Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Deep collaborative filtering models with audiovisual content-aware algorithms for movie recommendations

Author:
Robert Lewis

Supervisor:
Ognjen Rudovic

Submitted in partial fulfillment of the requirements for the MSc degree in Computing Science of Imperial College London

September 6, 2019

Abstract

In this work we propose a way to enhance the recommendation capabilities of collaborative filtering algorithms for movie recommendations by using the audiovisual content of movie trailers.

Our approach makes the collaborative filtering algorithms significantly more effective at making recommendations in the item cold start scenario as judged by a broad suite of evalutation metrics. Moreover, our approach shows promise at improving the quality and diversity of recommendations in the item warm start scenario.

Our approach centers around a deep learning architecture which combines the state of the art in recommender system matrix factorisation techniques with leading techniques in video content interpretation and learning.

Moreover, the work for this project has resulted in the construction of a flexible audiovisual content feature extraction pipeline that is capable of efficiently processing arbitrarily large video datasets. For the purposes of this work we used this pipeline to extract audiovisual feature descriptors for over 12,000 movie trailers (totalling more than 650 hours of content).

Acknowledgements

I would like to thank Oggi for his insight and for the infectious enthusiasm he showed for difficult problems throughout the project. I would also like to thank my partner Stella and my family who were tremendously supportive as always.

Contents

List of Figures

1 Introduction

In this section we introduce the research direction and compelling commercial applications of our work. Then, in Section 2 we outline the relevant theory and related work - including a consideration of the ethics of our work - before discussing our proposed extension to this in Section 3.

We then present our experimental approach and results in Section 4 before discussion their implications in 5. Finally, in Section 6 we summarise the conclusions from our findings and discuss interesting avenues of future work.

1.1 The case for audiovisual content-aware recommender systems

The case for recommender systems generally is well-known[12]. In an age of online information and content overload, users require the assistance of an information-retrieval device to help them track down the content that matters most to them in a reasonable time[54].

The longer this process takes, and the less personalised it feels, the more it reduces a user's satisfaction - leading them to conclude that there is nothing in the company's catalogue that appeals to them - and thus increasing the risk that they will *churn*[1] from the platform or service. In commercial terms this translates to a loss in revenue for the company.

Therefore, recommender systems are now an integral part of corporate strategy for online content providers and retailers. The online streaming service Netflix believe that the combined effect of personalisation and recommendations save them more than $1B per year[33]; similarly, it has been estimated that 30% of Amazon.com's page views result from recommendations[45, 2].

To achieve such numbers recommender systems address various issues throughout a company's business. For example, they increase customer acquisition and retention by personalising the list of items they recommended to any given user and in doing so increasing user satisfaction and fidelity.

Recommender systems also help a company grow its sales and brand by recommending more diverse items from its product catalogue (including seemingly unpopular ones to specific users - a calculated risk that may not have been possible without the personalised analysis of a recommender system). Moreover, by providing an insight into what the customer base - both individually and collectively - enjoys in items, they can play an important role in areas such as new content generation and new

[1]The churn rate of a customer base refers to the proportion of contractual customers or subscribers who leave a supplier during a given time period. It directly affects a company's *recurring revenue* and thus it is of utmost importance to reduce the churn rate if a company is to grow profitably.

market entry.

The case for *audiovisual content-aware* recommender systems - and the focus of this research - is perhaps less apparent given its novel nature. As such we offer our perspectives on how such a system may contribute to the recommender systems already employed at online streaming service providers.

First and foremost, the audiovisual content of an item is perhaps the most important feature in determining how a user responds to it. Thus, by building a system that takes it into account a company should be able to more precisely recommend items that match the audiovisual aesthetic or sensory experience their customers are looking for.

The value of such a system is increased when one considers a common scenario for items - the *item cold start scenario* where items are introduced to the catalogue and have no (or very few) user reviews nor other forms of feedback (such as tagging or viewing history).

With current *collaborative filtering* based recommender systems (defined in Section 2.1.2) there is a large risk that such items are overlooked permanently due to effects such as *popularity bias*[2]. However, the audiovisual content of a media item is present from day one and thus if one can build a model that relates it to user review - and thus is able to recommend it - then the *item cold start* issue can be addressed.

Thus, we see mitigating the negative economics of such a scenario as the primary commercial application of our work. However, there is more. While it is tempting to see recommender systems as purely a content recommendation/retrieval system, this is shortsighted: recommender systems should be seen as a powerful tool throughout the entire lifecycle of a product from content ideation through to the delivery of the final content to a user.

On what grounds can we justify such a statement? The central premise of this statement is that by learning how low-level item content relates to user review a recommender system can provide insight on whether or not a user is going to like an item **even before the item is finalised**. Therefore, the recommender system has a central and active *creative director* style role to play in the marketing, creation and even initial proposal of the content.

Proposing tangible use cases is the best way to elaborate on this argument

- *How should we promote existing content to target users?*

[2]Popularity bias refers to the phenomenon in recommender systems that popular items are recommended frequently while less popular, niche products, are recommended rarely or not at all[36]. It is very notably an issue for *collaborative filtering* systems that rely on exploiting similarities between item and user review histories to generate new recommendations.

  - Here the system could assist in the trailer creation stage of the production process. By this point, the movie itself is usually cut and thus creating a trailer is about finding a way to summarise the content in a way that generates the broadest user appeal.
  - An audiovisual content enhanced recommender system contains information on every user's individual reactions to different content styles and, moreover, it provides an accurate feedback mechanism on whether a user will like an item's content that does not require having to actually contact or interview the user
  - Thus, there is a conceivable scenario where a production studio could iterate versions of a film's trailer, each time running it through the recommender system to gauge the expected review/engagement it is likely to receive, and then re-cutting it to improve on these metrics.
  - More powerful still is the personalised, generative scenario where the recommender system is set up in such a way that it can select (or even produce) a personalised trailer for each user[3]. In their simplest form these trailers may contain different scenes from the movie containing content that each user would enjoy/engage with most

- *How should we produce original content that appeals to target users?*
  - Here the system contributes to the actual content creation process. Again, it can be used purely for feedback - for example, generating a score for each scene - or as an ideas generator - proposing what type of scenes would make the movie appealing to a broader or specific user base.
  - The fact that a company could also combine the recommender system in this creative guru guise with the customer churn analysis generated by its strategy team adds further weight to this use case
  - For example, the churn analysis would identify groups of users who are at risk of churning. The recommender system could then run an analysis of what type of content these users like and a gap analysis between this and the company's current content catalogue could be performed. This gap analysis, combined with the system's generative ability, would then allow the system to advise on what scenes to include in content already being produced, or indeed what new content ideas to consider, with the explicit goal of engaging these users and mitigating their risk of churn

- *What would be the expected return of producing content like this?*
  - A final potential use case of a content-aware recommender system is as a financial analysis aide to an executive who decides whether or not a piece of content gets produced (or gets another series). As such, it informs

[3]*Generative* in this context refers to the capacity of the recommender system to generate candidate trailers from the film's content. In related generative work, Netflix has already released a way of personalising the artwork used to promote an item to each user[27].

the executive on whether or not an item has significant enough *return on investment* capacity to merit the investment of a production budget.

- Here the introduction of additional content modalities such as the movie script are critical as the content is just an idea at this stage and thus has little or none audiovisual content (although if we extend the remit of the system to the music industry then the audio capabilities would be useful at this very early stage).
- Then, in this role, the recommender system could provide an estimate of the review each user would give to the movie script if it was produced in its current form. Aggregate financial analysis could be performed on this - for example, estimating that X% of the company's user base would consider the movie at 4 stars or above and are thus likely to watch it - to arrive at an expected revenue figure.
- The executive could then weigh off this expected revenue with expectations on the cost to produce the movie to arrive at a decision on whether or not to produce it. Once again, a generative version of the system could make proposals on what the script is missing in its current state and this could fuel an iterative process with the writer.

It should be noted that our experimental focus in this project is on how our audio-visual content enhanced recommender system can address *item cold start* issues for items that already exist. However, in our future work section (see Section 6.2) we provide proposals on how the work could be extended to address these additional and exciting use cases.

1.2 Research overview

In this project we structured our work in such a way so as to address the following research questions

- **RQ1.** Does the audio-visual content of movie trailers enhance the capabilities of recommender systems in a) the *item warm start* scenario and b) the *item cold start* scenario?
- **RQ2.** How can audio-visual content be summarised to enable its use in CF-based recommender systems?

Taking each of these questions in turn, **RQ1** allows us to investigate the interplay between a movie's content and the user's response to this. Our in-going hypothesis is that audio-visual content should improve the quality of the recommender system - both in terms of recommendation accuracy and other recommender system metrics (such as the diversity of items recommended).

Our favoured approach is to look for ways to enhance existing *collaborative filtering* models as these achieve state of the art results in *warm start scenarios*[11, 40] yet at the same time have a severe limitation in the *item cold start* scenario that our content aware approach may help them mitigate.

Our choice of using movie trailers over the actual movie itself is predicated on our proposal that the trailer of a movie is probably the single most decisive factor in determining whether or not a user will watch a new movie and that a trailer is highly representative of the movie's overall content. This latter premise is supported by the literature, where it has been reported that the audiovisual features of a movie's trailer correlates heavily with the audiovisual features of the actual movie itself[57].

RQ2 is an essential prerequisite to enable investigation of **RQ1.** but is by no means a trivial task. Here our goal is to find a representation of the audiovisual content that is closely aligned with the way that a user will perceive it. Thus, our representation must be both i) *perceptually complete*[4] and ii) *semantically meaningful*[5].

Working with the raw audiovisual features is not an option due to the *curse of dimensionality* - a phenomenon where increasing dimensionality increases the gap between data points in the feature space, and thus similar features are not actually that close together and are thus hard to identify. This curse also greatly increases computation times and memory requirements and thus it is not desirable from an engineering perspective either.

Therefore, our goal in this piece of the work is to find a low-dimensional representation of the audiovisual content that preserves the requirements i) and ii) of good feature representations discussed above.

[4]*Perceptually complete* means that the representation should contain at least one representation for each of the media through which a user will experience the trailer - in our case audio and visual

[5]*Semantically meaningful*[1, 26] feature extraction refers to the process of finding a lower-dimensional representation of the original input within which we can assign/infer meaning (i.e. semantics) to/from different input instance values, and that these assigned/inferred semantics are relevant to the target domain - in other words they provide meaningful variance in the input data that explains different results in the target domain.

2 Theory and related work

2.1 Recommender Systems

2.1.1 Formulating the recommender system problem

Before venturing into our discussion on the various ways to build recommender systems, it's important to first define exactly the problem we are trying to solve. In Section 1, we outlined why we need recommender systems and how they are used, but what precisely is the underlying problem we are asking them to model?

Broadly speaking there are two ways of formulating the recommender system problem

1. Recommender systems as a **rating prediction** problem - aka the *matrix completion problem*: in this view of the problem, we are presented with an incomplete set of user ratings for items as our training data and our target is to predict the missing ratings for items specific users have not yet reviewed. Here the entity using the recommender system is asking the question *exactly how much will a user like this item?*

2. Recommender systems as a **ranking prediction** problem - aka the *top-k recommendation problem*: in this alternative view of the problem, the goal is to predict a user's relative preference for items in a list. As a result, it is not essential to predict the users' ratings for items (though the model still learns from user ratings in either explicit or implicit form). Here the entity using the recommender system is asking the question *what list of items should I recommend to a user?*

Both approaches are prevalent in recent literature[4] and it's important to note that the first formulation of the problem can be easily cast into the second by ranking items on the model's predicted ratings for them. In our work, our model explicitly predicts ratings for items but we also follow this translation approach so as to create a set of evaluation metrics that are prevalent in the literature (see Section 2.1.6 for more details).

Explicit vs implicit ratings and the utility matrix

Regardless of how the model generates recommendations, it must learn from a dataset of user ratings. As such, it is important to categorise how these ratings are encoded.

At a high level the two types of ratings are *explicit ratings* - that is to say an explicit indication by the user of how much she liked an item - and *implicit ratings*, which are more subtle. Implicit ratings are potential indications of the user's enjoyment of or interest in an item inferred from signals relating to how she interacted with that item.

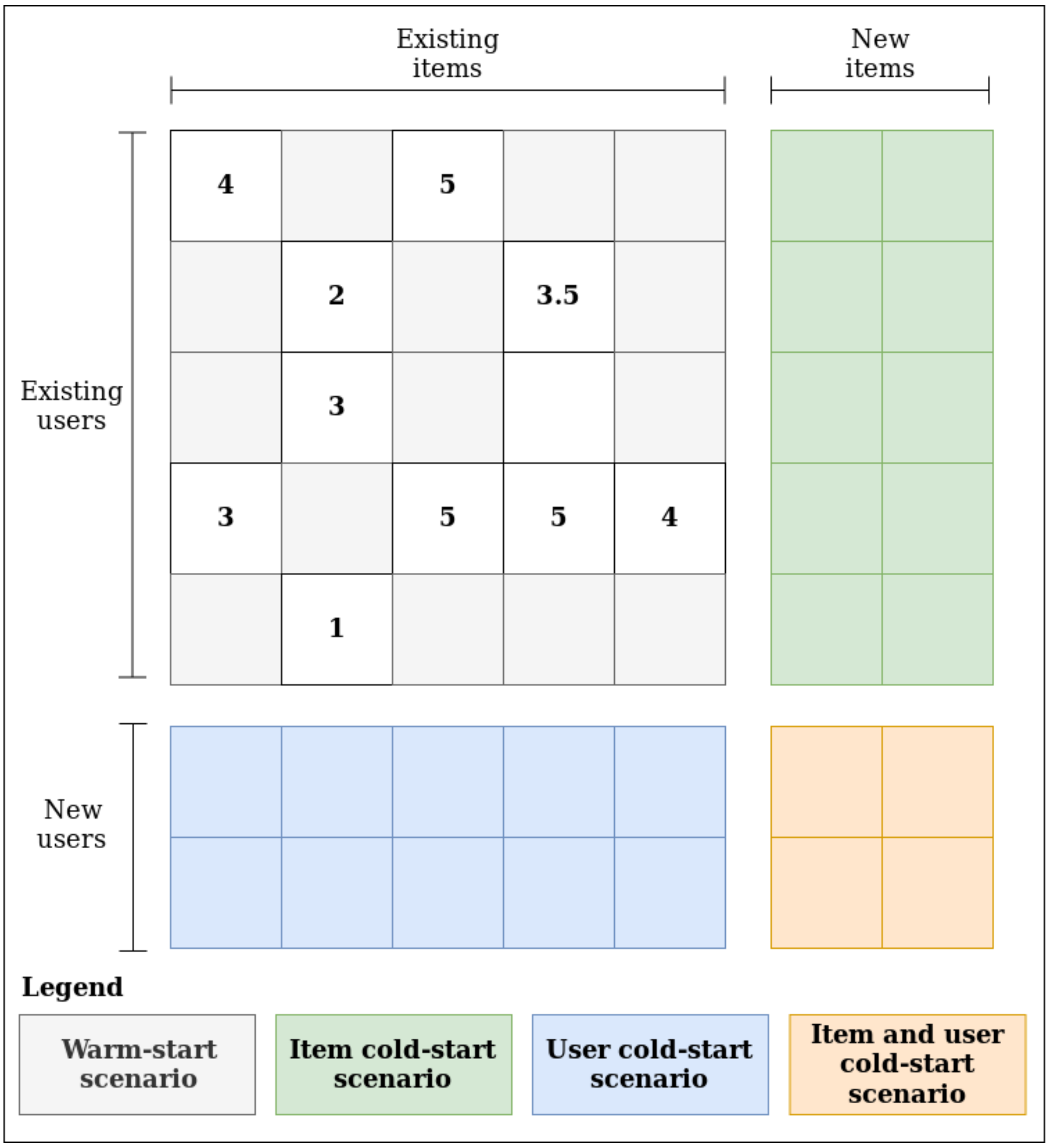


Figure 1: The matrix completion problem and the various data scenarios.

For example, watching a movie on a streaming service and stopping it half way might suggest that the user did not enjoy the item; whereas, watching the movie and then sharing it with friends might on the contrary suggest heightened enjoyment from consuming the item. Other signals from which implicit ratings can be inferred include preemptive ones such as saving an item to a list to watch later and the user's physiological response when consuming the item[19].

Explicit ratings are usually specified on an ordinal scale (for example, Netflix/Amazon item reviews) whereas implicit ratings are usually *unary* in nature - meaning that there is a signal which indicates a user's like of an item but not one which indicates a user's dislike. For example, buying an item intuitively suggests a user likes an item but not buying an item does not necessarily mean they did not like it. This lack of negative feedback can prove tricky when designing recommender systems and it

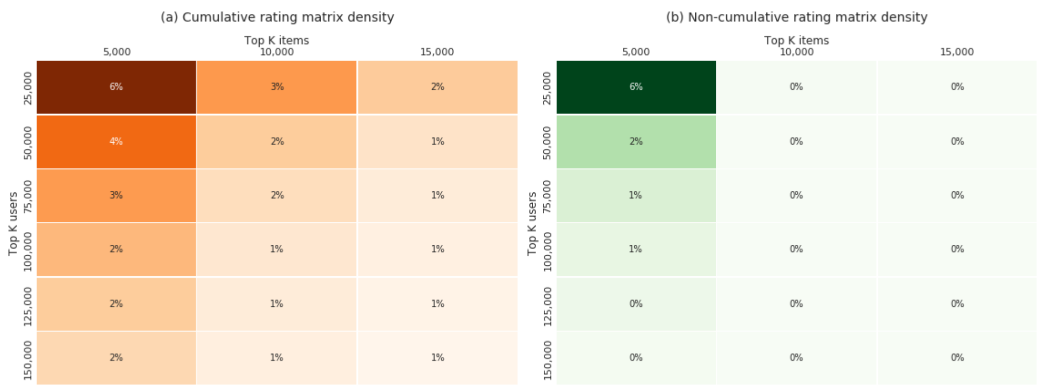


Figure 2: An illustration of rating matrix sparsity with our MovieLens AV dataset. More information on this dataset can be found in Section 4.1.

often pushes the designer to a *top-k recommendation* formulation of the problem.

Once the rating type has been decided it is common to present the rating data in a matrix. This matrix is often referred to as the *utility matrix* and it usually contains the user ratings of the items but not always[6]. In this report we will use the terms utility matrix and ratings matrix interchangeably as for our purpose the matrices we work with will always contain user ratings. Figure 1 shows examples of a *utility matrix* for explicit ratings.

Warm and cold start scenarios

Now that we know what form user ratings come in and how to predict them, it is important to consider the different scenarios in which we will operate. Figure 1 provides a graphical overview of the different *warm-start* and *cold-start* scenarios. Each is worthy of further explanation

1. The *warm-start scenario* refers to making recommendations for users and items already considered in the ratings matrix of the model. In other words, it is a case of making predictions of how *current* users' will rate the remaining items in the *current* catalogue. While potentially the simplest of the scenarios in the sense that it gives the modeller the most head start it is still not that simple as typically rating matrices are very *sparse* - meaning that they contain a high proportion of missing to non-missing entries - and, thus as the number of ratings for a user/item tends to zero, the warm-start scenario approaches the user cold-start and item cold-start scenario, respectively (see Figure 2 for an example of matrix sparsity). Under this scenario one often asks the question who is this user most like and what items do they like, as we will see in our section on collaborative filtering systems (which we discuss at length in Section 2.1.2)

[6]Formally, utility is defined as the amount of profit one takes from something and this choice of terminology is related to the commercial case for recommender systems outlined in Section1.1, where companies will track the profit they generate from effectively recommending products to customers.

2. The *item cold-start scenario* refers to recommending items that are new to the catalogue. In this scenario, one typically asks the question what item is this item most like which lends itself to systems that can recognise and recommend similar content - systems that work in this way are referred to as content-based systems (which we discuss in Section 2.1.3)

3. The *user cold-start scenario* refers to recommending existing catalogue items to new users. In this scenario we have no rating information from the user - so we cannot just recommend them items that users with similar review histories have enjoyed or items with similar content profiles to ones they have already reviewed. Instead we have to think about ways to work out what this new user may like from ancillary information (such as their demographics) and/or by asking for a subset of item ratings from them during customer onboarding

4. The *item and user cold-start scenario* is the most extreme in that we must recommend new items to new users. While this should be seen as an unlikely case (and one that we do not give much attention to in this work) it is not impossible. For example, companies in low volume and high unit price industries that sell unique items such as fine art and real estate may find recommender systems that can operate in this scenario useful as it is unlikely they have a large set of regular customers

Throughout the subsequent subsections we will outline various approaches for completing the utility matrix and their relative strengths and weaknesses under these scenarios.

2.1.2 Collaborative filtering

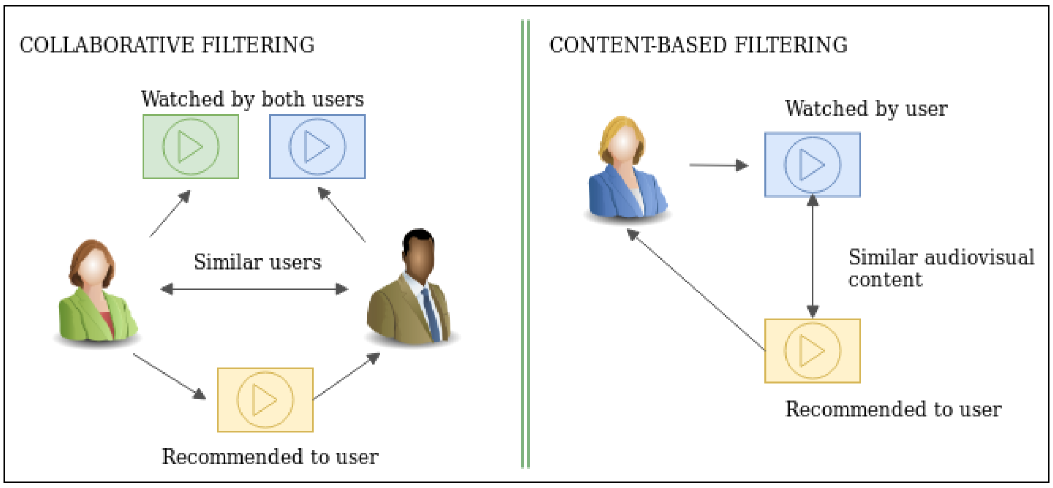


Figure 3: Conceptual overview of collaborative filtering versus content-based recommendation.

Broadly speaking there are two main types of recommender systems: *collaborative filtering* models and *content-based filtering* models. Figure 3 shows a graphical overview of these two types of systems. It is important to address the foundations and relevant literature of both of these areas as our proposal combines aspects from both of them into a *hybrid system* - for which we discuss the theory and design in Section 3. This subsection discusses collaborative filtering and the subsequent one addresses content-based filtering.

What are collaborative filtering recommender systems?

At a high-level, a collaborative filtering model looks at past user behaviour - for example movie ratings or product purchases - and uses this to find items that similar users have enjoyed to recommend to the target user[7]. The literature proposes two major categories of solutions to this problem

1. *Memory-based* or *neighbourhood* models: which look for neighbouring items or users (where neighbouring is defined by similarity metrics in the explicit or implicit rating space)
2. *Model-based* or *latent factor* models: which look to decompose the utility matrix into a lower-dimensional *latent* space where both users and items are described by latent factor vectors

While our work focuses on the second case, it is important to quickly address the former as we use them as baselines in our work. Memory-based models are typically implemented as k-nearest neighbour models (KNN) - a class of non-parametric/“lazy” machine learning algorithms[8] that are simple to implement.

Equations 1 and 2 outline the two most common formulations of k-nearest neighbour based rating prediction models. It is important to note that these equations capture the rating prediction problem from the perspective of using a set of similar users to triangulate the target user’s rating. It is equally viable to make the prediction using a set of similar items but we will not state the equations for brevity.

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)} \tag{1}$$

[7]It is worth noting that collaborative filtering can also be framed from an item-centric perspective, where the interaction history of the target item is compared to that of other items in order to predict unknown ratings. There are several reasons why a system might want to take an item-centric approach over a user-centric one. For example, if the number of items in the system is considerably less that the number of users then this can reduce model prediction time. Moreover, it can be argued that item collaborative filtering profiles are more stable than users profiles over time (as user taste is more liable to change) and thus item-centric approaches can result in more consistent and stable results[**item˙based˙cf**].

[8]Non-parametric/“lazy” algorithms are ones that only form an approximation to the target function at query time and based on the query data (i.e. they generate a local approximation of the target function). Thus, they offer a significant degree of flexibility and scalability, especially when dealing with sparse data but are prone to overfitting.

In these equations $\hat{r}_{ui}$ refers to the rating prediction for user u of item i; v refers to other users from the set of k other users; μ and σ are the k-sample mean and standard deviation, respectively. The similarity metric[9] $\text{sim}(u,v)$ is a measure of how similar user u is to users $v \in N_i^k(u)$ and in using this a weighting term is created that means the reviews from more similar users to the target users count for more towards the prediction.

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)} \tag{2}$$

We also implement an even simpler memory based model as a baseline in our work, which Equation 3 summarises. Here μ refers to the *global mean* - that is to say the mean rating for all currently known item and user pairs - and b_u and b_i refer to the user and item bias (with respect to the global mean), respectively

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i` \tag{3}$$

Matrix factorisation as a powerful solution to the matrix completion problem

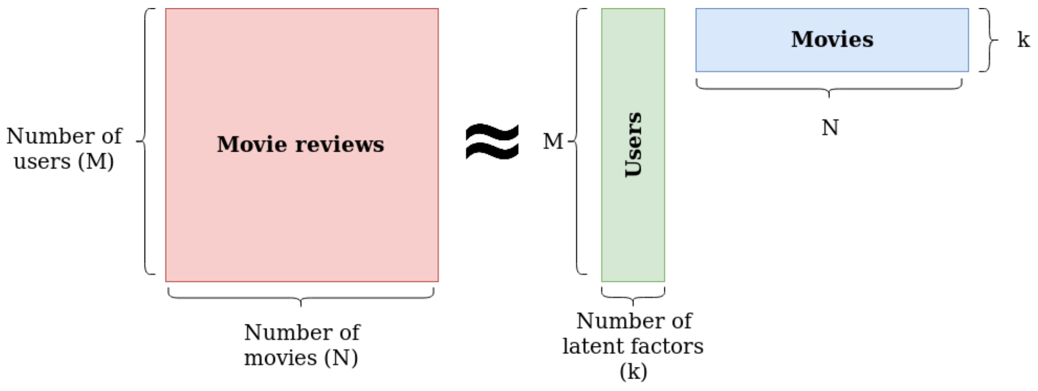


Figure 4: Conceptual overview of the matrix factorisation approach.

