Wide Web Conference*, 2019, pp. 3483–3482.

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[40] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 233–240.

[41] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 391–400.

[42] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley, "Visually-aware fashion recommendation and design with generative image models," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 207–216.

[43] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen, "Deep content-based music recommendation," in *Neural Information Processing Systems Conference (NIPS 2013)*. Neural Information Processing Systems Foundation (NIPS), 2013, vol. 26.

[44] Jongpil Lee, Kyungyun Lee, Jiyoung Park, Jangyeon Park, and Juhan Nam, "Deep content-user embedding model for music recommendation," *arXiv preprint arXiv:1807.06786*, 2018.

[45] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng, "Collaborative competitive filtering: learning recommender using context of user choice," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 295–304.

[46] Santosh Kabbur, Xia Ning, and George Karypis, "FISM: factored item similarity models for top-N recommender systems," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 659–667.

[47] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, "Collaborative filtering recommender systems," in *The adaptive web*, pp. 291–324. Springer, 2007.

[48] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[49] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[50] Maksims Volkovs and Richard Zemel, "Collaborative ranking with 17 parameters," *Advances in neural information processing systems*, vol. 25, pp. 2294–2302, 2012.

[51] Chaosheng Fan, Yanyan Lan, Jiafeng Guo, Zuoquan Lin, and Xueqi Cheng, "Collaborative factorization for recommender systems," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 949–953.

[52] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] Karl F MacDorman and Hiroshi Ishiguro, "The uncanny advantage of using androids in cognitive and social science research," *Interaction Studies*, vol. 7, no. 3, pp. 297–337, 2006.

[54] Joe F Hair, Christian M Ringle, and Marko Sarstedt, "PLS-SEM: Indeed a silver bullet," *Journal of Marketing theory and Practice*, vol. 19, no. 2, pp. 139–152, 2011.

A. APPENDIX

A.1. PCA Embedding Visualizations

In this section, additional PCA visualizations of the extracted sequence embeddings are shown. The PCA uses only the first two principal components and each color represents all sequence embeddings of a movie trailer. The explained variance from the first two principal components is 43.5%.

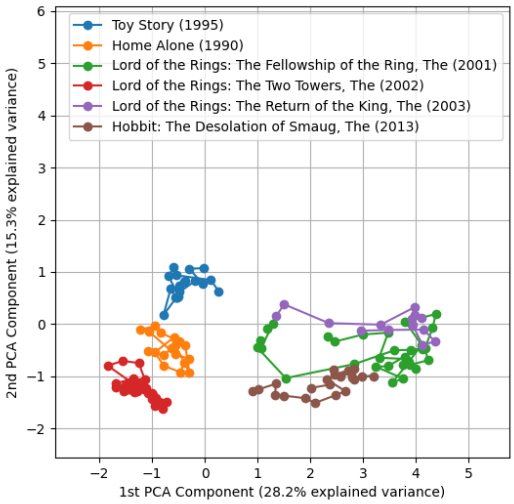

Fig. 10. PCA visualization of extracted sequence embeddings, showing different movies from the "Lord of the Rings".

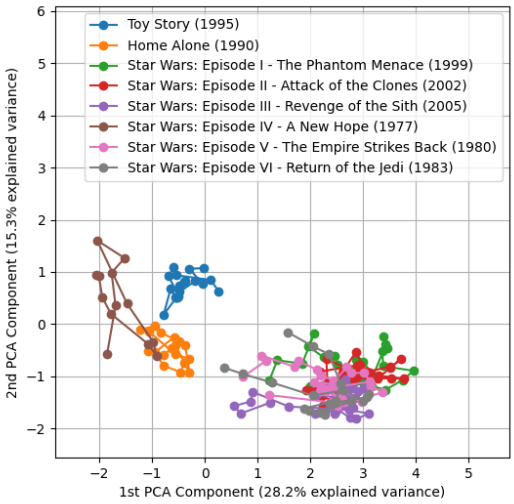

Fig. 11. PCA visualization of extracted sequence embeddings, showing movies from both "Star Wars" trilogies.

A.2. Linear Regression Fit and Residuals

To further visualize the points raised in Section 3.5, we show an example of the applied linear regression. The independent variable X is the like/dislike-ratio for the movie trailers on YouTube and the fitted prediction is the mean rating $\hat{y}$, across all ratings for each movie in the filtered ML20M-YT dataset.

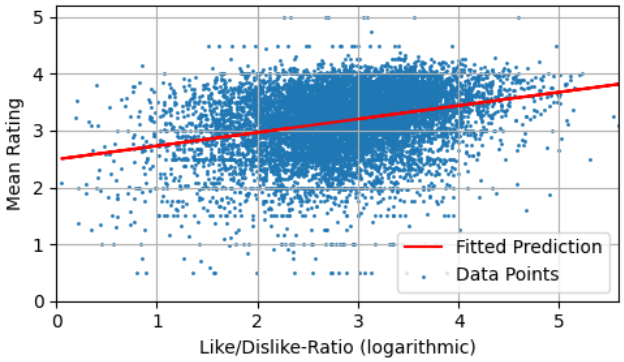

Fig. 12. Linear regression prediction of the mean rating, where the like/dislike ratio is the dependent variable.

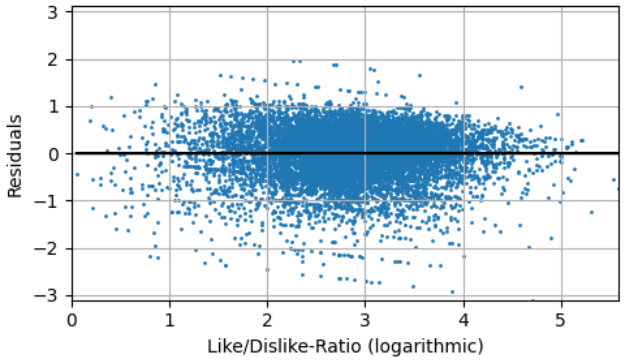

Fig. 13. Residual plot of the linear regression, fitted on the relation between the like/dislike-ratio and the mean rating.