While the aforementioned *neighbourhood* models look for ways to predict missing ratings and make recommendations by navigating the existing user-item rating space, *latent factor* models reduce the dimensionality of the data to describe user and items in terms of information rich *latent factor vectors* and then recombine these

[9]A range of similarity metrics can be used to identify similar users and items. In practice, the most common for users and items in the explicit rating case is the *cosine similarity*:

$$\text{cosine\_sim}(u,v) = \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}}$$

In the implicit rating case metrics such as the *Jaccard similarity* are used.

vectors to calculate estimates for any missing ratings.

The overarching intuition for this matrix factorisation is that the observed data in a higher dimensional space (for us this is the rating space) can be fully (or almost fully) explained by factors in a lower dimensional hidden space - aka the latent space. Figure 4 shows the matrix factorisation algorithm graphically.

More formally, if we take the rating matrix to be R (of dimensions m users by n items), then matrix factorisation is about finding the best approximation to the matrix factors it is composed of - which we define as U for the user factor matrix (dimensions mxk) and V for the item factor matrix (dimensions nxk) where importantly k is the number of *latent factors* or concepts that we can treat as a hyperparameter

$$R \approx UV^T \tag{4}$$

The fact that the second dimension of the latent factor sub-matrix is the same for both the items and users means that this dimension can be interpreted in quite an intuitive way: for each user the extent of each element in the latent factor vector corresponds to how much the user likes those elements/concepts in an item. Similarly, for each item the extent of each element corresponds to how much that item possesses those elements/concepts.

It is for this reason that the extent of each element is often referred to as its *affinity*. However, it is worth noting that only sometimes do the elements/concepts in this latent dimension relate to actual features of the item and more often than not they are intangible.

So once we have reduced the known data to a joint latent space how do we use this to predict the missing user-item interactions? For each user-item pair this is as simple as taking the inner product of their latent factor vectors:

$$\hat{r}_{ui} = q_i^T p_u \tag{5}$$

In this equation $q_i \in R^f$ represents the latent vector for item i and $p_u \in R^f$ represents user u. As aforementioned, the value of each element corresponds to an affinity and therefore the inner product is intuitively predicting an overall score for how positive or negative the user's interaction with that item will be.

The rating prediction in Equation 5 can be improved further by introducing terms for user and item ratings bias $b_u + b_i$. If we are to believe that a user's review for an item is not just determined by their interactions with that particular item but also due to individual characteristics of the user and item indepedendent of each other (for example, the fact that user A may have a tendency to give higher ratings on average than user B; or the fact that a particular item is just not very good) then adding learnable bias terms makes a lot of sense.

The best way to add terms for both user and item bias is by formulating them as deviations from the *global average* of the rating dataset μ via $\hat{r}_{ui} = \mu + b_u + b_i$. These terms are then combined with the user-item interaction term learned through matrix factorisation to calculate a single estimate of the rating via the following equation

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{6}$$

Thus, the overarching challenge of building matrix factorisation based systems is working out how to most accurately compute the mapping of each item and user from the rating space to the latent factor space.

The long-standing solution to solve this problem is closely related to the *singular-value decomposition* (SVD)[10] - a technique that is used to reduce the dimensionality of data in a high-dimensional space to a lower-dimensional space whose dimensions are the most information rich *latent semantic* factors of the observed data.

While early matrix factorisation methods[11] had to grapple with missing data in the rating matrix in order to apply SVD (which is undefined when the matrix is incomplete), more modern matrix factorisation techniques reformulate SVD into an *incremental* algorithm[32, 58] that only works on the observed ratings - a feature that is very important for recommender systems where rating matrices are sparse as it greatly increases the scalability of the model in terms of shorter compute times and lower memory requirements.

This *incremental* approach to matrix factorisation is achieved by using *stochastic gradient descent* - a well known optimisation algorithm that updates a model's parameters over time in a way that minimises the *cost* function defined by the programmer. In the context of linear matrix factorisation the model parameters consist of the latent factor representation of the different items and users and the cost function is defined as

$$e_{ui} = \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2 \right) \tag{7}$$

In this equation r_{ui} refers to the true rating of item i by user u and $\hat{r}_{ui}$ is the model's prediction of this value. The first term in this equation is thus an error term - the *mean squared error* (MSE) to be precise (this metric is discussed in more detail in

[10]The *singular value decomposition* is a factorisation of a real or complex matrix. It is related to the *eigendecomposition* of a matrix - which refers to representing the matrix in terms of its eigenvalues and eigenvectors and usually only applies to *diagonal* matrices. SVD is a generalisation of this approach, allowing us to perform *eigendecomposition* on any mxn matrix.

[11]It is relevant to note that earlier matrix factorisation techniques relied on heuristics or imputation to fill in missing values in the rating matrix so regular SVD could be used[**pre˙incr˙svd**, 51]. This approach had two significant shortcomings: firstly, it greatly increased the number of ratings to $N_i.N_u$ which leads to significant scalability issues - both in terms of the additional time to compute the decomposition and the memory requirements to hold the whole rating matrix in memory at once. Secondly, imputing missing values based on various heuristics introduced significant bias to the model thus increasing the risk of the model learning inaccurate approximations of the true target function.

Section 2.1.6). The second term in this equation is a *regularisation* term - where λ is a hyperparameter to be optimised by cross-validation and $||q_i||$ and $||p_u||$ refer to the *Frobenius* norm of the item and user latent vectors, respectively.

This regularisation term has a very important role in reducing overfitting in the model which is essential when working with sparse datasets. In exactly the same way regularisation is used in classification and regression models, it introduces a term to the cost function that penalises large coefficient values in q_i and p_u - as these large values usually correspond to excessive variance in the target function approximation that has resulted from overfitting to noise in the training data. A full derivation and assessment of regularisation as an approach to control overfitting is beyond the scope of this section but can be found in [35, 49, 53].

With the cost function defined, stochastic gradient descent thus updates the learnable model parameters via the following set of update equations

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \tag{8}$$
$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \tag{9}$$
$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \tag{10}$$
$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \tag{11}$$

In this equation γ is the "learning" rate which is a programmable hyperparameter which is optimised through cross-validation. Thus, by running the matrix decomposition and reconstruction steps over several iterations - and updating the learnable parameters in the user and item latent factor matrices at the end of each iteration as per the rules in equations 8-11 - we can converge upon a good approximation of the missing values. This is the crux of the matrix factorisation approach to recommender systems.

Finally, we note that as matrix factorisation has been at the centre of state of the art recommender systems for over a decade the literature includes many variants that are not discussed in this short overview[11, 40].

Going beyond linearity - deep neural network solutions to the matrix factorisation problem

While the linear matrix factorisation-based methods described in the previous section have reigned supreme at warm-start scenario problems for the last decade[11, 40], non-linear techniques have recently emerged in the literature that use neural networks to achieve superior results. Our approach is largely based on these recent architectures so we take some time in this section to point out their advantages.

Firstly, to describe why linear-based techniques are suboptimal we use the example illustrated in Figure 5. This figure shows how the inner product function at the heart

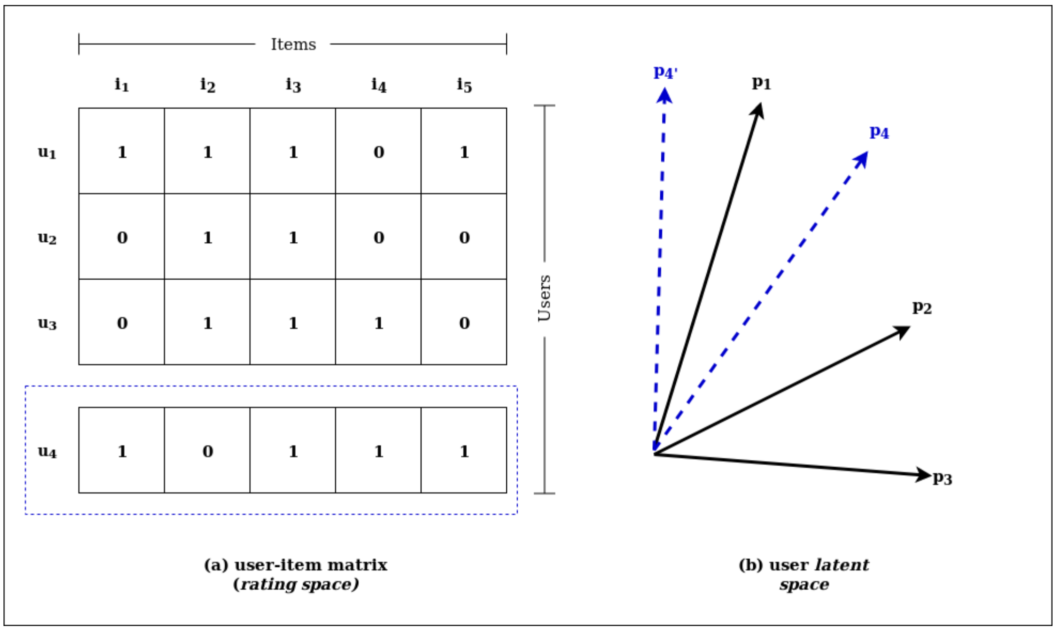


Figure 5: The problem with linear matrix factorisation. Despite u_4 being more similar to u_3 than u_2 in *rating space* there is nowhere it can be placed in *latent space* to satisfy this constraint while also satisfying the constraint that u_4 is most similar to u_1. NB: This figure comes with explicit thanks to Xiangnan He et al [23].

of linear matrix factorisation algorithms (see Equation 5) can limit their expressiveness. In the example of Figure 5 we are initially presented with the review vectors of three users (here the ratings are implicit and hence binary): a) shows the ratings for these users in *rating space* and b) shows the ratings for these users in *latent space*.

The challenge is where to place a new user u_4. Using the Jaccard similarity measure[12] we observe the following inequalities for existing users $(\text{sim}(u_2, u_3) = 0.66) > (\text{sim}(u_1, u_2) = 0.5) > (\text{sim}(u_1, u_3) = 0.4)$. However, for the new user $(\text{sim}(u_4, u_1) = 0.6) > (\text{sim}(u_4, u_3) = 0.4) > (\text{sim}(u_4, u_2) = 0.2)$ which creates a problem. While it is possible to satisfy the first rating space inequality for u_4 in the latent space also - by placing p_4 closest to p_1 (see the dotted lines in (b)) - in doing so we violate the second rating space inequality for u_4: that is to say, wherever we place p_4 it will be closer to p_2 which is not an accurate representation of the ground truth in the rating space. Such gaps in expressiveness result in prediction errors in linear matrix factorisation models.

So how can we introduce this non-linearity? By restating Equation 5 in the more general form of Equation 14, where f is the inner product (in other words the

[12]The Jaccard similarity measure is a common similarity measure for implicit rating systems and is defined as

$$\text{sim}(i, j) = \frac{|R_i| \cap |R_j|}{|R_i| \cup |R_j|}$$

If we were considering an explicit rating space we could use the cosine similarity defined previously.

element-wise product of the vectors followed by their sum) of the user vector p_u and the item vector q_i, we can tee up the conceptual leap.

$$\hat{r}_{ui} = f(u, i|p_u, q_i) = p_u^T q_i \tag{12}$$

While the operations that constitute f - in other words the model's approximation of the *target function* - can be programmed *a priori* it is not necessary to approach the problem with such inductive bias. Instead we can pose the problem as a more flexible learning task in which a deep neural network is used to provide a larger hypothesis space in which to search for a closer approximation to the target function[13]. Then during training of the network - using *backpropagation* of errors through the system to update network parameters accordingly - we can arrive at a higher-order, more-complex approximation to the *target function* which more accurately models the interaction between user and item vectors and thus achieves better recommendation results.

Thus, we can rewrite Equation 14 as

$$\hat{r}_{ui} = f(P^T v_u^U, Q^T v_i^I|P, Q, \Theta) = \phi_{out}(\phi_X(...\phi_2(\phi_1(P^T V_u^U, Q^T v_i^I))...)) \tag{13}$$

In this equation, v_u^U refers to an input feature vector for the user u and v_i^I refers to an input feature vector for the item i. $P \in \mathbb{R}^{m \times k}$ and $Q \in \mathbb{R}^{n \times k}$ refer to the user and item latent factor matrices, respectively and Θ refers to the model parameters. Finally, ϕ_{out} and $(\phi_x$ represent the *mapping* function of the output layer and the x-th layer of the neural network (of which there are X hidden layers in total).

Two different formulations of collaborative filtering as a deep learning problem are proposed by the pioneers of this technique[23] and these are displayed in Figure 6.

The first of these - termed *general matrix factorisation* - maintains the element-wise vector product operation of linear matrix-factorisation but introduces non-linearity by way of a non-linear sigmoid activation function at the end of the network (in place of the identity operation in linear matrix factorisation) which facilitates the learning of non-linear function approximations in the hidden layer h of the network. This can be summarised by Equation 14, where a_{out} is the sigmoid function and h is the only hidden layer of the network

$$\hat{r}_{ui} = a_{\text{out}}(h^T(p_u \odot q_i)) \tag{14}$$

The second technique concatenates the user and item latent factors and then passes them through a *multilayer perceptron*, which adds an arbitrary number of *fully-connected layers* with non-linear activation functions. Equation 13 is the most concise formalisation of this structure. This version of the model achieves state of the art results and thus we use a variant of it in our work.

[13]This use case of neural networks as *global function approximators* is well proven[42] and is utilised in many domains.

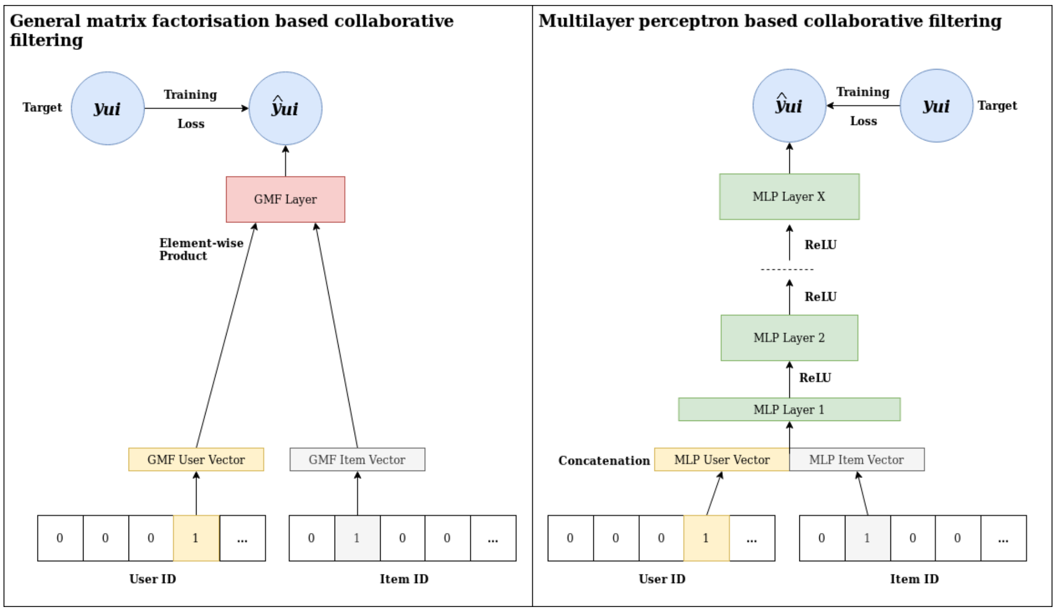


Figure 6: Two deep learning approaches to collaborative filtering: general matrix factorisation and MLP-based collaborative filtering.

2.1.3 Content-based filtering

With the state of the art in collaborative filtering now defined, we should next shift our focus to providing a brief overview of content-based filtering techniques as the overarching goal of this research is to enhance CF models with content-based capabilities.

What are content-based recommender systems?

Content-based recommender systems rely on the actual content of an item and the target user's reviews of it to make predictions of how that user will review other items (for which the content is available). As a result they are unlike collaborative filtering models in the sense that they do not require the reviews of other users (or items) in order to generate rating predictions.

The content descriptors of an item can come in many different forms and they can usually be described by one of two categories: either they are human generated at varying levels of details, such as user tags and labels about the properties of a product. Or, they are objective properties of the item itself, such as its visual aesthetic or its textual content (for example the trailer of a movie and the script of a movie). A collection of content descriptors that describe an item are usually referred to as an *item profile* and deciding on a meaningful *item profile* is a feature selection/weighting problem.

Central to the content-based problem is working out how individual users will react to the content of an item and thus any content-based approach must construct what is often referred to as a *user profile*. This is usually constructed by simply combining the ratings the user has given to items with the content of these items. This profile is then an indication of the type of content the user likes and can be used to recommend new items to them with similar content profiles.

Nearest neighbour based approach to content-based recommendations

A common approach to content-based recommendation is to perform a KNN-based prediction on a per user basis. If we define the items a user has reviewed as $i \in I_{u,train}$ and the items we can recommend to the user as $j \in I_{u,test}$, then the predictive model can be defined as

$$\hat{r}_{ui} = \frac{\sum_{j \in I_{u,train}}^{j \leq k} \text{sim}(i,j) \cdot r_{ui}}{\sum_{j \in I_{u,train}}^{j \leq k} \text{sim}(i,j)} \tag{15}$$

Where, very importantly, the similarity metric $\text{sim}(i, j)$ is calculated between the different content descriptors of the item and k is the number of nearest neighbours considered in the calculation.

We implement such a model as one of our baselines for our experiments (see Section 4.3) using handcoded genre features. It is nonparametric and thus enjoys the benefits that come with this.

For example, even if the target function is very complex it can still achieve good approximations for each test instance as it generates a *local approximation* at query time for each instance rather than having to commit to a *global approximation*. This may result in more accurate or more personalised recommendations. However, it also has the downsides of nonparametric functions - such as requiring a lot of work at *query time* to generate predictions and also not being robust to outliers.

Embedding based regression approach to content-based recommendations

The other main approach to content-based recommendation is to perform a regression on a per user basis. In its simplest form a regression model could be built per user, but the scalability of this system would be very poor and thus one must be more ingenious in designing the system.

A recent solution to this problem[56, 52] - and the one that we implement in our audiovisual content-based recommender system - is to create a model that has a *user embedding* for each user with the same dimensions as the content-based *item profile* (or *item content embedding*).

The model is provided with the *item profile* and the user ID as input (with which it

Table 1: Comparison of collaborative filtering to content-based models

	Collaborative filtering	**Content-based**
Warm start results	Achieves state of the art accuracy results if user and item review data is available during training	Significantly worse results to CF models in warm start scenario
Cold start results	Unable to make accurate predictions in both item and user cold start scenario	No item cold start scenario issues if user has provided some item ratings. However, still prone to user cold start issues.
Quality of recommondations	Can find surprising similarities between items (for example, items from different genres) that please the use - e.g. by creating a feeling of serendipity. However, it is also prone to *popularity bias* where more popular items dominate the recommendations.	Prone to recommending the most obvious item as it recommends those with the most similar content to what the user has already liked. This may or may not be desirable depending on the context.
Interpretability	Harder to build into the system as *latent factors* often do not have tangible meaning.	Easier to provide especially if *human-generated* content descriptors are used (e.g. recommended because it contains cat videos)

looks up the user's embedding) and makes a prediction for that specific user in a way that is entirely analogous to the matrix factorisation approaches for collaborative filtering discussed in the previous section. During training the user embedding is modified in the same way as in these approaches, with the main exception between the approaches being the fact that the content profile of the items remain fixed.

2.1.4 Pros and cons of collaborative filtering and content based models

Before moving on to how the best of collaborative filtering models can be combined with the best of content-based models, it's worth quickly summarising the relative advantages and disavantages of both methods. Table 1 provides this.

2.1.5 Hybrid recommender system techniques

As our proposed approach creates a *hybrid recommender system* we quickly define the term here. Hybrid systems are defined as those that combine two or more recommender system models with the intention of enhancing their capabilities and it can be done in one of several ways.

For example, ensembles of different models can be created with the resultant prediction decided by a *query by committee* style set up. Or, use of the component systems can be phased - for example, using one of the models in one scenario and the other in another scenario. Finally, the inputs to the models can be combined and the model designed in such a way so as to reach a single prediction.

The reason why one would look to create a hybrid recommender system is to alleviate shortcomings in the individual models. The most common reason for doing this is to treat cold-start issues, both user and item-based. We save further discussion on how we designed our hybrid recommender system until Section 3.

2.1.6 Evaluation methodology

As the suite of evaluation metrics used to evaluate recommender systems is vast in scope, we summarise them in Table 2. We also provide a “toy example” of the rank metric calculations in Appendix E.

The following subsections walk through the definitions of these metrics in detail.

The need for a broad suite of evaluation metrics

In Section 1 we outlined both the commercial and personal case for recommender systems. In this section we look at how we can quantitatively measure their performance across various different performance dimensions. Before diving into the detail, it is instructive to state upfront the importance of using a broad suite of evaluation metrics to assess the performance of a recommender system rather than just analysing the prediction accuracy (e.g. via a loss term).

The reason for this is twofold: firstly, the most relevant evaluation metric will usually depend on the context in which the recommender system is being used. So for example, a recommender system may only have the chance to recommend one item to a user or it may get to recommend a list of items, and thus the accuracy measures in the latter case will have to analyse the contents of a recommendation list whereas the former is just concerned with the top rated item.

A second important reason for employing a suite of metrics is that the desired features in a recommender system are not necessarily complementary to each other and thus tradeoffs must be made. For example, while recommending the items with the highest predicted review to a user may result in the greatest user satisfaction it may

Category	**Metric**	**Description**
Accuracy	MSE	Is a measure of loss for regression tasks that is typically used as the basis of a cost function in recommender systems.
	RMSE / NRMSE	Is a measure of the deviation of the recommender system's predicted ratings versus the ground truth value of the ratings. N refers to *normalised* - a transformation that ensures its within the range of the ground-truth min and max rating value.
	MAE / NMAE	Interpreted in a similar way to RMSE. Does not sum the square of the errors in its computation so is more robust to outlier values versus RMSE.
Rank	MRR	Is a measure that rewards the system more for placing the first relevant item higher in the list of recommended items. It uses a binary relevance term.
	MAP	Is a measure that rewards the system more for placing many relevant items in the list of recommended items. It also uses a binary relevance term.
	NDCG	Is a measure that rewards the system more for placing the many relevant items higher in the list of recommended items and differs from MAP in that it uses the ground true rating values as its relevance term.
ROC	Precision	Is a measure of the ability of the recommender system to only recommend items that are relevant in the ground truth data
	Recall	Is a measure if the ability of the recommender system to recommend as many of the items that are relevant in the ground truth data
	F1	Is the harmonic mean of precision and recall and represents the system's overall ability to perform both well at both of these metrics
Other	Catalogue coverage	Refers to the proportion of a company's overall item list that a recommender system includes in its recommendations
	Diversity	Refers to the diversity in the list of items recommended to a particular user
	Personalisation	Refers to how unique a particular user's list of recommended items is versus all other users

Table 2: Summary of the different evaluation metrics used to assess recommender system performance.

also mean that a large proportion of the company's catalogue is not being recommended to anyone. This may cause issues for the company as it might mean that a lot of their stock is never sold. Similarly, if all the items in a recommendation list are similar what happens if the model is wrong and the user actually dislikes all items with that content profile? It is for these reasons amongst others that we provide a range of evaluation metrics in our approach and theoretical write up.

Online versus offline evaluation

It is also worth briefly noting the definitions of the concepts of *online* versus *offline* evaluation in the context of recommender systems. *Online* evaluation refers to evaluating the effectiveness of recommendation algorithms through real interactions with users. In the past it has been typically performed by *ab-testing* different algorithms with different sub-groups of users then choosing the algorithm that performed best on the sub-groups to be used in future.

However, more recently reinforcement learning approaches based on *multi-armed* and *contextual* bandits have been adopted which adapt the recommendation algorithm during the evaluation session based on the user's feedback and in doing so minimise the *regret* incurred in the evaluation process[14].

By contrast *offline* evaluation refers to assessing the performance of a system using historical data and it is much more common in the literature as a) it is easier to organise and b) it makes it easier to test recommender system algorithms on a variety of data sets. Therefore, while *online* evaluation is a very exciting area and one that we considered pursuing for this work **we limit the evaluation of our system to the offline context** given the main contributions of this work are to assess the utility of low-level video content on the accuracy of recommendations and we propose the findings from this will have equal relevance in the online and offline context.

Raw recommender system accuracy

Once the data has been segmented the simplest and most obvious way to assess a recommender system's quality is by computing the error in its predictions versus the test set. As discussed in Section 2.1.1, the explicit and implicit rating prediction problem is usually cast as a regression problem and therefore typical regression error metrics can be used to evaluate recommender systems (and indeed to train them as Equation 7 shows).

The most common regression metric is the mean squared error (MSE) defined as

[14]Regret refers to the loss in user satisfaction (and therefore profit) that results from following the sub-optimal recommendation policy during algorithm evaluation. It is actually a huge source of lost profits to a company and thus *bandit* algorithms have been a significant breakthrough in this area

$$\text{MSE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2. \tag{16}$$

Where $|\hat{R}|$ is the number of ratings in the test set, $\hat{r}_{ui}$ is the predicted rating and r_{ui} is the ground-truth/actual rating. The shortcoming of using the MSE is that it is not in units of rating and thus it is less easy to tell quickly tell by how much the algorithm miscalculates the ratings. To improve upon this, the *root mean squared error* (RMSE) and the mean absolute error (MAE) are common variations

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}. \tag{17}$$

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}| \tag{18}$$

While these terms make it easier to interpret the magnitude of the error they can be improved further by *normalising* them, which in this context refers to modifying their values to the domain between the maximum and minimum rating value. This further improves interpretability as unbounded regression errors can easily be in a range far greater than actual range of the rating scale used

$$\text{NRMSE} = \frac{\text{RMSE}}{r_{max} - r_{min}} \tag{19}$$

$$\text{NMAE} = \frac{\text{MAE}}{r_{max} - r_{min}} \tag{20}$$

Ranking accuracy evaluation through utility

While the accuracy metrics from the previous section evaluate the system's accuracy over all data points, ranking metrics typically evaluate the system's effectiveness by just looking at the *top-k* items recommended. Given that only a subset of the items in a catalogue are recommended to a user in practice, it is often argued that analysing a system in this way yields results that are more relevant to how the system will be perceived in a real-world setting.

Generally speaking there are two forms of ranking metrics: those that compare the ground-truth *ranking* to the recommender system's *ranking* and those that use the ground-truth rating to analyse the system's *ranking*. The latter case are referred to as *utility* based methods and **we focus our analysis on these utility metrics**.

The rationale for using utility based measures over general ranking measures (and indeed over solely accuracy based metrics) is that we should analyse our system on the basis of providing recommendations that a user might actually find useful and therefore ranking items that the user actually likes highly is more important than simply getting the ranking order correct. This is particularly important in the typical case where recommended item lists are significantly shorter than the number of items in the catalogue: utility metrics allow us to make the items at the top of the list count for more than items further down the list.

So how do utility based metrics work in practice? The overarching idea is to have two utility terms in the analysis metrics: one that measures the ground-truth utility of the item to the user (where items with higher ground-truth ratings obviously having higher utility to the user) and one that measures the position of the item in the recommendation list (following the logic that items higher in the list have higher utility to the user as they are more likely to get noticed and thus consumed).

In order to quantify utility in this way the concepts of recommended items and relevant items and are important to understand. *Recommended items* are simpler to understand - these are the items that the system actually recommends to the user. The logic for what items get recommended is usually based on a threshold value where any item that has a rating greater than this threshold makes it into the recommended list.

Relevance in the context of items recommended is defined as those items that are actually liked by the user. It is thus based off of the *ground truth* data and again a threshold is used to decide which items are relevant but there is further nuance in the sense that relevance can can be binary or continuous/ordinal valued depending on the evaluation metric used.

With these preparatory definitions out of the way we now consider the three main ranking metrics we use in our system.

Mean reciprocal rank / average reciprocal hit-rate

The first metric we use is the *mean reciprocal rank* (MRR) (also known as the average reciprocal hit-rate in the literature [29]). The logic behind this metric is to rank the system based on the position of the *first relevant* item in the list and it is defined as follows

$$\mathrm{MRR} = \frac{1}{\mathrm{m}} \sum_{u=1}^{m} \frac{1}{\mathrm{rank}_{uj}} \tag{21}$$

In this equation m is the number of users, item j is the first relevant item, and rank_{uj} is the rank position of the first relevant item. Importantly, we use a binary definition of relevance in our version of this metric and thus the numerator is 1. As rank is the denominator, the MRR reduces as the first relevant item is found further down the recommended list. The intuition behind this metric is thus how good the system is at pushing relevant items to the user.

Mean average precision

While MRR rank is a good first start towards a utility based evaluation system it can be improved upon by taking into account not just the first relevant item in the recommended items set but *all relevant items* in the set. To do this we use a metric called the *mean average precision* (MAP) which is based off of the definitions of precision and recall in the previous section. Mean average precision is calculated by looking at the *average precision* (AP) of the system for each user and this latter metric is defined as follows

$$\mathrm{AP} = \frac{1}{\mathrm{N}} \sum_{k=1}^{N} P@\mathrm{k} \cdot \mathrm{rel}(k) \tag{22}$$

Here N is the number of recommended items to consider (where $N <= N'$ and N' is the total number of items recommended for each user and the logic for this limit being that a system will only recommend a subset of the items it calculates predictions for) and k is each subset of these items from just the first item to the N^{th} item. The term $P@\mathrm{k}$ refers to the precision at k defined exactly as in Equation 26 and $\mathrm{rel}(k)$ is the relevance of item k which is binary in this metric.

Mean average precision is then defined as the mean of the *average precision* across all users m

$$\mathrm{MAP} = \frac{1}{\mathrm{m}} \sum_{u=1}^{m} \mathrm{AP}(u) \tag{23}$$

The average precision part of the MAP metric is what allows it to improve upon the MRR metric. The intuition is as follows: firstly, the MAP rewards the system for all relevant recommendations made through the binary relevance term $\mathrm{rel}(k)$. Moreover, through the $P(k)$ term it rewards the system for front-loading the relevant predictions - as $P(k)$ is greatest when all recommendations up to and including k are correct. Thus, MAP provides us with an overall utility score for all items on the recommended list.

Normalised discounted cumulative gain

Can we go just one step further and reward the system for all relevant items in the recommended list *taking into account the ground-truth utility of each item?*. Indeed we can, the metric to use for this is the *discounted cumulative gain* (DCG) which rewards the system by using a *non-binary relevance* and is defined as

$$\mathrm{DCG} = \frac{1}{\mathrm{m}} \sum_{u=1}^{m} \sum_{k=1}^{N} \frac{g_{uk}}{\log_2(v_k + 1)} \tag{24}$$

The limits of this sum are firstly over all users u in m and secondly over all items $j \in I_u$ up to position N in the ranked list. The term g_{uj} refers to the ground-truth utility of the items defined as $g_{uj} = 2^{\mathrm{rel}_{uj}} - 1$ where rel_{uj} is the aforementioned *non-binary relevance* value of the item for which we use the ground truth ratings.

The term $\frac{1}{\log_2(v_j+1)}$ models the utility from the item's position in the system's generated list - where v_j is the rank of item j in the list and thus the contribution of an item to the DCG reduces when it becomes further down the list. Therefore, when combined we can see that these terms reward the system in a similar way to the terms in MAP except that the extent to which a user likes an item in the list (using the ground-truth rating as a proxy) is also incorporated.

A useful extension to the DCG score is to *normalise* it so that values of the metric can be compared between different data cuts and data sets. To achieve this the raw DCG score is divided by the *ideal discounted cumulative gain* (IDCG) of the system which is computed via equation 24 by using the ground-truth ranking in place of the system's ranking

$$NDCG = \frac{DCG}{IDCG} \tag{25}$$

Ranking accuracy evaluation through receiver operating characteristic curves

An alternative way to evaluate ranking in recommender systems is by using *precision* and *recall* concepts to plot *receiver operator characteristic* curves that illustrate the trade-offs to be made when deciding on the size of the recommendation list. The concepts of precision and recall in recommender systems are similar to the same concepts in the context of classification, except that the recommender systems proposes a list of items the user might like rather than a single class label. Given this list-based output and the fact that a recommender system will probably not recommend all items that it calculates a prediction for it is common to see precision and recall "@*k*" which refers to the precision and recall rate for the first *k* items in the list of length N'.

Formally, *precision@k* in our context is defined as

$$\text{Precision@k} = 100 \cdot \frac{|\text{Recommended}(k) \cap \text{Relevant}(k)|}{|\text{Recommended}(k)|} \tag{26}$$

Similarly, *recall@k* is defined as

$$\text{Recall@k} = 100 \cdot \frac{|\text{Recommended}(k) \cap \text{Relevant}(k)|}{|\text{Relevant}|} \tag{27}$$

Intuitively, we interpret these metrics as follows: the precision of a recommender system algorithm rewards it for how well it recommends *true positives* (that is to say items the user likes - *relevant* items) without also recommending *false positives* (items the user does not like). On the other hand, recall rewards the system for how well it recovers all *true positives* and is unconcerned by *false positives*.

It should be clear from the definitions that a trade-off must be made between precision and recall and a very effective way to visualise this trade-off (and thus compare different recommender system algorithms) is by plotting a *receiver operating characteristic curve*. This consists of plotting the *true positive rate* versus the *false positive rate* for different values of k[15]. The line/curve that these points create thus forms the perimeter of an area - creatively called the *area under the curve* (AUC) - whose size is proportional to how effective the algorithm is at increasing the true positive rate while suppressing the false positive rate. We save further discussion on what exactly we can infer from these plots until Section 4.

We also note that while it is not as informative as viewing the plots, a version of the F_1 score exists for recommender system - which is the harmonic mean of precision and recall - and this is useful for rapidly analysing the trade-off. It is defined (similar to as in other domains) as

$$F_1\text{@k} = \frac{2 \cdot \text{Precision}(k) \cdot \text{Recall}(k)}{\text{Precision}(k) + \text{Recall}(k)} \tag{28}$$

Finally, it is important to address the question of why this alternative view of a system's ranking performance is useful. Ultimately, the level of true or false positive rates is an “acceptance criteria” that the company operating the recommender system must decide upon. If the recall is too low then the user might miss items that they actually like, causing the company to miss out on profits; by contrast, if the precision is too low the system might recommend too many items that the user does not like thus affecting the user experience and potentially causing churn from the platform.

To take a real world example - imagine how frustrating it would be as a user to constantly receive spam from a company recommending almost every item in their catalogue. Precision and recall allow the company to quantify and control experiences like these and decide on a level of both which is appropriate for their line of business.

Coverage, diversity and personalisation - other ways of looking at recommender system effectiveness

Beyond the conventional accuracy and rank based metrics there are several additional metrics that are worthy of note as they help to highlight additional features of a recommender system that may be as important as accuracy depending on the target domain[6]. Given our primary context is movie recommendation we opt to

[15] *True positive rate* is exactly equivalent to the definition for recall in equation 27 and the *false positive rate* is defined as (where I_u is all possible items)

$$\text{FPR}(k) = 100 \cdot \frac{|Recommended(k) - Relevant|}{|I_u - Relevant|}$$

also assess our system along the dimensions of *catalogue coverage*, *diversity* and *personalisation*.

- **Catalogue coverage**: refers to a simple assessment of what proportion of a company's catalogue the recommender system is currently recommending. It looks across all recommendations to all users and compares this to all unique items in the catalogue. In this equation m is the number of users and I is the set of all items

$$\mathrm{CC} = \frac{|\bigcup_{u=1}^{m} \text{topk-recommendations}(u)|}{|I|} \tag{29}$$

- **Diversity**: refers to the amount of variety within a single user's list measured using a similarity metric on the features of the items in the list[16]. In other words, it is a measure of the *intralist* similarity and the commonly held view is that lists should be diverse to a) reduce the user's fatigue at always seeing the same thing, and b) hedge the risk that if the user dislikes one item in the list then they are likely to dislike all its items if they are similar. In this equation $F(i_{u,x})$ refers to the features of items k and j, respectively

$$\mathit{Diversity} = \frac{1}{2m} \sum_{u=1}^{m} \sum_{i_{u,k} \in I_u} \sum_{i_{u,j} \in I_u}^{i_{u,k} \neq i_{u,j}} (1 - \text{cosine\_sim}(F(i_{u,k}), F(i_{u,j}))) \tag{30}$$

- **Personalisation**: where diversity refers to the amount of *intralist* variety generated by the algorithm, *personalisation* measures the amount of interlist diversity. In other words, it is a measure of how well the algorithm has learned the specific tastes of a user and this can be seen as a highly prized asset as it may help build a lot of brand loyalty with the customer (e.g., "XX really know what I like"). Importantly, additional feature representations do not need to be used for items in this metric - rather the similarity measure is calculated directly on the contents of the recommendation list, noted as $I(u_x)$ for each user i and j

$$\mathit{Personalisation} = \frac{1}{2} \sum_{u_i \in U} \sum_{u_j \in U}^{u_i \neq u_j} (1 - \text{cosine\_sim}(I(u_i), I(u_j))) \tag{31}$$

Scalability

Finally, no evaluation of recommender systems would be complete without also considering the scalability of the systems built. Recommender systems are a well-know "big data" task. To give an example, our target dataset (MovieLens 20M citemovielens) consists of 27,000 movies and 138,000 users. While only 20 million explicit

[16]For simplicity and efficiency, we use one-hot encoded features for each item which represent the genres of the movie

ratings are provided for this dataset the overall number of cells in the rating matrix is approximately 3.7 billion, and thus as many recommender system methods rely on the resconstruction of the full rating matrix (see Section 2.1.2), we must always be mindful of scalability when building recommender systems.

We consider the following scalability dimensions

- **Training time**: this refers to the time taken to train the model from a random initialisation state to a level of accuracy that is deemed acceptable to the operator of the system and we measure this in wall clock time
- **Prediction time**: this refers to the time taken to generate a list of recommendations for a set of users and again this is measured in wall clock time
- **Memory requirements**: this refers to the memory/RAM requirements of the system and we measure it in gigabytes (GB)

Interpretability

Finally, it is worth quickly noting another property that is becoming increasingly desirable in recommender systems as algorithms evolve - interpretability.

Despite the accuracy of cutting edge matrix factorisation methods, they typically fail at producing interpretable recommendations as the *latent factor* spaces they trawl through to generate recommendations often have little tangible meaning. Therefore, while users may be more pleased overall with the list of recommendations these algorithms have produced for them, the *human touch* of explaining why the recommender thinks they might like them is lost.

We do not formally assess our systems along interpretability lines in this project (as this is hard in an offline setting without user involvement) but it is worth noting that increasingly more research effort in the field is going into defining and designing explainable systems[59].

Included in these works is proposals for offline *interpretability* metrics such as *mean explainable precision*, which assesses the model's ability to prioritise only recommending items it can explain and recommending, and *model fidelity*, which rewards the model for having more explainable items in its recommendation lists.

2.2 Audio-visual content representation and analysis

2.2.1 Extracting deep visual features from audiovisual content

A visual stream of a video consists of a sequence of image frames shot at a certain rate of frames per second. As a result, the first step to extracting visual features from a video is to extract features from individual image frames.

For a long time the field of computer vision looked to handcrafted feature extraction techniques such as SIFT, SURF and HOG to generate these features but the mantle of "go to" method has since been seized by *deep convolutional neural networks*(CNNs) which can produce information rich feature descriptors for images they have not seen before (so long as they have been pretrained on a sufficiently large training image set)[41].

A detailed explanation of CNNs is out of scope for this project but a brief explanation of how they represent images is important. As images - which are represented in raw pixel form in the input - are passed through the network, the network reduces their dimensionality and performs a type of feature selection on them (which is invariant to the location of features in the images).

The complexity of these feature representations is greater the deeper into the architecture they occur. So for example, in early layers the features may be simple shapes such as lines and edges but further into the network they may represent specific features of images (like human facial features).

The final layers of the CNN architecture are then used in a *discriminatory* way to make predictions (often on what the raw image contains) based off of these inner feature representations. Networks such as *AlexNet*[25], *VGG16*[61] and *ResNet*[16] are well-known for their performance on such tasks (for example on classifying images in the vast image catalogue that is ImageNet[28]).

However, the great power of these networks - and how we use them - is that the deep feature representations they generate need not be used for the same discriminatory task on which they are trained. In fact, they can easily be extracted from the network and used as input into a completely different model.

Moreover, there is no need that the images run through the CNN when using it in this *feature extraction* use case be from the same domain as the images used to train the model. As the CNN has built up an internal understanding of the contents of images, the features extracted from it will still be information-rich and meaningful even if it has not seen the images from which we wish to extract features before.

Thus, given the benefits of this method, as well as its prevalence throughout the literature in a broad range of domains[41] (including in the field of recommender systems[24]), we choose to implement it as our visual feature extraction technique. For more details on exactly how we implement it in our feature extraction pipeline please see Section 4.2 for more details.

2.2.2 Extracting deep audio features from audiovisual content

Audio features can be extracted from the audio stream of videos in a way that is largely analogous to the visual feature extraction discussed in the previous section. What ImageNet is to visual feature representation learning, AudioSet[14] is to audio feature representation learning - a vast, labelled audio clip catalogue which researchers can use as a training ground for their audio classification models.

Taking a conceptually similar approach to how *AlexNet* and *VGG16* process raw images[17], *VGGish*[21] is a neural network built with *convolutional layers* that achieves excellent results when taking raw audio files from AudioSet and performing various discriminatory tasks with them (for example, the *audio event detection* classification task). Thus, the same *unsupervised* feature extraction process can be performed with *VGGish* with the same result - a compact, but information-rich feature descriptor for an audio frame.

However, the additional challenge with raw audio signal processing - is that it does not come in the convenient extraction ready form that raw visual signals do. Rather, audio signal is one-dimensional and sampled at a very high *sample rate* (in the case of our movie trailers 44,1K per second) and thus some preprocessing must be performed before it can be provided to *VGGish*.

The solution to this preprocessing challenge is to create *mel spectrogram* features from the raw audio signal first. In brief, a spectrogram is a representation of the spectrum of frequencies of a signal as it varies over time. They are typically computed using Fourier Transforms to convert the audio signal to the frequency domain and then gathering together these frequencies over time.

The *mel* part of *mel spectrogram* refers to converting these resultant frequencies to the *Mel Scale*. This is an alternative frequency space which has been intentionally designed to place frequencies at distances from one another that match how humans perceive the distance between sounds at those frequencies. It thus puts a lot of emphasis on sounds recorded that are in the human audible range.

2.2.3 Learning from sequences of audiovisual features

Recurrent neural networks

While traditional neural networks (such as the two-dimensional *visual* CNNs discussed in the previous section) have many powerful applications, one thing that they lack is the ability to persist state between items of data belonging to the same sequence. In other words, they have no memory capacity.

This can be a severe limitation when working with data that has temporal dependencies (for example, the different scenes within a trailer) but it can be solved by making use of a branch of *memory-enabled* neural network architectures known as

[17]It should be noted that the main conceptual difference between these approaches is that whereas image CNNs operate on two-dimensional images at a single timestep, audio CNNs operate on a one-dimensional audio signal with the second dimension being time. Despite this deviation in the semantics of the dimensions, convolutional layers are just as effective at learning meaningful representations of the signals.

recurrent neural networks (RNNs).

Recurrent neural networks extend the standard neural network equations (which can be stated concisely as $\hat{y} = f(W_{hy}^T h(x)+b_y)$ and $h(x) = f(W_{xh}^T x+b_h)$) by introducing a *recurrence* term into the formulation whereby each hidden state at time step t is calculated by considering both a contribution from the input at t and a contribution from the state of the network in the previous time step $t-1$. More formally

$$h_t = f'(W_{xh}^T x_t + W_{hh}^T h_{t-1} + b_h) \tag{32}$$

$$\hat{y}_t = f(W_{hy}^T h_t + b_y) \tag{33}$$

Here W and b are the learnable parameters of the network and h represents the activation of hidden layers in the network. Perhaps more intuitively, the network control flow this creates is visualised in Figure 7 where on the left hand side the network is visualised in its recurrent form and on the right hand side it is *unrolled*, showing how state propagates through the system.

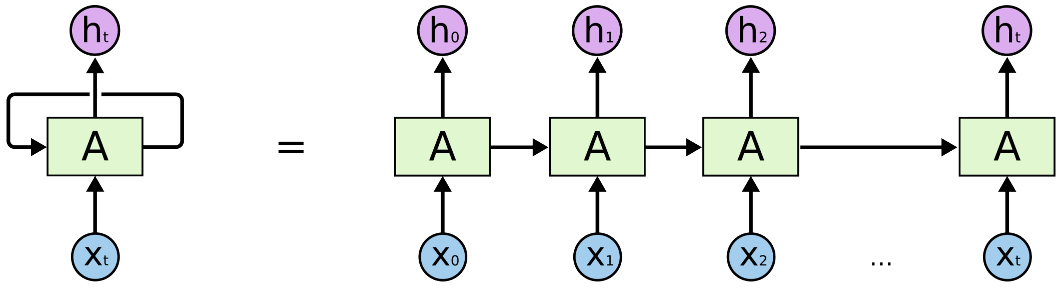


Figure 7: The conceptual structure of a recurrent neural network (RNN). The diagram on the right shows it in *unrolled* form.

The training of RNNs is not dissimilar to the training of other neural network architectures. *Gradient descent* and *backpropagation* are used to minimise the error of the network relative to ground truth, with the only difference for RNNs being that the gradient is propagated back through the hidden layers at t and to the hidden layers at $t-1$.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \tag{34}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \tag{35}$$

In these equations E represents the error. Equation 34 represents the fact that the total error in ths system is the sum of each error at each time step[18]. Equation 35

[18]However, it is important to note that there is only an error at a time step if there is a corresponding label at that time step, which is not always the case depending on the problem the network is addressing.

shows how the error is backpropagated through the system (which has a temporal contribution recalling the recurrence in Equation 32).

Long short-term memory networks (LSTMs)

However, recurrent neural networks have a major shortcoming when used in the form of Equations 34 and 35 - namely, the *vanishing gradient problem*[19].

This refers to the phenomenon of the gradient in the error (with respect to the network parameters) becoming very small as it is backpropagated further back through the network. It tends to happen when the RNN has grown to a certain length and it is a problem as it means none of the network parameters receive any error gradient signal with which to update themselves after a certain point in the network. Thus, training stalls in a far from optimal state. This has the disappointing consequence that RNNs cannot be used to modelled sequences beyond a certain length and thus long-term temporal dependencies cannot be learned.

Fortunately, there is a solution to this: modifying the architecture of the network to that of a *long short-term memory network* (LSTM). The conceptual architecture of such a network is displayed in Figure 8.

LSTMs have been designed in such a way so as to mitigate the *vanishing gradient problem* and thus long-term dependencies in the data are easily learned and retained throughout training. They achieve this by the carefully configured cell structure displayed in Figure 8.

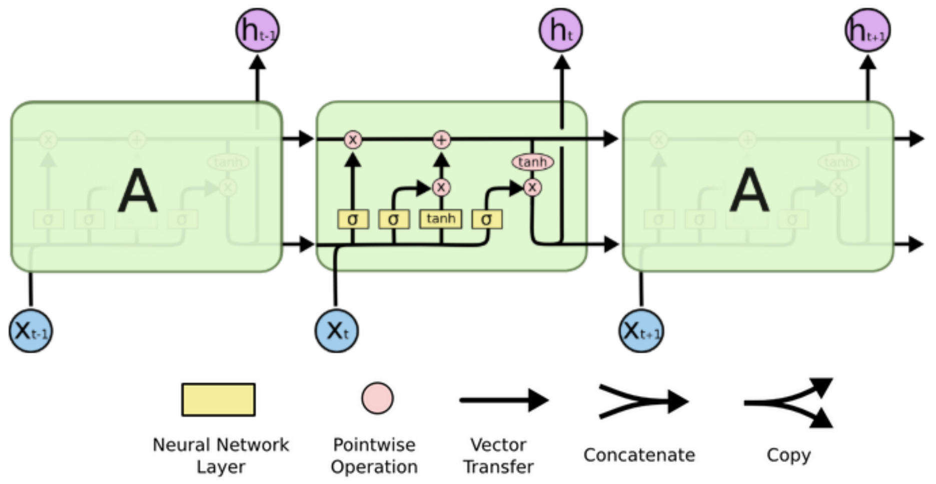


Figure 8: The conceptual structure of a long short term memory network (LSTM). With explicit thanks to Christopher Olah[50].

[19]It is worthy of a quick note that the opposite can also happen to the network - the so called *exploding gradient* problem - with similarly disastrous repercussions. Whether one or the other occurs depends on the extent of the recurrent weights matrix W_{hh} from Equation 32

At a very high-level, the configuration of logic gates within the cell - both those consisting of pointwise operations and learnable neural network operations - allow it to learn a strategy on what information to keep, what information to forget, what information to output, and how to update the cell state. It achieves each of these in turn by the following series of equations

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && (36)\\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && (37)\\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && (38)\\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && (39)\\
h_t &= o_t \otimes \tanh(C_T) && (40)\\
C_t &= f_{t t-1} + i_t \otimes \tilde{C}_t && (41)
\end{aligned}
$$

Namely, Equation 36 is the *forget gate layer*, Equations 37 and 38 are the *input gate layer*, Equations 39 and 40 are the *output gate layer*, and Equation 41 represents the update in cell state (where X_{t-1} is the output of the cell at $t-1$). C_t represents the state of the LSTM cell at time t and is a separate state from the input data x_t and the hidden weight activations h_t.

Thus, a network structure is born that can learn how to prioritise what information it retains over time and which can be used on sequences of arbitrary length without falling foul of the *vanishing gradient problem*. For more detail on how exactly the gates defined above allow the LSTM to achieve this please reference the original paper[20].

At this juncture we should also briefly mention that *gated recurrent units* (GRUs)[15] were also considered as a recurrent network architecture for this work. They provide similar performance to LSTMs with a largely similar architecture. While they could be equally valuable in the task of audiovisual content learning we chose to persevere with LSTMs because of their prevalence in the audiovisual learning literature.

Long-term recurrent convolutional networks (LRCNs)

Long-term recurrent convolutional networks (LRCNs)[13] are a type of deep network architecture that excel at audiovisual learning tasks. They combine the best of the CNN and RNN architectures discussed in the previous sections and are as such often referred to as *doubly deep* in the sense that they are deep in both the spatial and temporal dimensions.

This architecture type has been put to effective use in a broad range of visual information processing tasks such as activity recognition, video captioning and human emotion detection from facial expressions[18] with excellent results versus non-recurrent baselines[13].

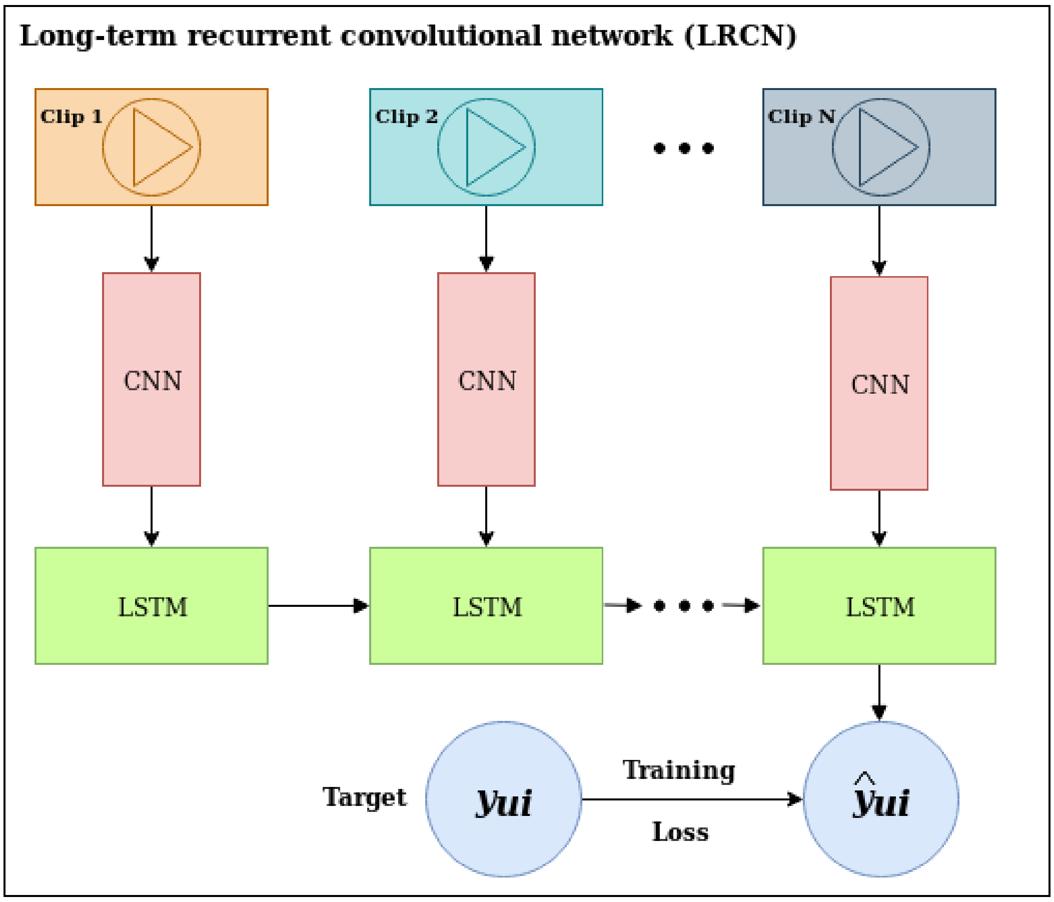


Figure 9: The conceptual structure of a long-term recurrent convolutional networks (LRCNs)

Broadly speaking the tasks which it can perform can be categorised into three main types

1. Sequential input, static output: $\langle x_1, x_2.., x_T \rangle \mapsto y$. Where, for example, the goal would be to predict an overall label for the video taking into account all of its frames

2. Static input, sequential output: $x \mapsto \langle y_1, y_2.., y_{T'} \rangle$. Where, for example, a static image must be categorised with a sequence of labels - something that arises when one needs to caption an image

3. Sequential input, sequential output: $\langle x_1, x_2.., x_T \rangle \mapsto \langle y_1, y_2.., y_{T'} \rangle$ where, for example, the goal would be to predict a sequence of labels for a video based on a sequence of input frames (noting that $T \setminus T' \neq \emptyset$ is permitted)

Briefly, LRCNs achieve their performance increase over non-temporal visual baselines on these tasks by allowing the visual representations that they learn to include temporal dependencies. As video content typically gets a lot of its semantics from the sequence of frames it contains - rather than from individual frames in isolation - it follows that an architecture that can model how this content changes over time is a better model for the system.

As our dataset provides one label per movie trailer we focus on use case (1) of LRCNs extending the reasoning to also model audio content over time. The conceptual architecture for this use case is displayed in Figure 9.

It should also be noted that we operate the LRCN architecture in a *two step* fashion, whereby we first extract visual and audio features from the trailers and then feed them to the LSTM regression model. The rationale for this was stemmed from the size of our audiovisual input (typically 180s long with visual streams at 25 FPS and audio at 41,1K FPS with over 10K trailers in total).

While a *one step* LRCN architecture can be used on data of this size it presents a significant compute and RAM overhead which is not a necessary challenge for us to tackle to prove the viability of our approach. More details on our approach can be found in Sections 3 and 4.2.

2.2.4 The semantic gap

Finally, no review of the audiovisual processing literature would be complete without quickly addressing the notion of the *semantic gap*[3, 46] between the low-level representation of audiovisual content and how users actually perceive and interpret it.

In reality, the size of the semantic gap usually depends on a trade-off between the level of interpretability and the level of descriptive power to endow a model with. In the context of content-based recommender systems (see Section 2.1.3), interpretability has historically been achieved by learning from human encoded content descriptors (such as genre and sentiment tags) which by definition have semantics that humans can understand.

However, these labels are limited in their expressiveness - for example, two films from the same genre may appear similar from their high-level tags but in practice they actually provide a very different audiovisual and emotional experience to their viewers. Moreover, human labellers are expensive and, as the amount of video content available online is growing at an exponential rate, they cannot solely be relied upon to provide interpretations to video content[20].

As a result, researchers and companies are looking towards the low-level video representation learning techniques discussed in the previous sections to provide a more granular representation of the video content and to do so in an automated and scalable fashion. However, with this additional descriptive power comes the downside

[20]For audiovisual content growth figures please reference: `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html`

of less tangible interpretations for the new dimensions - hence the trade-off.

Unfortunately, there is not yet a *silver bullet* solution to bridge this gap when working with low-level content features. However, there are some heuristics that can be followed to narrow the gap such as using feature descriptors for inputs from models that have achieved semantically understandable results and formulating the problem in such a way so as to learn as many correlations as possible between low-level content and human labels.

In our work we follow these guidelines where we can. However, formally designing and assessing our system to quantify its semantic gap is saved for future work (see Section 6.2).

3 Our approach

3.1 Using audiovisual content to enhance collaborative filtering models

As outlined in Section 1.2, our research questions whether audiovisual trailer content has a part to play in improving the quality of recommender systems. Because collaborative filtering models have achieved the best results for over a decade - and because their deployment is ubiquitous in industry - we choose to focus our work on how audiovisual content can enhance this type of recommender system specifically.

This research aim thus straddles our work between the two exciting new fields of neural collaborative filtering[23] - which achieves state of the art recommendation accuracy results[**dlrsreview**] - and deep learning for automated audiovisual content analysis and interpretation[5].

As both of these areas base their work on a deep learning toolkit we are encouraged to continue working with these technologies and architectures for the benefits this will bring in terms of being able to replicate the results of their work and to more easily achieve interoperability between the different modules of our model (as everything can be written in the same deep learning library).

Regarding the taxonomy of recommender system types discussed in Section 2, our research aims guide us in the direction of designing a *hybrid* recommender system. Specifically, our model can be seen as a *feature combination hybrid*[4] as we leverage information from collaborative and content-based sources to reach a single prediction of user rating.

The following subsections outline the theory of our proposed approach.

3.2 Theory of our proposal

To assess our hypothesis that audiovisual content information can enhance collaborative filtering models we devise a model for which the target function is defined as

$$\hat{r}_{ui} = f(P^T v_u^U, Q^T v_i^I, f_i(t_i \in T_i))|P, Q, \Theta) = \phi_{out}(\phi_X(...(P^T V_u^U, Q^T v_i^I, f_i)...)) \quad (42)$$

Here, all of the terms are defined as in Equation 13 with the exception of $f_i(t_i \in T_i)$ which is a new longitudinal feature term representing the audiovisual content of the item over the duration of the trailer (where t_i represents a temporal portion of trailer i and T_i represents all time portions within trailer i).

Equation 42 can be stated in an alternative form so as to emphasise the different "branches" of our model for the collaborative filtering and audiovisual content information

$$\hat{r}_{ui} = \phi_{out}(\phi_X(...(\phi_{CF,N}(P^T V_u^U, Q^T v_i^I) + \beta\phi_{AV,M}(f_i))...)) \tag{43}$$

$$\phi_{CF,N}(P^T V_u^U, Q^T v_i^I) = \phi_{CF,N-1}(...\phi_{CF,2}(\phi_{CF,1}(P^T V_u^U, Q^T v_i^I))...) \tag{44}$$

$$\phi_{AV,M}(f_i) = \phi_{AV,M-1}(...\phi_{AV,2}(\phi_{AV,1}(f_i))...) \tag{45}$$

In these equations, N, M and X represent the number of hidden layers in the collaborative filtering, audiovisual and "joined" branches of the network respectively. β is a weighting term that controls the influence of the two branches when they are concatenated together.

The audiovisual content is in the form of movie trailers which on average consist of about 180 seconds of audio and visual frames at 44,1K and 25 frames per second, respectively. Therefore, using the data in this raw form is not feasible due to the *curse of dimensionality* and the negative consequences this brings (as discussed in Section 1.2). Thus, it is essential that our model incorporates a dimensionality reduction approach that reduces the feature space volume while preserving its variance and thus the ability to use it in predictive models.

We therefore propose a model that reduces the dimensionality of the raw features before they are provided to the model that implements Equation 42. This model has the form

$$f_i(t_i \in T_i) = \phi_e(t_i' \in T_i' | \Theta_e) \tag{46}$$

Where $t_i' \in T_i'$ (where $|T_i'| >> |T_i|$) represents the features in their full-dimensional space and ϕ_e represents the dimensionality reduction model (with model parameters Θ_e).

As finding the most *semantically meaningful* reduced dimensionality representation of audiovisual content is a full research project in its own right - and not the focus of this project - we opt to draw from the relevant literature on the semantic reduction and segmentation of audiovisual content (as summarised in Section 2.2) rather than to propose anything novel on this front.

Therefore, we program the parameters for the model in Equation 46 *a priori* based on the findings of these works, making minor adjustments during the experimental works to ensure the extracted features are preserving a satisfactory level of semantics. The full details of how we implemented this submodel can be found in Section 4.2.

We elaborate on the theory of how we propose to implement Equation 42 in the next section.

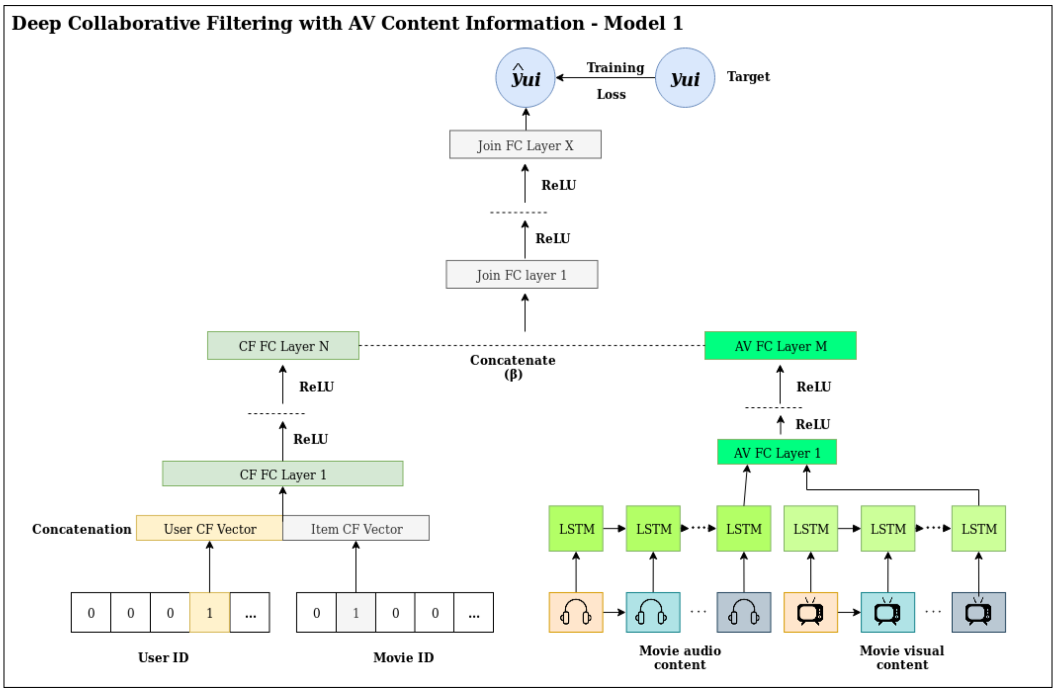


Figure 10: Deep Collaborative Filtering with AV Content Information - Model 1.

3.3 Our proposed network architectures

To combine audiovisual content with collaborative filtering data we need a model architecture that has a module for both modalities and then a *fusion module* that combines their outputs into a final prediction of the user review. As such, we propose the following network architectures (depicted graphically in Figures 10-12)

1. **Deep Collaborative Filtering with AV Content Information - Model 1**

   - This model combines the core of the neural collaborative filtering model discussed in Section 2.1.2 (Equation 13) with the core of the long-term recurrent convolutional network (LRCN) model[21] discussed in Section 2.2.3
   - The audiovisual content branch of the model consists of a separate LSTM sub-branch for the audio and visual modalities followed by dense layers to fuse them. The rationale for separate modality sub-branches is that a) it ensures neither modality dominates the other at the input stage (it should be noted that visual features are 4096 dimensional whereas audio features are 128 dimensional) and b) that additional modalities could in future be added to the content branch with minimal rework (see Section 6.2)

[21]However, it should be noted that we operate our LRCN model in a *two step* fashion by placing the convolutional unit in the feature extraction process and the recurrent unit in the collaborative filtering model. As such, the weights of the convolutional part of the model are not learnable. Operating the model "as one" would be an exciting extension to the work but presents compute and memory challenges due to the size of the raw trailer features.

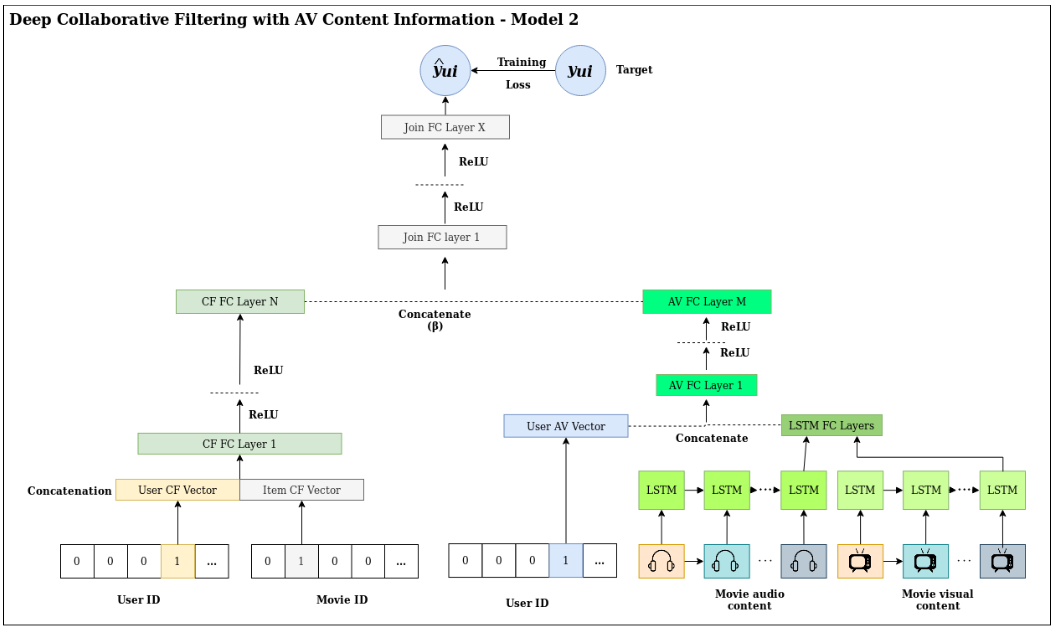


Figure 11: Deep Collaborative Filtering with AV Content Information - Model 2.

- The logic behind the overall model architecture is to allow both branches to transform their respective input features in the early layers of the network before concatenating them at a certain point in the architecture and applying more hidden layers with non-linear activation functions after the concatenation point
- Thus, the model is able to learn the most relevant way to fuse these modalities and we treat the point at which the two branches are concatenated as a hyperparameter in our experimental approach (β in Equation 43)

2. **Deep Collaborative Filtering with AV Content Information Model 2**

- This model is similar to (1) but varies in the sense that there is a different user embedding module for both the collaborative filtering module and the av-content based module. This makes the architecture closer to that proposed by the authors of [52] who assess how images of products can enhance collaborative filtering models for online marketplaces (such as Amazon.com)
- The logic behind having a second user embedding module is that it should allow the av-content branch of the model to take into account individual user effects early in its transformation of the input features which may lead to better predictions
- A further argument for adopting this architecture is that it facilitates pre-training of the model's individual components - an approach that is well-known to enhance a model's learning capacity[10] and that has achieved promising results in other neural collaborative filtering applications[23]

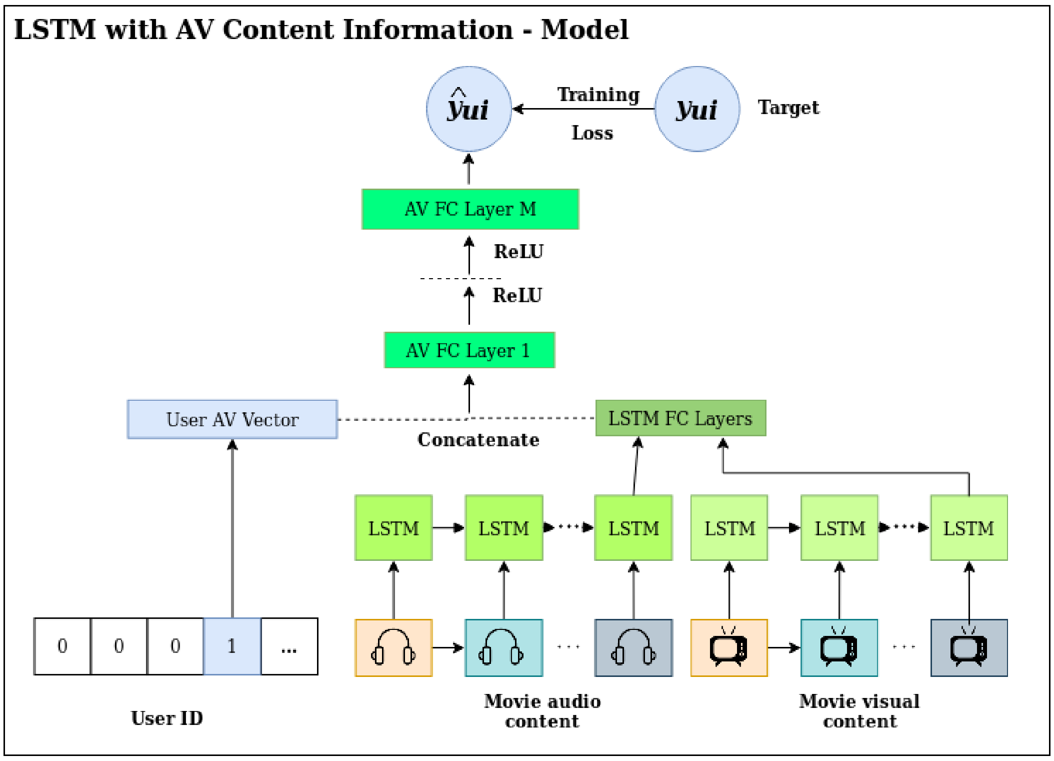


Figure 12: LSTM with AV Content Information Model.

- The pre-training approach with this architecture consists of first training the optimal deep collaborative filtering branch and av-content branch separately before combining them into this model's architecture and resuming model training

3. **LSTM with AV Content Information Model (LstmCont)**

   - Finally, this model just contains the av-content branch of (2) with a final dense layer to allow it to predict. The logic for separating this branch into its own model was a) to allow us to isolate and analyse the effect of incorporating audiovisual features into the model, and b) to facilitate the pretraining of (2) discussed above

All of our models are parameterised with non-linear hidden layer activation functions (such as RELU) and additional layers (such as *dropout* layers) that are known to promote a deep model's learning and generalisation capabilities. The cost function used to train the network is discussed in the next section.

3.4 Our formulation of the rating prediction problem

As discussed in Section 2.1.1, the recommender system problem can be formulated in many ways. Therefore, it is important to specify how we have cast it in our approach.

We treat the problem as a *matrix completion* problem where the rating matrix contains *explicit ratings*. As such, the cost function for our model is

$$e_{ui} = \sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \sum_{w} w^2 \tag{47}$$

Where $\hat{r}_{ui}$ are the ratings our model predicts, r_{ui} are the ground truth rating, and λ is a regularisation term we apply to our network weights (to prevent overfitting).

As this formulation of the problem makes it a regression - which is intended for values in a continuous and unbounded domain - whereas our rating domain is ordinal between 0.5 and 5.0, we constrain the prediction of our model by applying the sigmoid activation function to its final neuron output (thus forcing it to lie in the range [0,1]) and then scaling this value to lie in the range [0.5, 5.0]. The final layer of our model is thus parameterised as

$$\hat{r}_{ui} = \sigma(\theta_X(P^T v_u^U, Q^T v_i^I, f_i(t_i \in T_i))) * (R_{max} - R_{min}) + R_{min} \tag{48}$$

Where σ represents the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$), R_{min} and R_{max} are the min and max ratings in the training data, and $\theta_X(P^T v_u^U, Q^T v_i^I, f_i(t_i \in T_i))$ represents the output of the final hidden layer of the network.

We briefly considered encoding the data in such a way so as to cast the problem as an *ordinal regression* problem[39] but found this was not a common approach in the literature. Moreover, we considered stating the problem for implicit rating prediction but assign this to future work, reassured by the literature that our overall architecture[23] could be easily modified to generate these types of predictions.

Finally, detailed information on how we prepared the rating data to assess the *item non cold* and *item hard cold* start scenario can be found in Section 4.1.2.

3.5 Elements not considered in the scope of our project

Finally, we quickly summarise possible modifications / extensions to our work that were considered but not included in the scope of this project for various reasons

- **Different video summarisation methodologies**:
  - On the visual stream side we considered even more recent architectures than LRCN that achieve state of the art results[7, 9]. These are based around “3D convolutional” units with the extra dimension representing time. While it is highly likely that the embeddings these models can produce would be a more meaningful summary of visual video content, running these models requires far greater compute, memory and parameterisation effort

  - On the audio stream side we considered using *i-vectors* in place of embeddings extracted from the *VGGish* model given their prevalence in the audio classification and speaker recognition literature[17]. However, after an in-depth assessment felt that the parameterisation process (whereby a global extraction model must be trained on the training data before individual embeddings can be extracted) risked making our results too specific to our chosen dataset. Moreover, the strong results of *VGGish* - a very recent model - versus baselines such as *i-vectors* on many audio tasks suggest it will usher in a paradigm shift in how audio content is represented

- **Different content modalities**
  - We considered adding additional content modalities to our architecture and workflow such as the script of the trailers. However, as this was not essential to prove the validity of our approach we assign this extension to our future work

- **User cold start scenarios**
  - We also do not consider any *user cold start scenarios* in our work which are an important issue that collaborative filtering based recommender systems face. Our rationale for this is that it is unlikely adding information on the audiovisual content of the items will address the *user cold start scenario*. However, combining such capabilities with our model - most notably through recent *active learning* techniques[48, 31] - would be an exciting piece of future work

4 Experiments and results

4.1 Ground truth data overview

4.1.1 Data overview

In this sub-section we provide an introduction to the rating data that we use as *ground truth* in our experiments before discussing in detail in the next sub-section how we sample from this data to create meaningful model assessment scenarios.

Given we are interested in the interplay between audio-visual film content and user enjoyment we use the MovieLens dataset[34] as our primary source. The MovieLens project provides different sized sets of movie ratings (and associated metadata such as genre tags) for researchers to use freely.

Of the sets available, we opted to use the “MovieLens 20M” dataset for two reasons: firstly, it is the largest and most up to date version of their data, thus giving us as many ground truth data points to work with as possible, and secondly because GroupLens provide a list of YouTube *urls* that link a significant portion of the movies in the dataset to their trailers on YouTube. Therefore, the task of retrieving and processing the trailers for each movie was made more efficient by this mapping.

During our trailer download work (see Section 4.2), we found that not all movies in the “MovieLens 20M” dataset had high quality trailers available on YouTube (for example, the links provided might be broken or not available in our region) but that a significant portion did. Thus, rather than searching elsewhere for missing trailers we opted to work with a marginally smaller version of the original dataset.

For clarity in the discussion we term our version of the dataset “MovieLens AV” (AV for “audiovisual”). Table 3 provides a high-level summary of the difference between our ground-truth rating set and the original.

Table 3: Summary of MovieLens AV dataset versus original MovieLens 20M dataset[22]

	$\lvert M\rvert$	$\lvert U\rvert$	$\lvert R\rvert$	$\frac{\lvert R\rvert}{\lvert M\rvert}$	$\frac{\lvert R\rvert}{\lvert U\rvert}$	**Years**	**Genres**	**Scale**
ML 20M	26,7K	138,4K	20.0M	747.8	144.4	1900-2015	20	0.5-5.0
ML AV	12,3K	138,4K	13.1M	1062.7	94.8	1970-2015	20	0.5-5.0

It is important to highlight several interesting characteristics of the ground truth rating data in this section as they explain how and why we parameterised our experiments in the way we did (which we present in the subsequent sections)

- **Decision to take a subset of available years**: despite having trailers available for movies all the way back until 1900 we decided to stop our assessment of

[22] $\lvert M\rvert$ is number of movies, $\lvert U\rvert$ is number of users and $\lvert R\rvert$ is number of ratings. NB: more than one genre tag can apply to a single movie - see Figure 13.

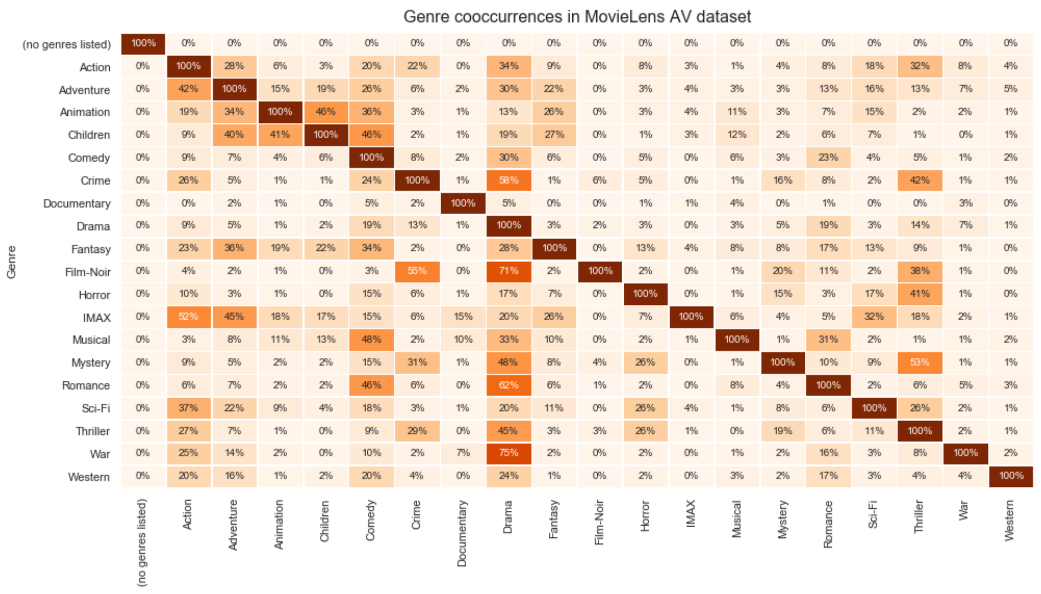


Figure 13: Cooccurrence of genres in MovieLens AV dataset.

trailers at 1970. The rationale for this is that the trailers from before 1970 were a) often in black and white (which would require us to correct for the effect of colour in our visual feature extraction pipeline), and b) often of notably different quality than trailers after 1970 (for example, many of the videos were of lower audio-visual quality as well as editing quality).

- **Rating value distributions**
  - *Overall*: the overall rating value distribution is displayed in Figure 14. Visualising this distribution is important as it allows us to recognise the general bias of users towards higher ratings (and reluctance to give very low ratings). While we do not explicitly correct for this in our models - as to do so would introduce a modelling bias that is not representative of the ground-truth - it is very helpful when interpreting our results
  - *By decade*: the summary statistics for rating values by decade are displayed in Figure 15. From this we can draw two important insights
    1. It is clear that the central tendency of the entire dataset (both mean and median) is towards 3.5. Thus, when deciding on a value to use as the *relevance/recommended* threshold value for our ranking results it made most sense to take this value (see Section 2.1.6 on ranking metrics)
    2. There is limited difference in the rating value distributions between decades (with the exception of 1970-1979, which has a larger proportion of higher rated movies). This is relevant as when designing and initially training our model we worked on subsets of the data by decade (for example, the last two decades) and thus knowing the rating distribution of this sample matched the population distribution

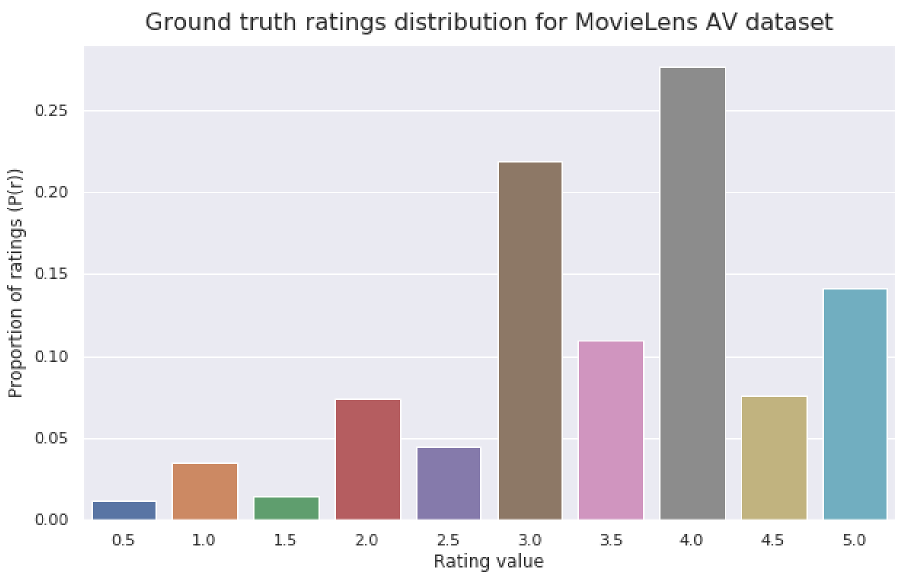


Figure 14: Ground truth rating distributions by value in MovieLens AV dataset.

gave us confidence that our model and results would scale to larger datasets

– *By genre*: we also analysed the rating distribution by genre as shown in Figure 16.a which shows comparatively more variance in rating, for example with Film Noir and War films generally being rated the highest and Horror films being rated the lowest. This distribution is most descriptive when used in conjunction with Figure 16.b - which shows the number of films by genre by decade. Together these plots give us insight into how the data is distributed in decade based subsets which gives more intuition when training the models. For example, we see that in the half-decade between 2010-15 IMAX films are far more prevalent than they are in any past decade and as they are rated on average higher this may widen our rating distribution. Similarly, in this same timeframe there are proportionally less Comedy movies - which tend to be rated slightly below average - which combined with the IMAX trend may explain why the average for 2010-15 is slightly above the previous two decades in Figure 15

- **Rating count distributions**: Finally, in Figures 17 and 18 we show several representations of the "long-tail" phenomena for items and users, respectively. We make extensive reference to how we use the knowledge from these plots to design the evaluation scenarios in the next subsection

4.1.2 Data scenarios

Given our research hypotheses that incorporating audiovisual content into collaborative filtering models would enhance recommendation quality (in particular in the

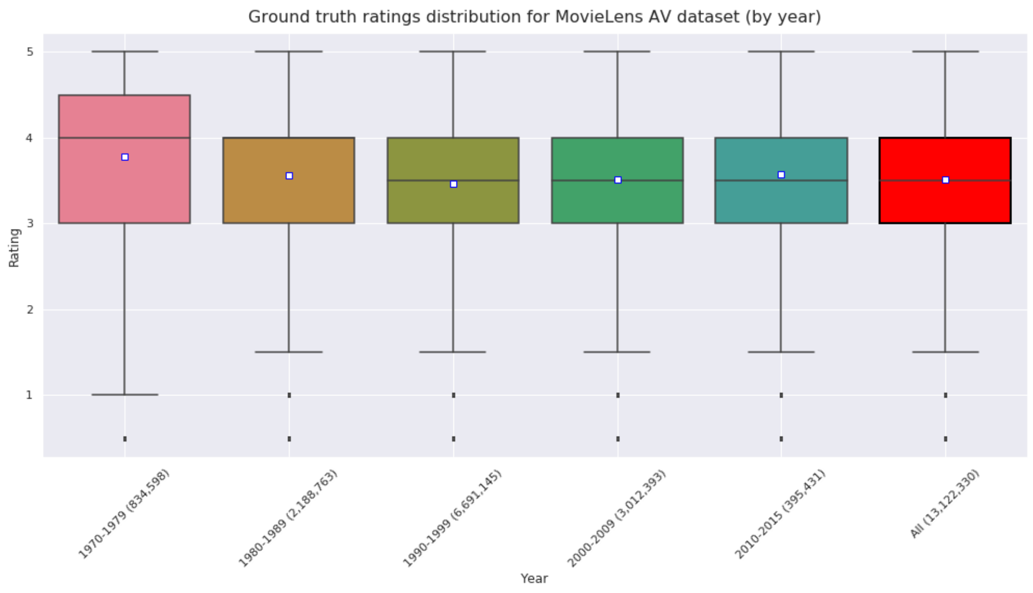


Figure 15: Ground truth rating value distribution by decade in MovieLens AV dataset.

item cold start regime) it was important that we set up data scenarios that isolated the impact of the audiovisual feature injection. To this end, we devised two scenarios in which to evaluate our models (which are depicted graphically in Figure 19)[23]

1. **Item non-cold start**: where all items in the *test set* are available during training time and there are many training reviews for each of these items (i.e. 50% of the data for the *test items* is added to the training set)

2. **Item hard-cold start**: where no items in the *test set* are available during training time, and thus the model only learns on items that it will not be evaluated on (i.e. 0% of the data for the *test items* is added to the training set)

Importantly, as we are interested in assessing the impact of audiovisual content on alleviating *item cold start* we impose a simple constraint on the number of reviews per user in the rating data (and in doing so we also constrain the matrix sparsity).

We do this by sampling a fixed number of reviews per user from the starting rating data[24] *before* performing a *train/validation/test* split for the given scenario. To provide at least some variability in user review sparsity we consider the settings where

[23]It should be noted that for meaningful comparison of results *between* scenarios we used random number generator seeds to ensure that the contents of the test set were exactly the same between scenarios. All of our results were tested and replicable from a variety of initial seeds and thus we are confident that are results represent the generalisation capability of our model.

[24]Given the size of the dataset (and as a result the training times) we started our scenario sampling from subsets of the rating data. We specifically created starting subsets of: a) the top 400 items and top 600 users, b) the top 1000 items and top 1500 users, and c) the top 4000 items and top 6000 users. Here “top” is defined as the items/users who have received/provided the most reviews. Overall we found that our model performs consistently over the size of the dataset and we plan to run our model on even larger datasets as part of future work.

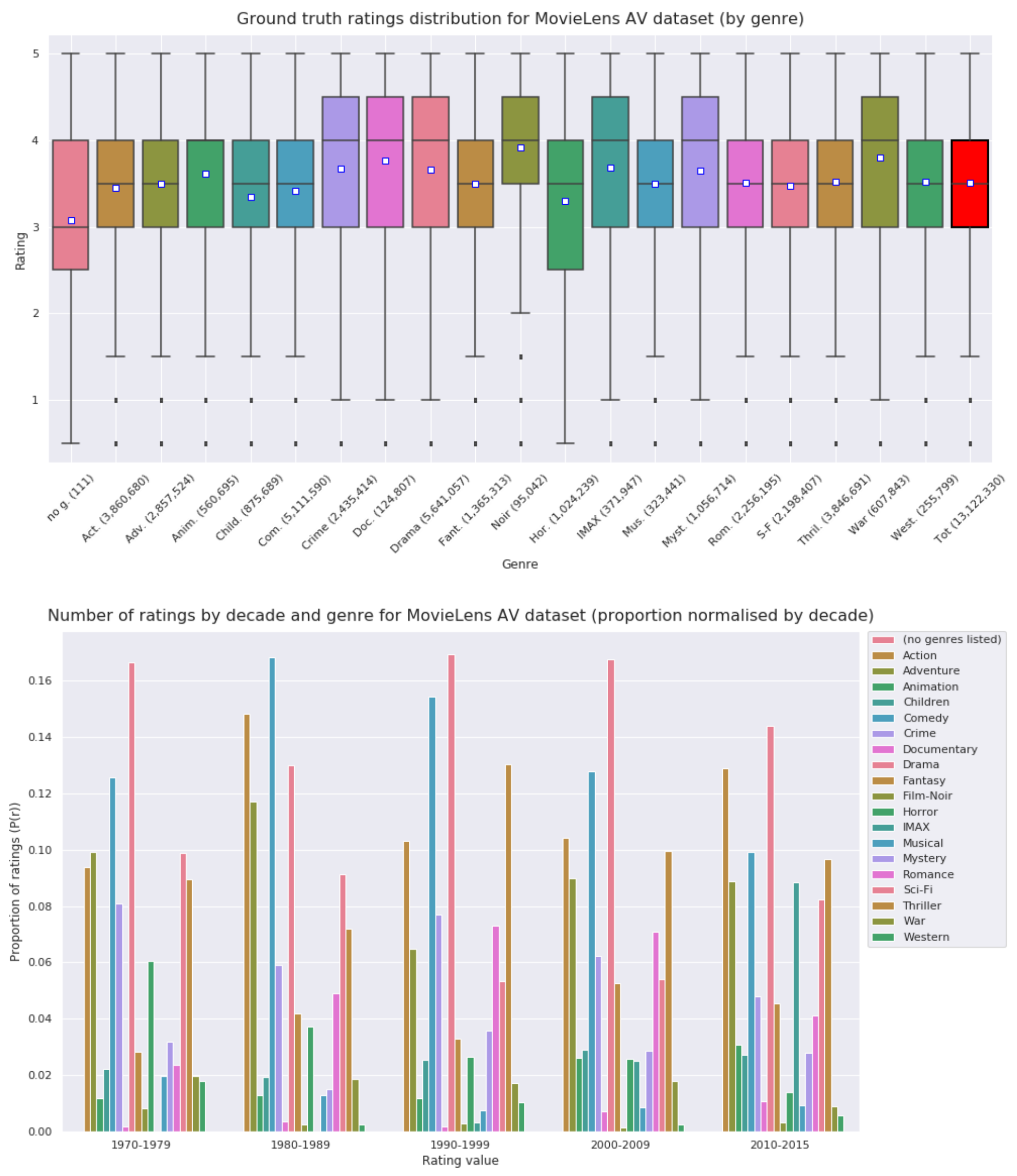


Figure 16: (a) The distribution of rating values by genre, and (b) the proportion of genre reviews per decade.

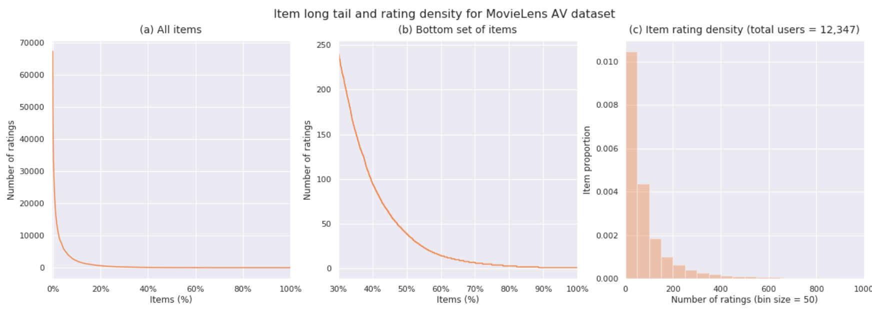


Figure 17: Movie rating count distribution in MovieLens AV dataset.

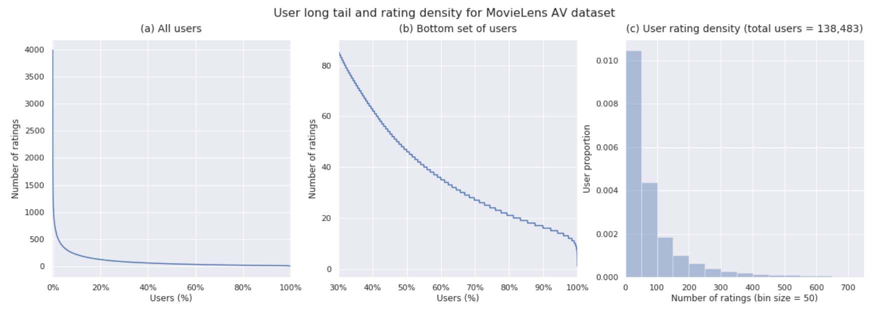


Figure 18: User rating count distribution in MovieLens AV dataset.

each user starts (before train/val/test split) with 20, 40 and 80 item reviews.

This user rating sampling strategy has 3 notable benefits: a) it removes any effects that may result from *user cold start* in the results (or indeed the opposite, where a small set of users may account for the majority of reviews) thus making our results more representative of item cold start, b) it imposes a rating sparsity that is representative for the majority of users and items in a company's catalogue (rather than of those with the most reviews), and c) it also reduces the size of the training data thus reducing model training times (which can be very useful when running hyperparameter optimisation).

By referencing Figures 17 and 18 we can position our scenarios relative to the actual density of ratings by items and users, respectively. Figures 17.a and 17.b show the movie "long tail" for all items and the bottom 70% of items, respectively. Our scenarios test the model at different positions on this curve. So, for example, the *item non-cold start* scenario tests the model towards the left hand side of the curve whereas the *item hard-cold start* scenario tests the model to the extreme right of the curve, where the items have not received any reviews yet.

For the additional user sparsity adjustment Figures 18.b and 18.c are the most informative plots. By limiting the number of ratings per user to between 20 and 80 we can see that our model is representative of the middle 50% of users (between 30% and 80%) in the data - in other words the users who have provided neither the most nor the least ratings.

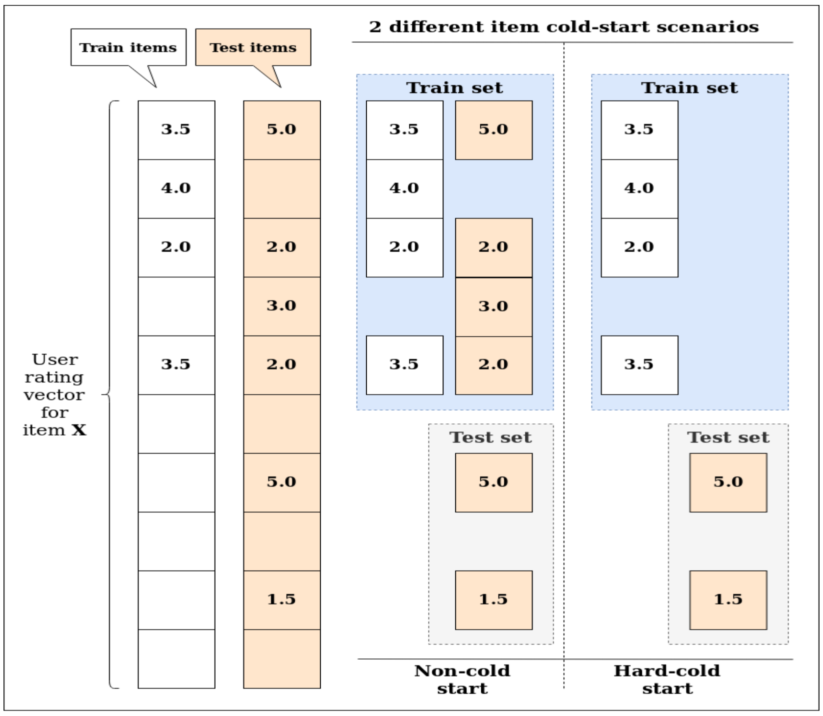


Figure 19: Data scenarios used to assess the recommendation capacity of our models.

Additionally, it should be noted that while maintaining these scenarios we strictly follow the *cross-validation* requirements for recommender systems (and machine learning models more generally). That is to say, we create a *train*, *validation* and *test* fold in our data before training the model and only allow the model to "see" the *train* data for weight updates and *validation* data for hyperparameter tuning (by way of calculating the validation loss every *N* training epochs and updating e.g. the learning rate accordingly) during training time.

The *test* data is then only used for evaluation of the model at the end of training - i.e. it is at no point used in the training process - so that our reported results are representative of the model's ability to generalise to new data. As our data is large we create two folds for our experiments; therefore, all items are evaluated in the test set exactly once.

4.2 Audio-visual feature extraction

4.2.1 Experimental approach

Given the length and complexity of movie trailers as an input type the feature extraction portion of our work was not trivial.

With the MovieLens AV dataset - that consists of 12,3K trailers of ∼180 seconds in length - we had to build a feature extraction pipeline that could process ∼615 hours of audiovisual data (or ∼55,3M individual visual frames and ∼90,7B individual audio frames).

In Appendix D.2 we briefly discuss the engineering challenges faced and resulting system design as they are important learnings from the project; in this section we focus on our implementation of the feature extraction theory introduced in 2.2.

The overarching goal of the feature extraction process - as discussed in Section 1.2 - was extracting features for subclips of the movie trailers that were both i) *perceptually complete* and ii) *semantically meaningful*.

To be *perceptually complete* was relatively straightforward - we chose to initially extract one feature representation for each media type through which a user experiences a trailer. Hence, the decision to have an audio and visual feature representation.

To be *semantically meaningful* was more complex. As discussed in Section 2.2.4 the phenomenon of the *semantic gap* is well documented in content-based recommender systems that use low-level audiovisual content and it is logical to conclude that this will affect our approach also. The issue is that it is hard to elicit whether or not a feature representation is meaningful or not until actually running experiments with it in the target domain (and even then the conclusions of these experiments may not be trivial).

Thus, we adopted a pragmatic approach in our feature extraction work whereby we extracted features by modality as per the state of the art literature recommendation for *semantically meaningful* representations and created variations of these features so that we could investigate how varying them impacted prediction results.

Then, before using our features in the rating prediction scenario discussed in Section 4.3 we applied simple analyses to them to check that at least some instances with different semantics were being extracted (the results of this are presented in Section 4.2), with the intention of leaving the assessment of whether these semantics were truly meaningful until the collaborative filtering stage.

The specific details of our feature extraction process are as follows

- **Visual feature extraction**: for our visual feature representation we extract *neural features* for each video frame (of dimensionality 4,096) by using a pre-trained convolutional neural network (this technique is discussed in more depth in Section 2.2.1). We opted to use the pretrained *AlexNet*[25] [25] model as it is relatively lightweight (in terms of number of parameters and thus memory requirements) while still being accurate. Thus, features can be extracted with a large batch size which is good for feature extraction run times. However, it

[25]Which is available pretrained from the PyTorch model zoo - `https://pytorch.org/docs/stable/torchvision/models.html`.

is worth pointing out that any other pretrained CNN (e.g. such as *VGG16*[61]) could be simply swapped into our architecture and this could be an interesting avenue of future work

- **Audio feature extraction**: for our audio feature representation we also extract *neural features* from fixed audio frame windows (resultant features are 128 dimensions) this time using the pretrained *VGGish* model[26][21]. The parameterisation of this model is to use 96 audio frames per *mel-spectogram patch* and one feature is produced for each *mel-spectogram patch*. The theory and parameterisation behind this model are discussed in detail in Section 2.2.2
- **Aggregation methods**:
  - *Aggregation technique and frequency*: as our goal in the rating prediction model was to use recurrent neural networks to learn the effect of temporal as well as audiovisual information in the trailers we need more than one audio and one visual feature per trailer. However, the frame-level features are too numerous (for example, there are approximately $180\,\mathrm{s} \times 25\,\mathrm{frames\ s^{-1}} = 4500\,$visual frames per trailer) and thus must be aggregated to a more manageable number. To do this we decide on a number of “subclips” of equal length[27] to extract per trailer (in our case, we consider 10, 20, 30 and 40 per trailer) and then use different statistics to aggregate the frames in each of these subclips. These aggregation schemes are as follows
    - Average - where the mean of the individual feature values in the subclips is computed
    - Average and variance - where the mean and the variance of the individual feature values in the subclip are computed and then concatenated into one single vector (of dimensions 8,192 in the visual case and 256 in the audio case)
    - Median - where the median of the individual feature values in the subclip is computed
    - Median and median absolution deviation - where the median and absolute deviation of the individual feature values in the subclip are computed and then concatenated into one single vector (with the same dimensions as the average and variance vector)
  - *Modality fusion*: we do not apply any modality fusion at this point in the feature extraction pipeline (rather we apply the fusion midway through our content-enhanced collaborative filtering model) but we do ensure that

[26]Which is available with the TensorFlow source code - `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`.

[27]To be clear, the subclips are of equal length *within* a single trailer - e.g. a trailer of 180 seconds would have 10 x 18 s subclips - but they are not necessarily of equal length *between* trailers. This is because we need to have exactly the same number of subclips per trailer to feed them to our LSTM architectures. As such, to provide some control on the comparative length of subclips we introduce a constraint in our audiovisual pipeline that no trailer can be longer than 5 minutes in length.

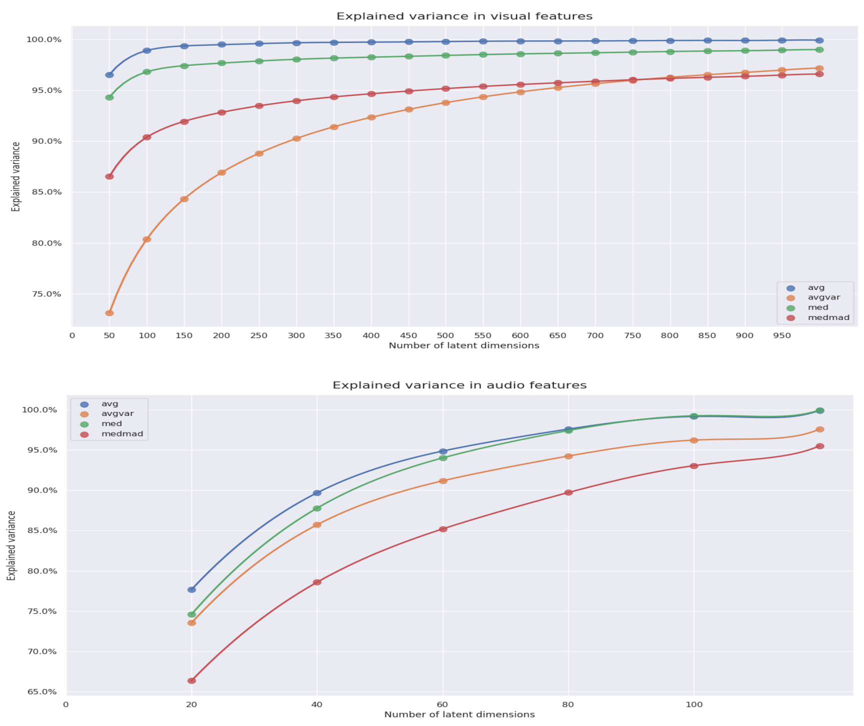


Figure 20: Explained variance left in latent dimensional data when applying incremental PCA to aggregated visual and audio data.

the aggregation scheme is entirely consistent between the audio and visual modalities (such that the subclips refer to exactly the same time point in the trailer)

- **Feature dimensionality reduction**: finally, given that the dimensionality of the visual features are still quite large after being passed through *AlexNet* and aggregated to the subclip level we opt to apply some linear dimensionality reduction to them as this greatly reduces the training times of the LSTM models we use (as well as reducing the memory requirements). When doing this we use an *incremental PCA* algorithm[28] [60] and ensure that we choose a large enough number of latent dimensions such that the majority of the *explained variance* remains in the lower-dimensional space (see Figure 20; as a general heuristic we worked with feature sets that maintained ≥ 95% of the variance in the data. So, for example greater than 600 dimensions was acceptable for the average and variance and median and mean absolution deviation sets.)

4.2.2 Results

In this section we present the results from our feature extraction approach.

[28]This is a version of the very well-known PCA algorithm that does not require all instances of the data to be in memory at once. Thus, it means that linear dimensionality reduction can be applied to data larger than can fit into RAM at once - which is the size limit of the standard PCA algorithm.

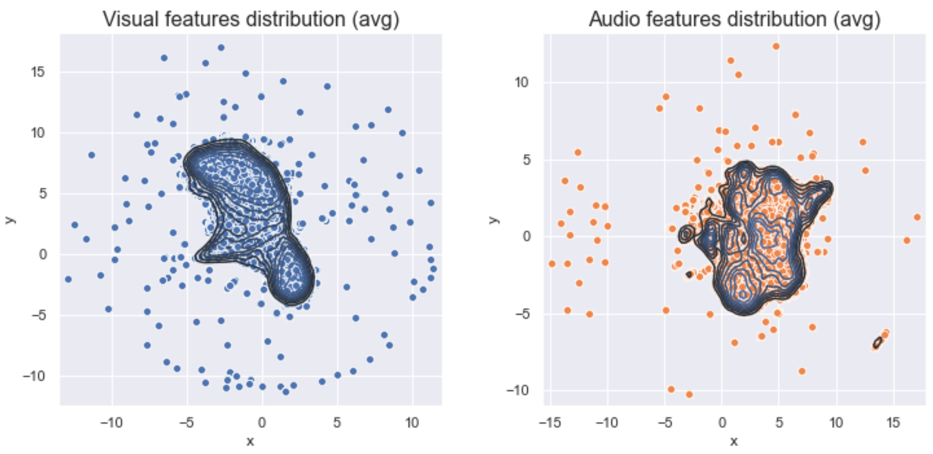


Figure 21: a) Visual and b) audio feature distribution after dimensionality reduction with UMAP.

Figure 21 shows a *UMAP*[29] reduction of the average visual and audio features, respectively, where 10 subclips have been extracted per trailer. The goal of non-linear dimensionality reduction techniques such as UMAP is to preserve the relative distances between data points from the higher-dimensional feature space when transforming the data to the lower-dimensional, latent space (which in our case is two-dimensional).

Thus, from the apparent clustering of visual and audio data points in Figure 21 - where areas of dense clustering are represented by many contour lines - it is apparent that our feature extraction approach has successfully extracted several sets of features with similar semantics.

However, plotting all of the data together in these plots obscures many of its underlying trends and thus to dive deeper into the contents and relations of our audiovisual features we break them down by time and context.

Figures 22 and 23 show how the distribution of visual and audio features change by clip number. From the visual features in Figure 22 three clear phases to the trailer can be seen. The first and last subclips tend to cluster by themselves, which suggests that trailers have similar beginnings and endings. Conversely, the middle eight subclips are consistent with each other in their clustering pattern - with two notable clusters - but their pattern is also more spread out over a large area of latent space, which suggests that the content of these subclips is more varied between subclips and trailers.

The audio distribution in Figure 23 is not dissimilar to this - with a notably dense

[29]UMAP[44] is a non-linear dimensionality reduction technique that operates with similar properties to t-SNE[47] but with considerably better computational efficiency - $O(N^{1.14})$ for UMAP vs $O(N^2)$ for t-SNE.

cluster in the first subclip and a homogeneous clustering pattern for the set of middle sub clips - but it also has a far less distinct final subclip.

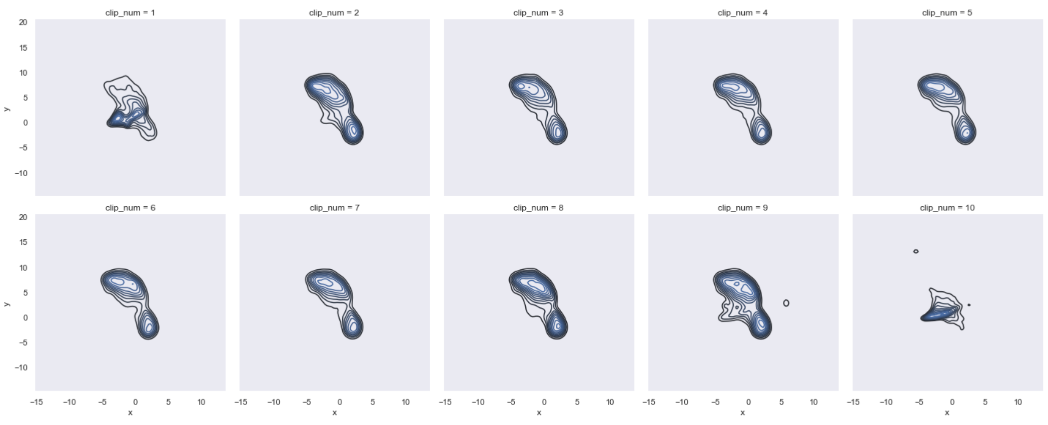


Figure 22: Visual feature distribution by sub clip number in MovieLens AV dataset after dimensionality reduction with UMAP.

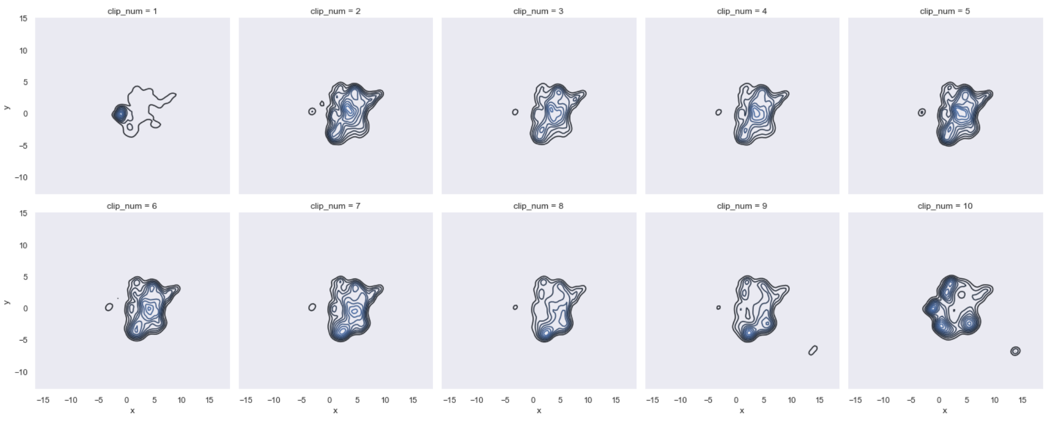


Figure 23: Audio feature distribution by sub clip number in MovieLens AV dataset after dimensionality reduction with UMAP.

A more quantitative analysis of the similarity of the subclip distributions was performed by computing the *Bhattacharyya coefficients*[30] between them. As the shading in Figures 24.a and 24.b show, the results of the quantitative analysis reinforce the

[30]The Bhattacharyya coefficient is a measure of the overlap between two statistical samples and it is often used in feature extraction work to gauge the level of separation between feature types[43]. The limits of the coefficient are $[0,1]$ with 0 suggesting there is no overlap (or similarity) between the distributions and conversely 1 suggesting there is perfect overlap.
For completeness, the Bhattacharyya distance is defined as

$$D_B(X_1,X_2) = \frac{1}{8}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)+\frac{1}{2}\ln\left(\frac{\det\Sigma}{\sqrt{\det\Sigma_1\,\det\Sigma_2}}\right)$$

Where μ_i and Σ_i are the mean and covariance of distribution X_i, respectively, and Σ is the mean of the covariances. The Bhattarcharyya coefficient can then be approximated by solving for $BC(X_1,X_2)$

conclusions we make by inspection of Figures 22 and 23.

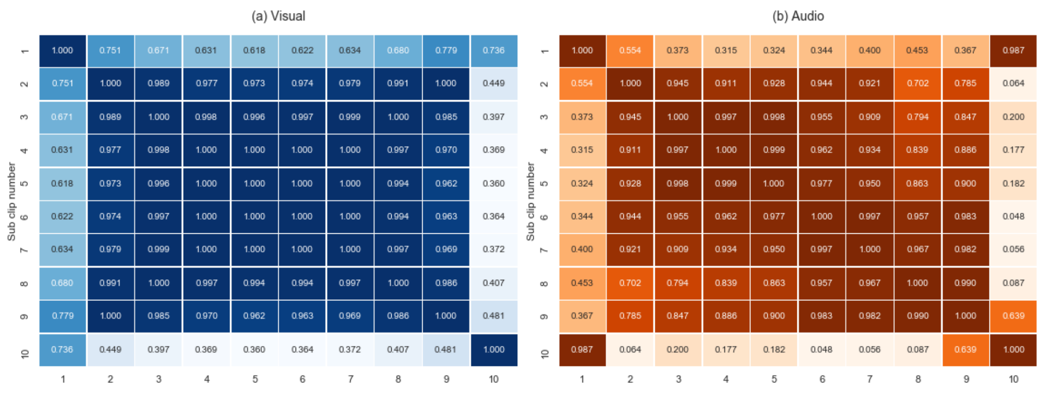


Figure 24: Bhattacharyya coefficients between subclip distributions for (a) visual and (b) audio features. Bhattacharyya coefficients are calculated on the latent space representation of the features and on the inner tercile of each sample.

Figures 25 and 26 break down the clustering patter by genre. Again, this more granular level of presentation helps to elicit some of the underlying trends in the data. Notable here, for example, is the dense clustering pattern for visual features relating to the *Animation* genre (in the top left corner of the subplot at the top right of Figure 25). This very dense clustering suggests that trailers in this genre have very similar visual aesthetics - which is immediately understandable from first-hand experience as animation as a content type is notably different from *real-world* content.

Beyond this notable trend there are some additionally interesting trends by genre - for example, from inspecting the audio distributions in Figure 26 we can see that: *IMAX* movies form a very dense cluster; *Children* and *Animation* films cluster together frequently (although the fact that multiple genre labels can be applied to a single movie should be taken into account here - recall Figure 13); *Drama* movies are the most varied in terms of audio content; and, *Documentary* movies seem to cluster around a centroid largely unique from the distributions of other genres.

Once again the quantitative analysis of genre samples using Bhattacharyya coefficients in Figure 27 echo the trends we conclude visually from the scatter distributions. However, the trends are much less pronounced for genre than they are for time step within the trailer. This is hardly a surprising finding because - as discussed in Section 2.2.4 - high-level content tags (such as genre) are a poor reflection of the actual audiovisual stimuli that a user experiences[31].

in the equation

$$D_B(X_1, X_2) = -\ln(BC(X_1, X_2))$$

[31]It should also be noted that the multivariate form of the *Bhattacharyya distance* we used to calculate the *Bhattacharyya coefficients* is parametric. In other words it assumes that the underlying distribution of the data is *Gaussian*. This is clearly not the case for our data - in particular for the genre

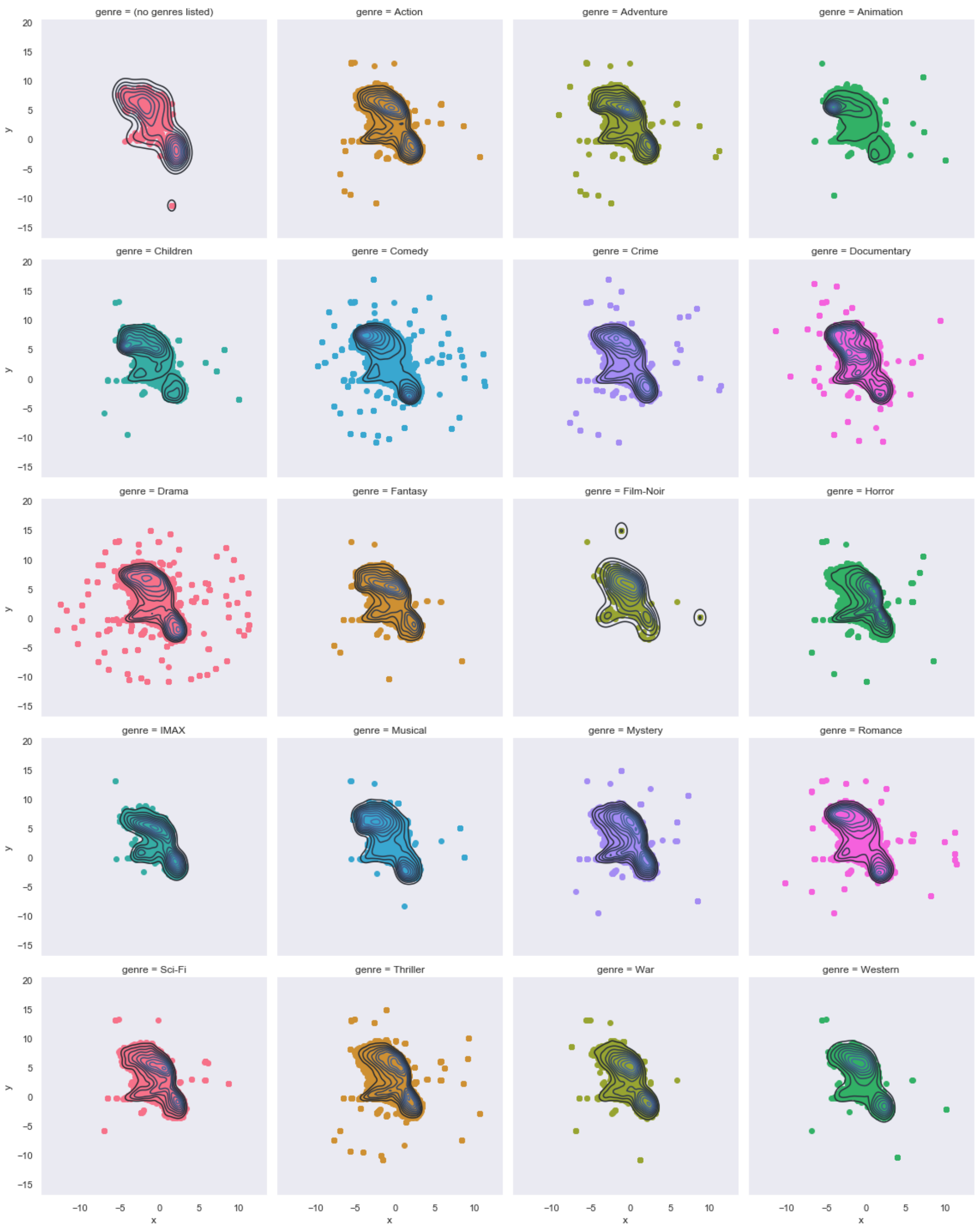


Figure 25: Visual feature distribution by genre in MovieLens AV dataset after dimensionality reduction with UMAP.

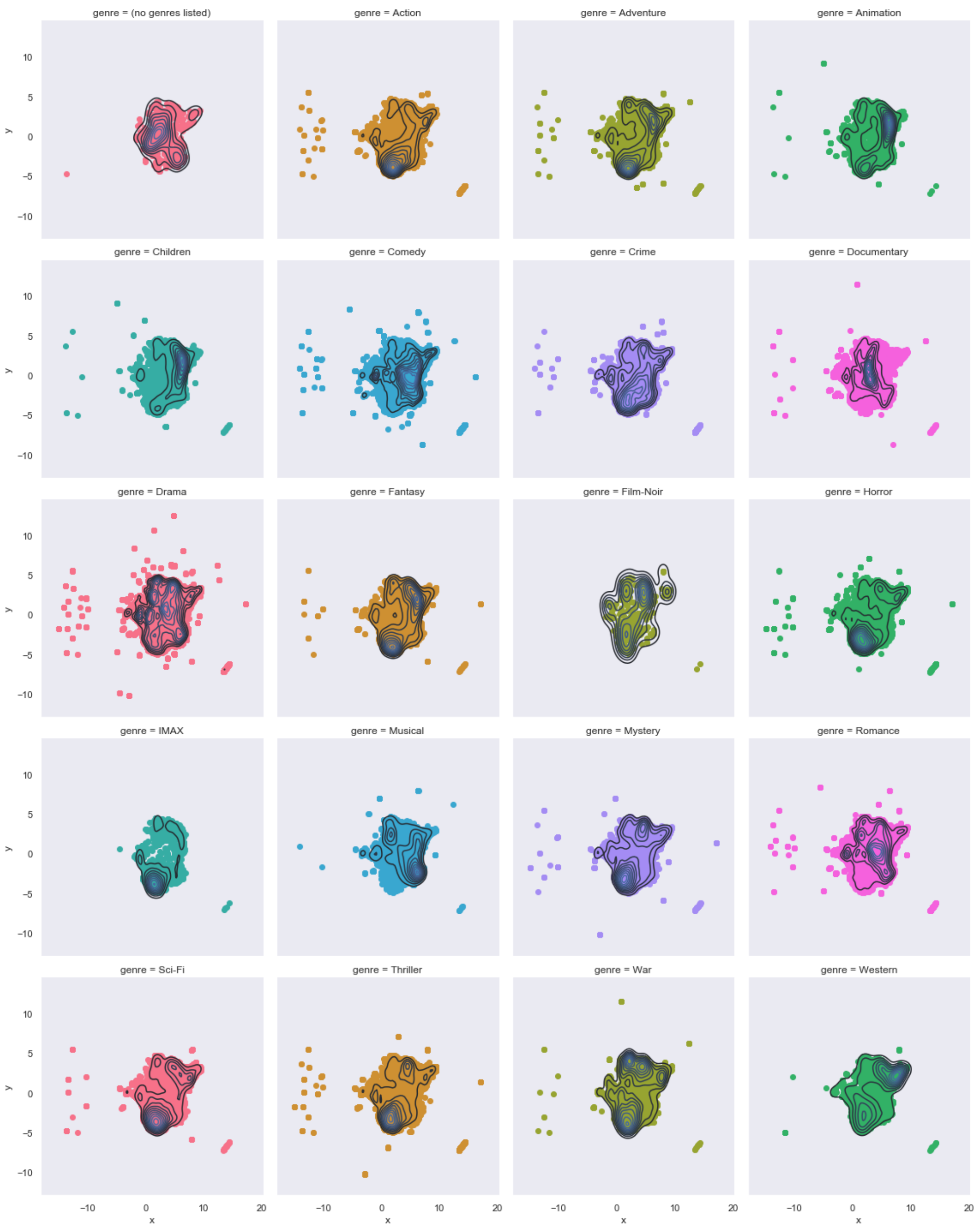


Figure 26: Audio feature distribution by sub clip number in MovieLens AV dataset after dimensionality reduction with UMAP.

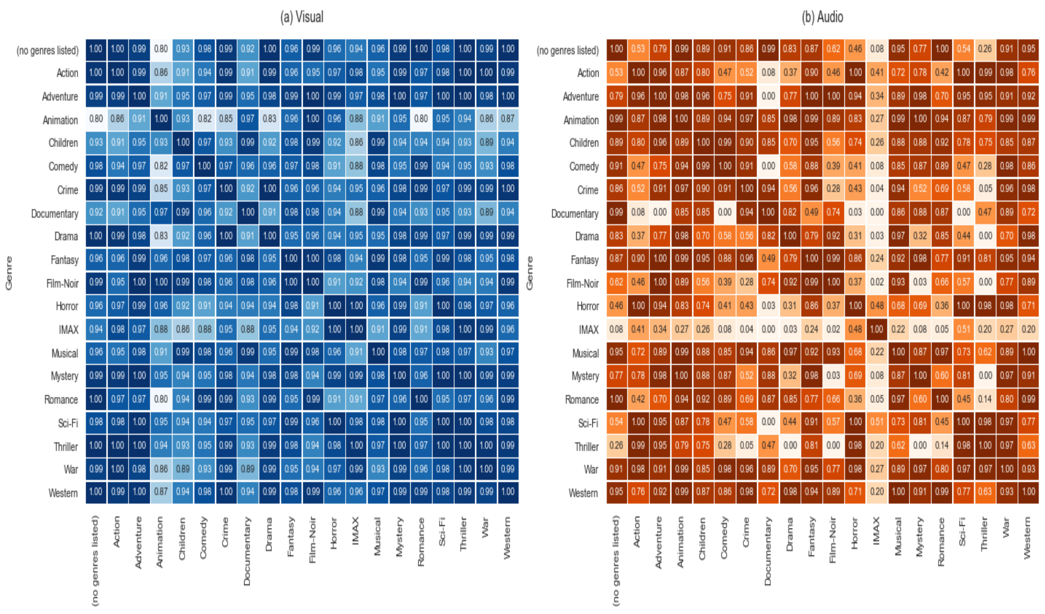


Figure 27: Bhattacharyya coefficients between genre distributions for (a) visual and (b) audio features. Bhattacharyya coefficients are calculated on the latent space representation of the features and on the inner tercile of each sample.

Thus, our conclusions from this section of the work are as follows

- Our proposed feature extraction process is clearly extracting some semantics given the tendency of the data to form clusters
- Analysing the membership of these clusters over time and by genre suggests that these semantics have both temporal and contextual connotations, with notably dense clustering at the beginning and end of trailers for both audio and visual streams, and dense clustering for the *Animation* genre for visual modality
- We defer discussion of whether or not these semantics are *meaningful* to the following subsection, with the rationale that if they correlate with a user's tendency to like or dislike a movie then they must have meaning in the domain of recommender systems

4.3 Audiovisual content enhanced collaborative filtering

4.3.1 Experimental approach

In this section we summarise the experimental setup we devised to evaluate our audiovisual enhanced collaborative filtering models before discussing the results pro-

distributions - and so we would see a slightly more pronounced set of coefficients if a nonparametric distance measure was used but expect the trend to be largely similar.

duced by them at length in the following section.

Labelling of models and choice of baselines

Critical to any investigation into the capabilities of a new model is to compare the results they produce against well-known baselines. As such we select the following models as our baselines

- **Random**: this model draws random samples from a normal distribution defined by the training data's mean and standard deviation
- **Baseline**: this model makes rating predictions by taking the global mean for the training data and adjusting each rating prediction by the user and item bias (see Equation 3)
- **User KNN (Z)**: is an implementation of the user-based *kth nearest neighbour* approach to collaborative filtering discussed in Section 2.1.2. We implemented both of Equations 1 and 2 in our experiments but found their results to be very similar. As such, we only display the results for Equation 2 in our results
- **Item KNN (Z)**: is an implementation of the item-based *kth nearest neighbour* version of Equation 2
- **Regularised SVD (RegSVD)**: is an implementation of the linear matrix factorisation presented in Equation 6
- **Deep Collaborative Filtering (DeepCf)**: is our own implementation of the neural collaborative filtering model summarised by Equation 13. As our work extends this model with audiovisual content information it is important that we know how our results compare to this model without the enhancements
- **Genre content (GenreCont)**: finally, we also introduce a content-based recommender system as a baseline given our model is a hybrid collaborative-content system. This model is an implementation of Equation 15 using genre features as content. While our success criteria for this work do not require us to outperform other content based models we include it to ensure that our results are within a similar range to those produce by other content-based models

The audiovisual content enhanced models we use are those described in Section 3. For clarity: **DCfAvCont(1)** refers to model (1); **DCfAvCont(2)** refers to model (2); and **LSTMAvCont** refers to model (3).

We implement all of our models in the opensource machine learning library *Keras* with the exception of the CF KNN baselines which use the *Surprise* library[37] and the GenreCont model which we implement in native *Python*.

Model training

To train our models we must tune a lot of hyperparameters. Specifically, for the audiovisual content enhanced collaborative filtering models these consist of

- **Network training hyperparameters**
  - *The optimiser*: we use the ADAM optimiser[30] for all standard training runs which outperforms other optimisers on most well-known learning tasks and it maintains an adaptive learning rate per parameter which thus removes the need to tune the learning rate
  - *The learning rate*: as we are using ADAM we commence all training runs with a learning rate of 0.001
  - *The number of epochs*: we implement *early stopping* in our models - where training is stopped if the validation loss starts increasing for a certain number of epochs - and thus set the number of epochs to a large number so that all models train to convergence with the global (or a local) optimum
  - *The batch size*: we do not tune the batch size. Instead we choose a batch size of 128 for all of our experiments as this provides a nice tradeoff between fast network training times without too much GPU memory usage
  - *The weight initialiser*: all of our network weights are initialised with the Keras default weight initialisation settings
- **Network architecture hyperparameters**:
  - The number of dense hidden layers
  - The number of neurons per layer
  - The number of LSTM layers (and number of LSTM units within these)
  - Where the network layers should be joined
  - The level of regularisation to apply and where and how to apply it (both L2 regularisation and dropout are considered)

To find the optimal network hyperparameters we employ a *grid search* technique whereby the model is run and evaluated on all combinations of hyperparameters within a pre-specified range. We considered using *Bayesian optimisation* techniques[38] but did not feel they were necessary given the programming overhead they introduce and the fact that our research goals do not necessitate that we find the absolute best hyperparameter combinations.
We also used *Tensorboard* throughout the training process which greatly aids the debugging of neural network training issues. For example, it creates interactive loss curves that allow for the easy comparison of training behaviour between runs and weight histograms that allow you to audit model parameter values over time to check they are updating as expected.

Table 4: Accuracy metrics in non and hard-cold start scenario

	Item Non Cold Start			**Item Hard Cold Start**		
	MAE	RMSE	MSE	MAE	RMSE	MSE
DCfAvCont(1)	**0.632**	**0.818**	**0.668**	0.747	0.952	0.906
DCfAvCont(2)	0.632	0.820	0.672	0.746	0.949	0.900
LSTMAvCont	0.719	0.917	0.842	**0.744**	**0.942**	**0.888**
DeepCf	0.636	0.820	0.673	0.751	0.947	0.897
RegSVD	0.674	0.856	0.732	0.801	0.988	0.977
User KNN (Z)	0.638	0.830	0.689	0.806	1.001	1.003
Item KNN (Z)	0.641	0.830	0.689	0.806	1.001	1.003
GenreCont	0.763	0.982	0.965	0.767	0.987	0.974
Baseline	0.641	0.825	0.680	0.753	0.948	0.898
Random	1.110	1.386	1.922	1.109	1.386	1.920

4.3.2 Results

As discussed in Section 2.1.6, it is important to analyse recommender systems with a range of evaluation metrics as different properties of the system may be desirable depending on the use case. In this section we assess our system and the baselines against all of the evaluation metrics outline in Section 2.1.6.

NB:

- All results in Tables 4-10 are presented on the same subsample of the MovieLens AV dataset containing 1000 movies and 1500 users with 40 ratings per user
- All results are those achieved from the optimal hyperparameters found through the grid search approach discussed in the previous sub-section
- All results were tested for replicability on this dataset by rerunning the models from different random seeds. Encouragingly the same trends were found and the metric values were of very similar magnitude
- Finally, all of the trends in the results were found to scale with the size of the dataset (specifically on the top 400 movie 600 user dataset and the top 4000 movie 6000 user dataset) as well as with the number of items sampled per user (please see Appendix B where these results are listed).

Accuracy

Table 4 presents the accuracy measures that assess how effective the system is at predicting explicit ratings for each user-item combination. The results produced by our proposed audiovisual enhanced models are encouraging.

In the *item non cold start* scenario our audiovisual enhanced deep collaborative filtering models outperform all baselines, including the “state of the art” deep collaborative filtering model without audiovisual information. However, this difference versus the deep collaborative filtering model is only slight and does vary with the number of starting items per user (see Appendix B for further results). Thus, we see it as a promising but not decisive finding.

In the *item hard cold start* scenario the audiovisual enhanced models - this time also including the LSTM model with a user content embedding but no collaborative filtering module - outperform all baselines comprehensively.

There are several trends worthy of note here

- **Performance of audiovisual enhanced and baseline collaborative filtering models between scenarios**:
  - Overall there is an unsurprising worsening in the performance of all the collaborative filtering models in the transition from the non-cold start to the hard-cold start scenario. While our audiovisual content enhanced models continue to outperform the baselines in the hard cold start scenario, they still suffer a notable reduction from this scenario transition.
  - However, it can be seen that their relative reduction in performance versus the collaborative filtering baselines of RegSVD, UserKNN and ItemKNN is less, with these baseline models showing a drop off in MAE of 0.127, 0.168, and 0.165, respectively versus 0.115 and 0.114 for our proposed DCfAvCont(1) and DCfAvCont(2) models.
  - The exception to this trend is the DeepCf baseline which also only shows a drop off in MAE of 0.115 between scenarios. Thus, it is possible that the “insulation” towards item cold start scenarios - as far as the accuracy related metrics go - results from the deep neural network architectures of these models more so than it does from the inclusion of audiovisual content.
  - This could perhaps be as a result of the fact that the deep architectures concatenate the user and item collaborative filtering embedding vectors and present them to the hidden layers of the network as a single unit (see Figures 10 and 6). Thus, the network learns relations to the user factors individually (as well as in combination with the item factors) and therefore it is still able to make use of the user relations it has learned from the training data in the *item cold start* scenario
  - However, it should be noted that the trends for the ranking metrics displayed in Tables 7 and 8 and discussed in the next subsection paint a different picture
- **Performance of audiovisual enhanced models versus genre content models**:

- The performance of our proposed models relative to the genre-based content baseline is a very promising finding of our work. We see that our proposed models achieve comparable (and indeed superior) results to the genre content baseline in both the *non cold* and *hard cold* scenario on the accuracy metrics
- To reflect on what this implies, it is most meaningful to compare the LSTMAvContent model with the GenreCont model as this version of our model isolates the effect of the audiovisual content (in other words it contains no collaborative filtering module). The fact that our model achieves comparable accuracy metrics - with a MAE score of 0.744 in the item cold start scenario versus 0.767 for GenreCoont - shows that using the audiovisual content from movie trailers - extracted and preprocessed by our feature extraction pipeline - is a viable form of content to enhance recommender systems.
- This finding thus validates one of our central research hypotheses that summarised audiovisual content has a part to play in improving the capabilities of recommender systems.
- Moreover, linking this back to Section 4.2 while it is not absolute proof that the features we have extracted are the most *semantically meaningful*, it does prove that they are at least somewhat *semantically meaningful* as otherwise they would not match the performance of clearly *semantically meaningful* features in the form of genre tags that have been hand-coded by a human
- With that said, we should reiterate that the goal of this research is not to outperform recommender systems based on other forms of content (as time does not permit for this), but rather to propose a framework that incorporates audiovisual information with collaborative filtering models and alleviates their *item cold start* issues as a result
- It is also worth quickly noting the comparatively weak performance of both LSTMAvCont and GenreCont in the *non cold start* scenario versus the collaborative filtering models, which reinforces the literature by showing that collaborative filtering models produce the best results when item review data is plentiful

• **Performance of audiovisual enhanced CF models versus one and other**:

- We also briefly note that the different versions of our models perform approximately equally with one another in the *hard cold start* scenario, with a slight superiority for LSTMAvContent
- However, in the *non cold start* scenario the models with collaborative filtering modules are significantly better, suggest the dominance of this technique and their ability to exploit it in this setting

Finally, to conclude this subsection we make a quick comment about the nuances between the accuracy measures presented (more information on them can be found in Section 2.1.6).

The main point to comment on is the difference between the RMSE and MAE metric. The RMSE sums the square of the errors and thus is more affected by outliers than the MAE metric. Thus, while the RMSE does have benefits in applications where robustness of predictions is very important, it is also not a true reflection of the average error as it amplifies the impact of outliers. Therefore, we focus our conclusions on the MAE as it is more representative of the true error in our system.

Nevertheless, both the RMSE and MAE are presented given their prevalence in the literature. The MSE - which is the loss function used to train the model - is also presented for completeness.

Ranking metrics - non cold start scenario

Table 5: Rank metrics in non-cold start scenario (k=4)

	Item Non Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	0.704	**0.681**	**0.692**	0.813	**0.620**	0.635
DCfAvCont(2)	0.703	0.625	0.662	0.802	0.588	0.665
LSTMAvCont	0.406	0.367	0.386	0.469	0.335	0.517
DeepCf	**0.718**	0.650	0.682	**0.822**	0.608	**0.667**
RegSVD	0.660	0.514	0.578	0.724	0.489	0.667
User KNN (Z)	0.682	0.598	0.637	0.767	0.561	0.660
Item KNN (Z)	0.695	0.617	0.654	0.794	0.579	0.659
GenreCont	0.533	0.475	0.503	0.629	0.415	0.481

Table 6: Rank metrics in non-cold start scenario (k=10)

	Item Non Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	0.680	**0.683**	**0.681**	0.813	**0.597**	0.735
DCfAvCont(2)	0.677	0.591	0.631	0.797	0.535	0.753
LSTMAvCont	0.394	0.393	0.394	0.469	0.336	0.656
DeepCf	**0.694**	0.614	0.652	**0.822**	0.558	**0.757**
RegSVD	0.651	0.433	0.520	0.725	0.405	0.756
User KNN (Z)	0.662	0.555	0.604	0.767	0.506	0.753
Item KNN (Z)	0.677	0.568	0.618	0.794	0.519	0.752
GenreCont	0.526	0.462	0.492	0.630	0.385	0.634

Tables 5 and 6 shows the ranking results of the different models at $k = 4$ and $k = 10$

(this time with the most basic baselines removed for brevity). The variable k refers to the length of the recommended list that we consider when calculating the ranking metric. So, for example, $k = 4$ implies that the first 4 items in the recommended list are considered[32].

Regarding the *non-cold start scenario*, the following trends are worthy of discussion

- Unsurprisingly, the collaborative filtering models achieve far superior results to the content models on all of the rank metrics in this scenario. More encouraging still is the fact that the neural collaborative filtering models (including those with audiovisual content) achieve the best results - with the DeepCf model and the DCfAvCont(1) claiming the top spot in the metrics (with the DCfAvCont(2) model closely behind)
- The most noteworthy trend from this scenario for these metrics is the tradeoff in precision and recall capabilities between the models[33]. The audiovisual enhanced deep collaborative filtering models appear to gain recall at the expense of some precision.
- For example, for the metrics "@4" DCfAvCont(1) enjoys a higher recall (R_{adj}@4) of 0.681 versus 0.650 in the DeepCf model (an increase of 5%), but in doing so loses 2% in its precision (P@4) to rest at 0.704 versus 0.718 for the DeepCf model. This trend is echoed in the "@10" metrics and the relative tradeoffs are best summarised by the F1 metrics, which go in favour of the DCfAvCont(1) model
- Thus, we conclude that the introduction of audiovisual content boosts the recall capabilities of the model while almost maintaining its precision capabilities. This insight goes along way to explaining the MRR, MAP and NDCG metrics listed in Tables 5 and 6
- **For MRR** - which represents the model's capacity to place a relevant item highest up the recommendation list - we see that DeepCf marginally outperforms DCfAvCont(1)[34]
- By contrast, **for MAP** - which rewards the model's capacity to fill the recommended list with relevant items (i.e. all relevant items are rewarded rather

[32] For reference, a detailed definition and discussion of these metrics is provided in Section 2.1.6. We also provide a "toy example" of them being calculated in Appendix E.

[33] To recap, in the context of recommender systems: precision refers to the ability of the model to only recommend items that are actually *relevant* to the user in the ground truth data (thus minimising false positives), whereas recall refers to the ability of the model to ensure all ground truth *relevant* items are recommended to the user (thus maximising true positives). Please refer to Section 2.1.6 for further discussion.

[34] On the same test dataset, a model with more precision and less recall is likely *on average* to be generating shorter recommendation lists to users. A claim that is supported by the average recommendation list results in Table 9, where the DCfAvCont(1) model recommends on average 5.0 items to users whereas the DeepCf model recommends 4.4. Thus, on average we should expect the first relevant item to be at a higher rank (i.e. earlier position) in the list - hence the trend we observe in MRR

than just the first) - shows DCfAvCont(1) performing strongest. Referencing this back to the trend we observe in recall, it makes sense that a system which is recalling more of the ground truth *relevant* items should score higher on this metric

- Finally, **for NDCG** - which is similar to MAP in that it rewards the system for recommended relevant items but differs in the sense that it assigns a non-binary relevance weighting - we see that DeepCf narrowly outperforms DCfAvCont(2) (and more significantly so DCfAvCont(1)).
- This reversal in trend can potentially be explained by recognising that the NDCG metric rewards the system in a slightly different way than MAP. Where MAP has a tendency to reward systems highest that recall the most relevant items, the NDCG has the tendency to reward systems highest that place the relevant items that the user has ranked highest in the ground truth data at the top spots in the ranking. Thus, it is indeed possible for one algorithm to score higher on MAP and the other on NDCG (please reference Appendix E for a "toy" example of this phenomenon).

Ranking metrics - hard cold start scenario

Tables 5 and 6 present the ranking metrics for the models in the *hard cold start scenario*. Here the results of the audiovisual enhanced collaborative filtering models versus the baseline collaborative filtering models are very encouraging.

To summarise the key trends

- **Performance of audiovisual enhanced CF models versus CF baselines**:
  - We see that the audiovisual enhanced models outperform the DeepCf model on all metrics and by a significant amount. This suggests that the audiovisual enhanced models are making more relevant *cold start* item recommendations to existing users
  - This finding thus validates another of our central research hypotheses that the relative benefit of enhancing collaborative filtering models with audiovisual content is increased in the item cold start scenario
  - We posit a likely explanation for this in our Discussion (please see Section 5.1)
- **Performance of audiovisual enhanced CF models versus GenreCont model**:
  - However, the very good results of the GenreCont model on all metrics versus the DCfAvCont and LSTMAvCont models - with the exception of the NDCG metric - should be noted
  - Again, as this is a significant finding of our work we propose a likely explanation for it in our Discussion (please see Section 5.2)

Table 7: Rank metrics in hard-cold start scenario (k=4)

	Item Hard Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	0.295	**0.285**	0.290	0.338	**0.255**	**0.512**
DCfAvCont(2)	0.236	0.247	0.241	0.280	0.214	0.433
LSTMAvCont	**0.312**	0.277	**0.294**	**0.353**	0.249	0.508
DeepCf	0.207	0.215	0.211	0.240	0.185	0.417
RegSVD	0.000	0.000	0.000	0.000	0.000	0.424
User KNN (Z)	0.000	0.000	0.000	0.000	0.000	0.424
Item KNN (Z)	0.000	0.000	0.000	0.000	0.000	0.424
GenreCont	**0.515**	**0.464**	**0.488**	**0.612**	**0.404**	0.481

Table 8: Rank metrics in hard-cold start scenario (k=10)

	Item Hard Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	0.278	**0.327**	**0.301**	0.339	**0.271**	**0.654**
DCfAvCont(2)	0.235	0.320	0.271	0.281	0.251	0.600
LSTMAvCont	**0.302**	0.300	**0.301**	**0.353**	0.253	0.650
DeepCf	0.208	0.280	0.239	0.241	0.219	0.590
RegSVD	0.000	0.000	0.000	0.000	0.000	0.595
User KNN (Z)	0.000	0.000	0.000	0.000	0.000	0.594
Item KNN (Z)	0.000	0.000	0.000	0.000	0.000	0.594
GenreCont	**0.511**	**0.457**	**0.482**	**0.613**	**0.377**	0.633

- We also note here that the exception in the ranking metric results is that our proposed models do still beat the GenreCont model on the NDCG metric which is an important metric and means that they are placing the items that the user likes most in the ground truth data at the higher rankings in the recommendation list versus GenreCont
- A possible explanation for this reversal in trend can be linked back to the fact the DCfAvCont models also outperform the GenreCont model on accuracy metrics, as a system that more accurately predicts individual item ratings is more likely to then prioritise them appropriately

• **Performance of audiovisual enhanced CF models versus one and other**:

  - We also briefly comment on continued strong performance of DCfAvCont(1) and LSTMAvCont relative to one and other on the ranking metrics in this scenario. It is interesting that DCfAvCont(1) has higher recall capabilities but slightly lower precision than LSTMAvCont
  - DCfAvCont(2) also underperforms considerably compared to our other models on these metrics in this setting. As it has more network parameters (as a result of possessing two user embedding modules) it is potentially overfitting the data and thus underperforming on the test set

• **Performance of non-neural CF baselines**:

  - Finally, the explanation for the zero values for RegSVD and both KNN models should be briefly noted. As the test items have no review data these models operate in a very limited capacity.
  - For the RegSVD model, this means that the item *latent factor* vector is still in a random initialisation state centered around 0. Thus, the inner product term in Equation 6 evaluates to close to 0 with all users (as no strong *affinity* scores are generated on any of the vectors' elements). Therefore, the RegSVD model always predicts the global mean with a slight adjustment for the user bias in this scenario (which always evaluates to less than 3.5 for this particular instance of the data - therefore, nothing is recommended)
  - For the KNN model, the explanation is simpler. The similarity term of Equation 2 always evaluates to 0 as it is based off of a comparison of the test item's rating history to other similar items. As the test item has no test history there are no similar items. Therefore, the KNN model just predicts the global mean for each item and thus nothing is recommended
  - As aforementioned, the neural CF baseline does not fall foul of this fate as it can fall back on making predictions based solely off of the variance in the user's CF *latent factor* vector

Other metrics

Table 9: Other metrics in non-cold start scenario

	Item Non Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	0.921	**0.629**	0.981	**5.002**	**5.003**
DCfAvCont(2)	0.896	0.561	0.982	4.101	**5.003**
LSTMAvCont	**0.980**	0.349	**0.993**	3.193	**5.003**
DeepCf	0.921	0.583	0.982	4.357	**5.003**
RegSVD	0.759	0.471	0.983	3.021	**5.003**
User KNN (Z)	0.921	0.538	0.985	4.014	**5.003**
Item KNN (Z)	0.881	0.565	0.984	4.054	**5.003**
GenreCont	**0.999**	0.457	0.939	3.907	**5.003**

Table 10: Other metrics in hard-cold start scenario

	Item Hard Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	**0.999**	**0.268**	0.997	**2.840**	**5.003**
DCfAvCont(2)	0.918	0.246	0.996	2.819	**5.003**
LSTMAvCont	0.993	0.256	0.997	2.565	**5.003**
DeepCf	0.998	0.213	**0.998**	2.463	**5.003**
RegSVD	0.000	0.000	0.000	0.000	**5.003**
User KNN (Z)	0.000	0.000	0.000	0.000	**5.003**
Item KNN (Z)	0.000	0.000	0.000	0.000	**5.003**
GenreCont	**0.999**	**0.440**	0.941	**3.851**	**5.003**

As discussed in Section 2.1.6, it is important to assess recommender systems on characteristics beyond their ability to just recommend relevant items. As such we implemented metrics for coverage, diversity and personalisation and present the results of these in Tables 9 and 10.

- The stand out results from the other metrics are the improved diversity score for DCfAvCont(1) versus DeepCf in both the *non cold* and *hard cold* settings. Increased diversity in the recommended list mitigates against the risk that the user ends up disliking the entirety of the list because it is made up of a single content type that they dislike[35].

- All neural based models achieve good catalogue coverage and personalisation results, and it is interesting that they score more highly on these metrics in the *hard cold start* scenario - perhaps an outcome of the model being less sure about what to recommend for each user

- Finally, the GenreCont model once again records a strong performance in the *hard cold start* scenario and this is likely again due to its nonparametric nature (as discussed in the previous section). What strengthens that claim here is the observation that it generates significantly more recommendations in the *hard cold start* scenario versus the other models - with 3.851 on average versus 2.840 for the nearest deep model.

Effect of network architecture and pretraining on audiovisual enhanced collaborative filtering models

In this section, we briefly comment on trends in optimal network architectures that we observed when running hyperparameter optimisation to produce the results presented in the previous section.

- Generally speaking, we saw a trend towards smaller network architectures. For example, working with less than 64 user and item CF factors, and dense connected layers consisting of between 8 and 64 neurons with no more than 1-3 layers per module (CF, AV or join) in our model. This is in line with the observations from [23]

- We also found that an LSTM consisting of two LSTM layers for each modality was optimal. It appeared from our work that only one layer does not give the network enough capacity to learn whereas more than two layers causing the network to overfit drastically. All of our results were thus produced with LSTM layers of dimension [128, 64] for each audiovisual content modality

[35]It is worth caveating that the diversity metric has been calculated by computing the variety in the genre tags applied to the items in the recommendation list. Thus, while they are less homogeneous from a high-level genre perspective with the av-enhanced models this does not mean they are less homogeneous from a low-level audiovisual perspective (and indeed it is very likely that the opposite is true). Whether or not this is desirable depends entirely on the context and specific user.

- Finally, it is also important to note that we observed a trend for smaller network architectures to perform better on the accuracy metrics, whereas slightly larger networks performed better on the ranking metrics

Additionally, it is relevant to note that we attempted *pretraining / transfer learning* with our models (in line with our vision for these models laid out in Section 3.3). However, at the time of writing, this has not yet yielded a significant improvement in the results.

For completeness, our approach to pretraining was to first train the DeepCf and LSTMAvContent models with their optimal hyperparameters on a training set of the data and save their weights. We then instantiated an instance of the DCfAvCont(2) model with the same network architecture in its CF and AV modules as the DeepCF and LSTMAVContent models, respectively. The weights from these models were then transferred to the DCfAvCont(2) model and this model was then trained further on the same training set.

Pretraining is well-known to be a delicate art[8] yet from related literature [23] we remain confident that it will improve the accuracy of our models.

Effect of different feature aggregation techniques on audiovisual enhanced collaborative filtering models

We also briefly comment on the effect of different feature aggregation techniques. Table 26 of Appendix C shows the effect on the ranking scores of the different aggregation techniques we implemented. All the features produce comparable results, suggesting that they are all equally viable techniques of summarising the data and can be used interchangeably.

Scalability of our proposed models

Finally, we also benchmarked the training and prediction times of the different recommender system models and Table 11 presents the results from this.

Of note here is the relatively long training times of our proposed audiovisual models versus the collaborative filtering baselines. This can be attributed to the addition of the LSTM which has a lot of model parameters and thus takes longer to train (as there are more weight update operations to carry out in the backpropagation algorithm). It should also be noted that the LSTM can introduce memory issues if the number of parameters is too high but this can be mitigated by reducing the batch size (however, doing so in turn will increase training times).

Also of note is the long prediction times for the GenreCont model. This is because the model is nonparametric and thus it only commits to an approximation of the target function at query time. Moreover, the complexity of the algorithm is $O(N^2)$ for each user (therefore $O(N^3)$ overall) as it must compute the nearest train item

Table 11: Model training and prediction times for 1000 movie 1500 user MovieLens AV dataset (non-cold start scenario; 40 items per user). The neural network training times are over 100 epochs with a batch size of 128.
NB: the training times include the time taken to train the model on both folds of the data (and are therefore a fair estimate of how long it takes to train the model on all of the rating data).

	Training time	**Prediction time**
DCfAvCont(1)	900s	10s
DCfAvCont(2)	900s	10s
LSTMAvCont	800s	10s
DeepCf	150s	5s
RegSVD	80s	2s
User KNN	0s	60s
Item KNN	0s	60s
GenreCont	0s	1320s

neighbours for *each* item in the test set. Therefore, its run time is proportional to $|U| \times |I_{train}| \times |I_{test}|$. The other algorithms scale linearly with the size of the training and test data if the same network architecture is used.

For clarity, the timings in Table 11 correspond to models with the following architectures / parameters

- **DCfAvCont(1)**: 64 factor user and item collaborative filtering embeddings; 2 deep collaborative filtering layers with [32, 16] neurons; an individual LSTM component for visual and audio modalities both with [128, 64] LSTM units and dense layers of [32, 16] to merge the modalities; finally, 1 dense layers of [8] to join the collaborative filtering and content branches

- **DCfAvCont(2)**: the same architecture as DCfAvCont(1) except for the inclusion of a user content embedding unit of 64 factors

- **LSTMAvCont**: the same architecture as DCfAvCont(2) but without the collaborative filtering branch

- **DeepCf**: 64 factor user and item collaborative filtering embeddings and dense layers of [32, 16, 8]

- **RegSVD**: 64 factor user and item collaborative filtering embeddings

- **User KNN**: 20 nearest user neighbours

- **Item KNN**: 20 nearest item neighbours

- **GenreCont**: 5 nearest item neighbours (for each item in the test set)

All results were generated on a laptop with an Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz and 8 CPUs with 32GB of RAM and a Nvidia GeForce GTX 980M GPU. The models implemented in Keras (neural models + RegSVD) ran on the GPU and the GenreCont calculation was multiprocessed across the 8 available cores.

5 Discussion

In this section we summarise our findings from the various experimental work we conducted and align them with our proposed approach (see Section 3) and the relevant theory.

5.1 Performance of audiovisual enhanced collaborative filtering models versus collaborative filtering baselines

We find that our audiovisual enhanced collaborative filtering models significantly outperform the collaborative filtering baselines in the *item hard cold start* scenario on all metrics[36]. We also find that they outperform all CF baselines in the *item non cold start* scenario except for the state of the art *deep collaborative filtering* model on some of the ranking metrics.

Addressing these observations in turn, our models' superior performance in the *item hard cold start* scenario is likely to be as a result of the audiovisual content module they contain.

This gives them a predictive capability to fallback on when the collaborative filtering module has no information on the new item. By analysing the audiovisual content of the new item - which is present even when the item has no reviews - the models are able to make a prediction based on the extent to which the target user has enjoyed movies with similar audiovisual content in the past.

This conclusion is strongly supported by the performance of our "audiovisual content only" model (LSTMAvCont) which typically performs as well as the audiovisual enhanced collaborative filtering models in the *item hard cold start* scenario.

Regarding the *item non cold start* results, when compared to the *deep collaborative filtering* model, we see that our audiovisual content module endows the system with more recall at the expense of some precision but improves on the overall F1 measure.

The result of this enhanced recall ability is that our audiovisual enhanced models achieve higher mAP scores versus the *deep collaborative filtering* model at the expense of lower mRR scores.

Whether or not a larger mRR or larger mAP is desirable in recommender systems (if a tradeoff must be made) depends on the context. If the screen real-estate is small or the user has a short attention span then mRR might be more desirable as it rewards the system for ensuring a relevant item is at the highest rank possible. Whereas, if the user is more likely to browse through a list of recommendations - which seems

[36]With the exception of only a slight but non-negligible increase on the accuracy metrics versus the *deep collaborative filtering* model.

viable in the context of movie recommendations - then mAP may be more desirable.

On the other metrics for the *item non cold start* scenario, the performance between our models and the *deep collaborative filtering* model are similar which suggests there is still work to do in optimising our model architecture (see Section 5.3).

To round out the analysis with a perceivable downside of our models, they do take a significantly longer time to train than the *deep collaborative filtering* model. This is understandable given the introduction of the audiovisual content and the LSTM with the result that the network has far more network parameters. A potential mitigation strategy for this is to train the model in a distributed computing environment where it can access more GPUs and process data on distributed RAM.

5.2 Performance of audiovisual enhanced collaborative filtering models versus the genre baseline

We see very promising results for our models relative to the genre KNN baseline. Our models comprehensively outperform this model in the *item non cold start* scenario and show good (and sometimes superior) results to it in the *item hard cold start* scenario.

The superior performance of our models with collaborative filtering modules in the *item non cold start* scenario is not surprising. Collaborative filtering is a superior technique when rating data on the items exists and our model architecture exploits this.

The comparable results in the *item hard cold start* are an excellent result in many ways. This shows that from a completely unsupervised feature extraction technique - which is able to process over 650 hours of video in less than 24 hours - we are able to subsequently design a recommender system that uses these features to address the item hard cold start issue with comparable performance to a model that uses features that must be hand-coded by humans.

That said, our model is still significantly behind several of the metrics versus the genre baseline which suggest there is much room for improvement. We posit that why it drops behind at the moment is as a result of two factors.

The first is straightforward: while we have proven our extracted content features are meaningful in this domain, there are a multitude of ways in which they can be enhanced to contain more *semantically meaningful* information. See Sections 5.4 and 6.2.

The second requires some more explanation. Referencing back to the counts of rating by rating value (see Figure 14), the fact that the distribution of the ground truth data used to train the model is approximately Gaussian with its central tendency around 3.5 and a reasonably small standard deviation has a considerable implication on the way the models learn.

The genre-based content model is based on KNN and is thus nonparametric whereas our audiovisual enhanced deep collaborative filtering models are parametric. Therefore, our models must commit to a global approximation of the target function at training time whereas the genre KNN model can calculate a local approximation of the target function for each user at query time.

Combining these observations we can explain why the predictions from our models are more centered around the sample mean with far less "extreme value" predictions (such as 5.0 and below 1.0) versus those of the genre content model. This in turn means less rating predictions are above the threshold required to make a recommendation and hence why our models generate shorter recommendation lists on average and underperform on the rank metrics (but, interestingly, not on the accuracy metrics).

A potential solution to this problem would be to reformulate the loss function of the deep collaborative filtering models to explicitly reward ranking performance - a discussion which we save for Section 6.2. Similarly, one could apply an upsampling strategy to the training data, although this is not trivial given explicit rating prediction is a regression problem and upsampling must be done in a way that preserves the similarity trends between user and items in the rating matrix.

5.3 Performance of audiovisual enhanced collaborative filtering models versus one another

Our 3 proposed model architectures were outlined in Section 3.3. From the results, it would appear that "Deep Collaborative Filtering with AV Content Model 1" is the best performer overall as it keeps track with the state of the art Deep Collaborative Filtering model in the *item non cold start* scenario and significantly outperforms it in the *item hard cold start* scenario.

In Section 5.1 we argue why this model has these abilities; in this section we provide brief comment on why we suppose this model outperforms its audiovisually enhanced peers.

Regarding its performance versus LSTMAvCont, it is most probable that the additional *fully connected* layers after its collaborative filtering and audiovisual content branches join allow it to combine this information in an additive way and thus enhance its predictive capacity. Hence, it has learned how to make very good predictions in the *item non cold start* scenario - where collaborative filtering advice is better - but also knows to leverage its content capabilties when it is queried on an unseen item.

Regarding the underperformance of “Deep Collaborative Filtering with AV Content Model 2” versus Model 1, we propose that as a result of its additional user embedding module it is too complex in its current shape to learn as effectively.

We designed this version of the model so that we could *pretrain* its components and we attempted this in our experiments. However, at the time of writing, this pretraining has not improved the results of “Deep Collaborative Filtering with AV Content Model 2”. We believe this is simply because we have not yet discovered the optimal pretrain recipe and remain confident that this approach will yield improvements in future work.

5.4 Utility of audiovisual content feature descriptors in collaborative filtering models

Our strong experimental results using audiovisual content features in our models suggest that this is a viable form of content to enhance recommender systems with. As mentioned in Section 5.2, we see the fact that we compare favourably with results from a content based baseline as proof of this.

As a result, we feel that we have met our success criterion for this part of the work in the sense that this means our feature extraction process produces features that are *semantically meaningful* in the domain of recommender systems. However, we do not quantify this extent and thus we do not formally address the *semantic gap* we discussed in Section 2.2.4. Doing so would be an interesting item of future work.

6 Conclusions and future work

6.1 Conclusions

In this section we pin our findings back to the research questions we posed at the beginning of the project

- **RQ1.a** *Does the audiovisual content of movie trailers enhance the capabilities of recommender systems in the item warm start scenario?*
  - To a certain extent it does, most notably by endowing collaborative filtering algorithms with greater recall and as a result higher mAP scores while still matching the rating prediction accuracy scores of a state of the art deep collaborative filtering baseline
  - However, there is certainly scope for improvement in warm-start scenario results and these may be achievable in the near term through further assessment of network architectures and transfer learning / pretraining strategies
- **RQ1.b** *Does the audiovisual content of movie trailers enhance the capabilities of recommender systems in the item cold start scenario?*
  - Given our audiovisual enhanced models show a significant improvement on almost all evaluation metrics versus the un-enhanced baselines we can conclude confidently that it does
  - Moreover, the fact that our model can perform like this while at the same time achieving state of the art accuracy results in the warm start scenario suggests the audiovisual content module we enhance it with is additive to the system's performance and makes it more robust to a broader range of commonly encountered rating scenarios
- **RQ2.** *How can audio-visual content be summarised to enable an investigation of its usefulness in recommender systems?*
  - Using pretrained neural networks to extract features from audiovisual content on a frame-by-frame basis and then aggregating these frames to a subclip level is a viable form of summarising this type of content
  - These features are proven to be *semantically meaningful* - at least to a certain extent - by the fact that our algorithms can use this information to improve their performance significantly in item cold start scenarios

6.2 Future work

Drawing together all of our findings from this research, we propose the following items of future work

- **Using different loss functions to train the collaborative filtering models**: the loss function of the model could be parameterised in a different way so as to encourage the system to learn slightly different trends from the training data. For example, we could adopt the ranking loss function of [22] that explicitly rewards the system for learning the correct rank of different items

- **Implicit rating capabilities and affective feedback data**: related to the above, we could also reformulate the final layer of our network to allow it to learn from and predict *implicit ratings*. An area we are very interested in assessing further is the use of *affective feedback* signals from users (such as their facial expressions) as the *implicit rating* data in our system. Being able to use this type of data in recommender sytems would greatly increase the amount of data that the system has to learn with

- **Active learning as a complementary solution to user cold start**: in our work we do not address *user cold start* scenarios. However, we see recent advances in the field of active learning[31, 48] that use reinforcement learning approaches to help models learn quickly on a small data labelling budget as being a way to potentially enhance our system with a way to address *user cold start*

- **Additional content feature modalities**: we built our system in a modular way so that additional content modalities could be added to it with minimal rework. As a next step we would like to consider using the script of the movie as a form of content. We expect that this will give the system a greater understanding of what a trailer contains and thus increase its recommendation performance. Moreover, it will allow us to further enable some of the use cases we outlined in Section 1

- **Different representations of video content and different aggregation methods**: while the audiovisual content feature descriptors we create in this work are sufficient to prove the viability of our approach we believe that they can be enhanced, in particular the way in which they are aggregated. Supervised feature extraction techniques that make use of the Gini Index are something we could consider next. Alternatively, we could devise a deep model that learns for itself what the most relevant portions of the trailer are to use in a summary of it

- **Privacy preserving recommender systems**: finally, we would like to look into the feasibility of modifying our system so that it can become *privacy preserving*. To do so would require an assessment of our system to operate within the constraints of the “three pillars of privacy preserving AI” - differential privacy, federated learning and multi-party computation. The recent opensource library *PySyft*[55] provides all the tools required to build a deep learning model in line with these concepts and we feel it would be an excellent feature of our model in a generation where users expect companies to be more considerate of their privacy rights

References

[1] R.W. Picard et al. A. Pentland. “Photobook: Content-Based Manipulation of Image Databases”. In: (1995).

[2] D.J. Watts A. Sharma J.M. Hofman. “Estimating the causal impact of recommendation systems from observational data”. In: (2015).

[3] Benjamin Schrauwen Aaron van den Oord Sander Dieleman. “Deep content-based music recommendation”. In: (2013).

[4] Charu C. Aggarwal. Recommender Systems: The Textbook. 2016.

[5] Alberto Garcia-Garcia et al. “A survey on deep learning techniques for image and video semantic segmentation”. In: (2017).

[6] Cai-Nicolas Ziegler et al. “Improving Recommendation Lists Through Topic Diversification”. In: (2005).

[7] Carreira et al. “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”. In: (2017).

[8] Chuanqi Tan et al. “A Survey on Deep Transfer Learning”. In: (2018).

[9] Diba et al. “Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification”. In: (2017).

[10] Dumitru Erhan et al. “Why Does Unsupervised Pre-training Help Deep Learning?” In: (2010).

[11] Fidel Cacheda et al. “Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems”. In: (2011).

[12] Francesco Ricci et al. “Recommender Systems Handbook”. In: (2010).

[13] Jeff Donahue et al. “Long-term Recurrent Convolutional Networks for Visual Recognition and Description”. In: (2014).

[14] Jort F. Gemmeke et al. “Audio Set: An ontology and human-labeled dataset for audio events”. In: (2017).

[15] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: (2014).

[16] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: (2015).

[17] Mohammed Senoussaoui et al. “An i-vector Extractor Suitable for Speaker Recognition with both Microphoneand Telephone Speech”. In: (2010).

[18] Ognjen Rudovic et al. “Personalized machine learning for robot perception ofaffect and engagement in autism therapy”. In: (2018).

[19] Qian Zhao et al. “Explicit or Implicit Feedback? Engagement or Satisfaction?” In: (2018).

[20] S Hochreiter et al. “Long short-term memory”. In: (1997).

[21] Shawn Hershey et al. “CNN Architectures for Large-Scale Audio Classification”. In: (2017).

[22] Steffen Rendle et al. “BPR: Bayesian Personalized Ranking from Implicit Feedback”. In: (2009).

[23] Xiangnan He et al. “Neural Collaborative Filtering”. In: (2017).

[24] Yashar Deldjoo et al. “MMTF-14K: A Multifaceted Movie Trailer Feature Dataset for Recommendation and Retrieval”. In: (2018).

[25] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. “ImageNet Classification with Deep ConvolutionalNeural Networks”. In: (2012).

[26] et al Andrea Frome. “DeViSE: A Deep Visual-Semantic Embedding Model”. In: (2013).

[27] Justin Basilico Ashok Chandrashekar Fernando Amat and Tony Jebara. “Artwork Personalization at Netflix”. In: (2017).

[28] J. et al Deng. “ImageNet: A Large-Scale Hierarchical Image Database”. In: (2009).

[29] MUKUND DESHPANDE and GEORGE KARYPIS. “Item-Based Top-N Recommendation Algorithms”. In: (2004).

[30] Jimmy Ba Diederik P. Kingma. “Adam: A Method for Stochastic Optimization”. In: (2014).

[31] Meng Fang, Yuan Li, and Trevor Cohn. “Learning how to Active Learn: A Deep Reinforcement Learning Approach”. In: (2017).

[32] Simon Funk. “Netflix Update: Try This at Home”. In: (2006).

[33] CARLOS A. GOMEZ-URIBE and Inc. NEIL HUNT Netflix. “The Netflix Recommender System: Algorithms, Business Value, and Innovation”. In: (2016).

[34] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: (2016).

[35] Anders Krogh John A. Hertz. “Simplifying Neural Networks by Soft Weight-Sharing”. In: (1991).

[36] Bamshad Mobasher Himan Abdollahpouri Robin Burke. “Managing Popularity Bias in Recommender Systems with Personalized Re-ranking”. In: (2019).

[37] Nicolas Hug. “Surprise, a Python library for recommender systems”. In: (2017).

[38] Hugo Larochelle Jasper Snoek and Ryan P. Adams. “Practical Bayesian Optimization of MachineLearning Algorithms”. In: (2012).

[39] Zheng Wang Jianlin Cheng and Gianluca Pollastri. “A Neural Network Approach to Ordinal Regression”. In: (2007).

[40] Guy Lebanon Joonseok Lee Mingxuan Sun. “A Comparative Study of Collaborative Filtering Algorithms”. In: (2012).

[41] Andrea Vedaldi Karel Lenc. “Understanding Image Representations by Measuring Their Equivariance and Equivalence”. In: (2018).

[42] Halbert White Kurt Hornik Maxwell Stinchcombe. “Multilayer feedforward networks are universal approximators”. In: (1989).

[43] Euisun Choi Chulhee Lee. “Feature extraction based on the Bhattacharyya distance”. In: (2003).

[44] James Melville Leland McInnes John Healy. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: (2018).

[45] Brent Smith Greg Linden. “The Test of Time: Two Decades of Recommender Systems at Amazon.com”. In: (2017).

[46] et al Lu Jiang Shoou-I Yu. “Bridging the Ultimate Semantic Gap: A Semantic Search Engine for Internet Videos”. In: (2015).

[47] Laurens van der Maaten Geoffrey Hinton. “Visualizing Data using t-SNE”. In: (2008).

[48] Gholamreza Haffari Ming Liu Wray Buntine. “Learning How to Actively Learn: A Deep Imitation Learning Approach”. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics** (2018).

[49] Steven J. Nowlan and Geoffrey E. Hinton. “Simplifying Neural Networks by Soft Weight-Sharing”. In: (1992).

[50] Christopher Olah. “Understanding LSTM Networks”. In: (2015).

[51] Charu C. Aggarwal Srinivasan Parthasarathy. “Mining Massively Incomplete Data Sets by ConceptualReconstruction”. In: (2001).

[52] Julian McAuley Ruining He. “VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback”. In: (2015).

[53] Ruslan Salakhutdinov and Andriy Mnih. “Probabilistic Matrix Factorization”. In: (2008).

[54] Barry Schwartz. “The Paradox of Choice: Why More is Less”. In: (2005).

[55] Andrew Trask et al Theo Ryffel. “A generic framework for privacy preserving deep learning”. In: (2018).

[56] Elena Smirnova Flavian Vasile Thomas Nedelec. “Content2Vec: Specialiazing Joint Representations of Product Images and Text for the Task of Product Recommendation”. In: (2017).

[57] Paolo Cremonesi et al Yashar Deldjoo Mehdi Elahi. “Content-Based Video Recommendation System Based on Stylistic Visual Features”. In: (2016).

[58] Robert Bell Yehuda Koren and Chris Volinsky. “Matrix Factorization Techniques For Recommender Systems”. In: (2009).

[59] Yongfeng Zhang and Xu Chen. “Explainable Recommendation: ASurvey and New Perspectives”. In: (2019).

[60] Haitao Zhao and Pong Chi Yuen. “A Novel Incremental Principal Component Analysisand Its Application for Face Recognition”. In: (2006).

[61] Karen Simonyan Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: (2015).

A Ethical and professional considerations

Regarding the ethics checklist completed at the beginning of the project we had one legal issue to address - namely, the use of information for which there are legal implications. We use YouTube movie trailers extensively in this project for which copyright regulation is in effect in certain jurisdictions including the USA where our YouTube trailers were downloaded and preprocessed.

However, we refer to US federal government regulation on *fair use* which relates to this matter. To directly quote from `www.copyright.gov`:

> Fair use is a legal doctrine that promotes freedom of expression by permitting the unlicensed use of copyright-protected works in certain circumstances. Section 107 of the Copyright Act provides the statutory framework for determining whether something is a fair use and identifies certain types of usessuch as criticism, comment, news reporting, teaching, scholarship, and researchas examples of activities that may qualify as fair use.

As our usage of this data constitutes scholarship and/or research and as such qualifies as *fair use*.

Additionally, Section 1 provides the summary of our in depth assessment of how the commercialisation of this work could take effect.

B Additional collaborative filtering results

B.1 Data sampled at 20 reviews per user on 1000 movie 1500 user dataset

Table 12: Accuracy metrics in non and hard-cold start scenario with sampling at 20 ratings per user.

	Item Non Cold Start			**Item Hard Cold Start**		
	MAE	RMSE	MSE	MAE	RMSE	MSE
DCfAvCont(1)	0.708	0.911	0.829	0.806	1.012	1.025
DeepCf	**0.671**	**0.861**	**0.741**	**0.775**	**0.979**	**0.958**
RegSVD	0.732	0.922	0.850	0.815	1.007	1.015

Table 13: Rank metrics in non-cold start scenario (k=4) with sampling at 20 ratings per user.

	Item Non Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	0.522	0.447	0.481	0.605	0.395	0.612
DeepCf	**0.576**	**0.508**	**0.540**	**0.637**	**0.476**	0.690
RegSVD	0.471	0.311	0.375	0.494	0.301	**0.693**

Table 14: Rank metrics in hard-cold start scenario (k=4) with sampling at 20 ratings per user.

	Item Hard Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	**0.556**	**0.500**	**0.526**	**0.646**	**0.438**	**0.615**
DeepCf	0.263	0.276	0.270	0.309	0.240	0.589
RegSVD	0.000	0.000	0.000	0.000	0.000	0.583

Table 15: Rank metrics in non-cold start scenario (k=10) with sampling at 20 ratings per user.

	Item Non Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	0.521	0.435	0.474	0.606	0.381	0.665
DeepCf	**0.574**	**0.500**	**0.535**	**0.638**	**0.465**	0.720
RegSVD	0.470	0.295	0.363	0.494	0.285	**0.722**

Table 16: Rank metrics in hard-cold start scenario (k=10) with sampling at 20 ratings per user.

	Item Hard Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	**0.556**	**0.492**	**0.522**	**0.646**	**0.426**	**0.667**
DeepCf	0.264	0.297	0.280	0.309	0.251	0.646
RegSVD	0.000	0.000	0.000	0.000	0.000	0.644

Table 17: Other metrics in non-cold start scenario with sampling at 20 ratings per user.

	Item Non Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	0.628	**0.413**	0.992	**1.962**	**2.556**
DeepCf	**0.795**	0.382	0.993	1.937	**2.556**
RegSVD	0.537	0.216	**0.994**	1.067	**2.556**

Table 18: Other metrics in hard-cold start scenario with sampling at 20 ratings per user.

	Item Hard Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	0.573	**0.464**	0.991	**2.199**	**2.556**
DeepCf	**0.953**	0.232	**0.999**	1.398	**2.556**
RegSVD	0.000	0.000	0.000	0.000	**2.556**

B.2 Data sampled at 80 reviews per user on 1000 movie 1500 user dataset

Table 19: Accuracy metrics in non and hard-cold start scenario with sampling at 80 ratings per user.

	Item Non Cold Start			**Item Hard Cold Start**		
	MAE	RMSE	MSE	MAE	RMSE	MSE
DCfAvCont(1)	0.635	0.827	0.684	**0.744**	**0.941**	**0.886**
DeepCf	**0.624**	**0.809**	**0.655**	0.754	0.952	0.907
RegSVD	0.642	0.826	0.683	0.792	0.987	0.975

Table 20: Rank metrics in non-cold start scenario (k=4) with sampling at 80 ratings per user.

	Item Non Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	0.784	**0.730**	0.756	0.883	**0.690**	0.586
DeepCf	**0.790**	0.726	**0.756**	**0.883**	0.688	**0.586**
RegSVD	0.758	0.699	0.727	0.850	0.661	0.577

Table 21: Rank metrics in hard-cold start scenario (k=4) with sampling at 80 ratings per user.

	Item Hard Cold Start					
	P@4	R_{adj}@4	F1@4	MRR@4	MAPK@4	NDCG@4
DCfAvCont(1)	**0.424**	**0.358**	**0.388**	**0.479**	**0.327**	**0.397**
DeepCf	0.279	0.253	0.266	0.324	0.220	0.296
RegSVD	0.113	0.113	0.113	0.132	0.098	0.303

Table 22: Rank metrics in non-cold start scenario (k=10) with sampling at 80 ratings per user.

	Item Non Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	0.752	**0.605**	**0.671**	**0.883**	**0.553**	**0.687**
DeepCf	**0.757**	0.596	0.667	0.883	0.547	0.687
RegSVD	0.726	0.581	0.646	0.850	0.530	0.681

Table 23: Rank metrics in hard-cold start scenario (k=10) with sampling at 80 ratings per user.

	Item Hard Cold Start					
	P@10	R_{adj}@10	F1@10	MRR@10	MAPK@10	NDCG@10
DCfAvCont(1)	**0.402**	**0.302**	**0.345**	**0.463**	**0.260**	**0.511**
DeepCf	0.275	0.255	0.265	0.315	0.203	0.415
RegSVD	0.112	0.119	0.116	0.132	0.095	0.426

Table 24: Other metrics in non-cold start scenario with sampling at 80 ratings per user.

	Item Non Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	**0.940**	**0.660**	0.964	**6.685**	**8.334**
DeepCf	0.927	0.655	0.964	6.532	**8.334**
RegSVD	0.886	0.631	**0.965**	6.502	**8.334**

Table 25: Other metrics in hard-cold start scenario with sampling at 80 ratings per user.

	Item Hard Cold Start				
	Cov.	Div.	Pers.	Avg Rec L	Avg Rel L
DCfAvCont(1)	**0.999**	**0.348**	0.990	**3.619**	**8.334**
DeepCf	**0.999**	0.272	**0.995**	3.434	**8.334**
RegSVD	0.499	0.117	0.499	1.579	**8.334**

C Effect of different feature aggregation techniques on ranking results

Table 26: The effect of different feature aggregation techniques on model performance. All results are from the 1000 movie 1500 user dataset (40 ratings per user) and the audiovisual content data has been reduced to 600 dimensions per subclip using incremental PCA.

		Item Non Cold Start			**Item Hard Cold Start**		
		MRR	MAP	NDCG	MRR	MAP	NDCG
DCfAvCont(1)	Avg	0.769	0.546	0.665	0.287	0.223	0.509
	AvgVar	0.784	0.559	0.660	0.315	0.243	0.483
	Med	0.790	0.577	0.662	0.295	0.226	0.510
	MedMad	0.783	0.560	0.670	0.325	0.252	0.499
DCfAvCont(2)	Avg	0.790	0.578	0.668	0.257	0.193	0.417
	AvgVar	0.802	0.593	0.668	0.241	0.184	0.422
	Med	0.778	0.559	0.662	0.284	0.215	0.436
	MedMad	0.786	0.577	0.669	0.269	0.206	0.433
LSTMAvCont	Avg	0.767	0.554	0.654	0.517	0.358	0.508
	AvgVar	0.763	0.557	0.656	0.587	0.388	0.510
	Med	0.794	0.586	0.655	0.601	0.416	0.506
	MedMad	0.781	0.579	0.654	0.616	0.419	0.514

D System design

D.1 How to run the code

Our code is implemented in an object-oriented format. All run modes have an associated run driver and all run cans be configured and started from the *main.py* method. A UML diagram of the code is provided with the repository.

D.2 Designing and implementing a performant audiovisual feature extraction pipeline

To process the audiovisual data at the scale we did (¿12K trailers of 350GB in size in compressed form) we had to be very mindful of the RAM requirements of our models. Moreover, we had to diligently handle exceptions in the pipeline as otherwise overnight jobs could crash resulting in lost time.

Regarding RAM requirements, we made use of *generators* in Python. This is a way of implementing *lazy evaluation* - that is to say, a programming paradigm in which an item of data is only brought into memory when it is requested. This allowed us to feed the visual and audio frames of the trailer files to the feature extractors on a frame by frame basis.

Regarding exception handling, we wrote our own exception handling framework that caught all of the possible video file exceptions that our underlying video processing utility *ffmpeg* generated. These included various weird and wonderful types of corruption to the videos' audio and visual streams. Our general logic was to cease to use a trailer which generated an exception as our dataset is so large than we can afford to exclude certain trailers from it.

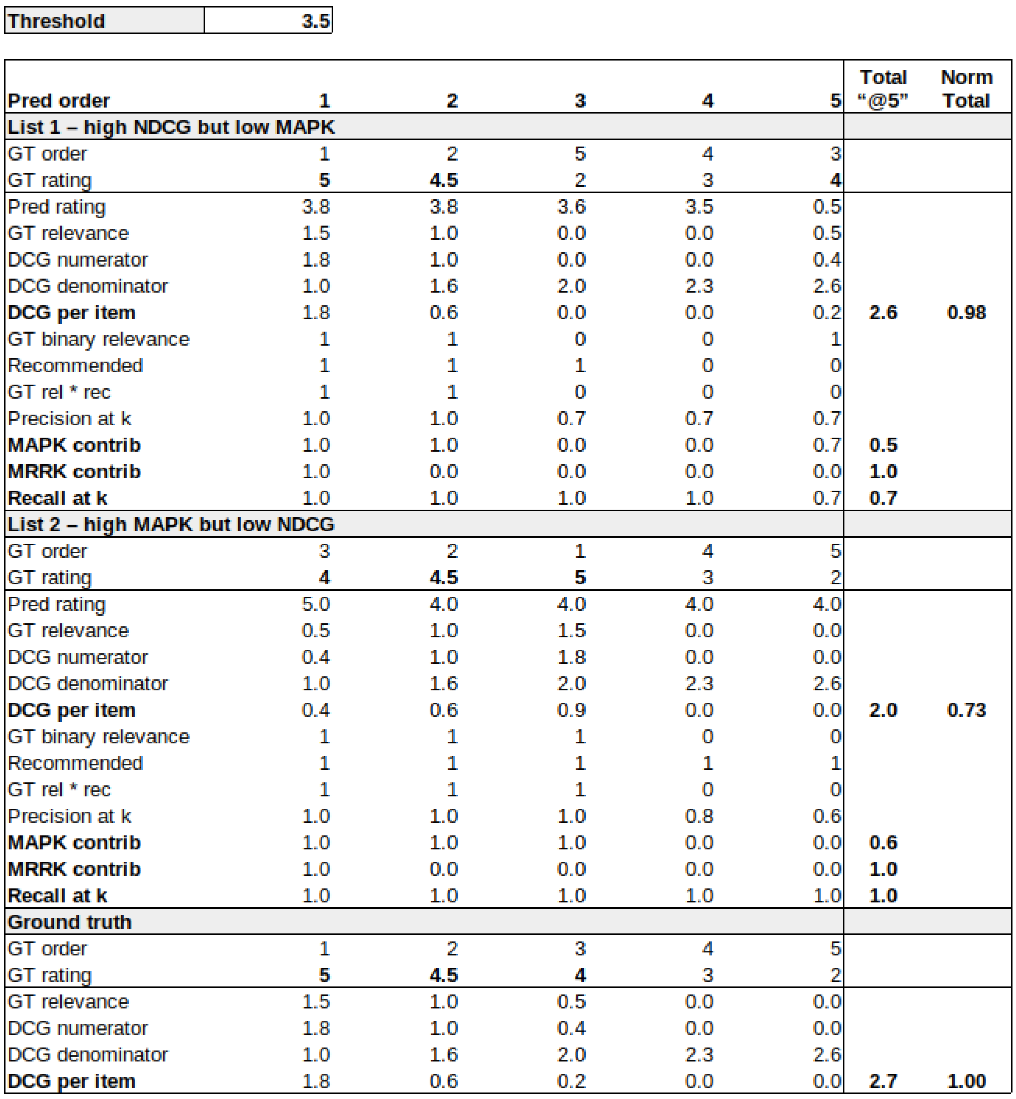
When passing our extracted features to our collaborative filtering neural networks we also made use of the recent *Python* utility *diskcache*. This is an excellent piece of technology that allows you to persit a key-value store to disk and access items from it on request with low-latency. Moreoever, it supports multiprocessing (i.e. it allows concurrent access to the store) and thus allowed us to build our own custom data generators and data loaders for Keras.

E Example calculation of recommender system rank metrics

Figure 28 shows two example recommendation lists generated for the same user and the associated rank metrics that result from this. What is important to note from this “toy” example is that different rank metrics reward slightly different properties of the recommended list.

For example, the algorithm that generates List 1 places the two highest reviewed ground truth items first and thus scores a larger NDCG@5 value versus the algorithm behind List 2. However, its recall capabilities are lower and thus it fails to to recommend ground truth item 3 and thus scores a lower MAPK@5 value.

Therefore, this illustrates how the NDCG metric puts more emphasis on placing the most relevant items at the top of the list whereas MAPK puts comparatively more emphasis on ensuring all relevant items are recalled.



Threshold	**3.5**

Pred order	**1**	**2**	**3**	**4**	**5**	**Total "@5"**	**Norm Total**
List 1 – high NDCG but low MAPK							
GT order	1	2	5	4	3		
GT rating	**5**	**4.5**	2	3	**4**		
Pred rating	3.8	3.8	3.6	3.5	0.5		
GT relevance	1.5	1.0	0.0	0.0	0.5		
DCG numerator	1.8	1.0	0.0	0.0	0.4		
DCG denominator	1.0	1.6	2.0	2.3	2.6		
DCG per item	1.8	0.6	0.0	0.0	0.2	**2.6**	**0.98**
GT binary relevance	1	1	0	0	1		
Recommended	1	1	1	0	0		
GT rel * rec	1	1	0	0	0		
Precision at k	1.0	1.0	0.7	0.7	0.7		
MAPK contrib	1.0	1.0	0.0	0.0	0.7	**0.5**	
MRRK contrib	1.0	0.0	0.0	0.0	0.0	**1.0**	
Recall at k	1.0	1.0	1.0	1.0	0.7	**0.7**	
List 2 – high MAPK but low NDCG							
GT order	3	2	1	4	5		
GT rating	**4**	**4.5**	**5**	3	2		
Pred rating	5.0	4.0	4.0	4.0	4.0		
GT relevance	0.5	1.0	1.5	0.0	0.0		
DCG numerator	0.4	1.0	1.8	0.0	0.0		
DCG denominator	1.0	1.6	2.0	2.3	2.6		
DCG per item	0.4	0.6	0.9	0.0	0.0	**2.0**	**0.73**
GT binary relevance	1	1	1	0	0		
Recommended	1	1	1	1	1		
GT rel * rec	1	1	1	0	0		
Precision at k	1.0	1.0	1.0	0.8	0.6		
MAPK contrib	1.0	1.0	1.0	0.0	0.0	**0.6**	
MRRK contrib	1.0	0.0	0.0	0.0	0.0	**1.0**	
Recall at k	1.0	1.0	1.0	1.0	1.0	**1.0**	
Ground truth							
GT order	1	2	3	4	5		
GT rating	**5**	**4.5**	**4**	3	2		
GT relevance	1.5	1.0	0.5	0.0	0.0		
DCG numerator	1.8	1.0	0.4	0.0	0.0		
DCG denominator	1.0	1.6	2.0	2.3	2.6		
DCG per item	1.8	0.6	0.2	0.0	0.0	**2.7**	**1.00**

Figure 28: Example of how ranking metric values vary relative to each other for two *dummy* recommendation lists.

F Additional views of MovieLens AV rating distributions

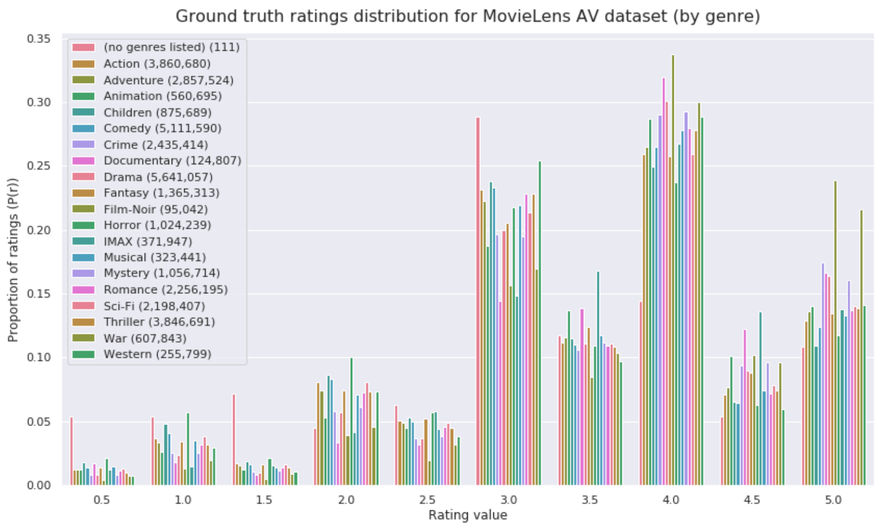


Figure 29: Ground truth rating value distribution by genre in MovieLens AV dataset.

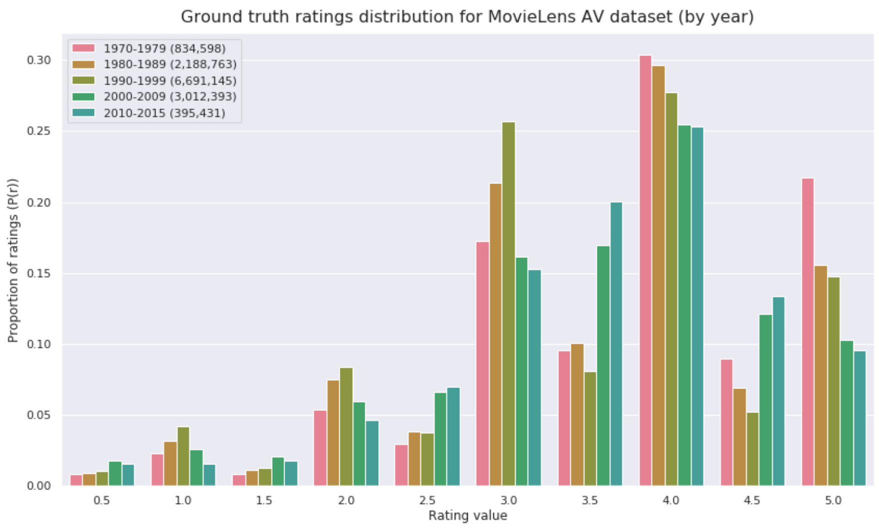


Figure 30: Ground truth rating value distribution by year in MovieLens AV dataset.