What Do you Like? - Multi-Network for Implicit Video Recommendations

Video recommendation systems rely on their viewers' past choices in order to predict their future preferences. This is a reliable method when viewers actively provide feedback (explicit) such as 1-5 ratings. However, when feedback is inferred by observing viewer's activity (implicit) these recommendation systems ignore a significant variable. This variable can be taken into account once answering the following question: are people always satisfied with their past decisions? The answer, as any human being would probably agree, seems to be a negative one. This paper aims to better predict implicit data recommendations by looking not only at a viewer's positive experience (likes), but also adding to the equation its negative experience (dislike); unlike most implicit data recommenders, our model can predict not only that John has disliked a movie such as 'There Will Be Blood' after watching it fully, but also deduce that his or hers dislike of this particular movie stems from the viewers' more general preference, of watching videos without any violence. To do so, we introduce the novel **Like** model: a multi-network combining an RNN and a funnel Neural Network meeting at the output layer: making use of a novel video subtitle feature as well as common user and video features as age and genre. Testing using Britain's Channel 4 database, our **Like** model proves to be predictively competitive with state of the art recommendation systems. The explicit learning of disliked video features contributes both to stronger predictions and avoidance from recommending a video which is predicted to be disliked by the viewer.

CCS Concepts: • **Computing methodologies** → **Learning from implicit feedback**; **Neural networks**; *Multi-task learning*;

Reference Format:
. What Do you Like? - Multi-Network for Implicit Video Recommendations. 1, 1 (March 2019), 10 pages. https://doi.org/

1 INTRODUCTION

"Did you like this video?"
Finishing a video at Netflix, this is a question you might have encountered. Yet, how many of those questions did you actually answer? Don't worry to say "none", you won't be alone. Implicit data, i.e. feedback retrieved indirectly from user's behavior, account for the majority of the data collected from video platforms. Still, most of video platforms recommend millions of videos to their users every day by asking themselves one single question: What did the user watch in the past? These video recommendation systems use implicit feedback data to base their predictions upon a person's past experience. But while this may seem a reasonable method to use, left on its on, it might lead to an unpleasant user experience. Take the following example: Jane used her Sunday night to watch "Troy" together with her parents. She did not like it. But how many period dramas does she now need to watch until something likable pops up again?

Author's address:

1.1 Problem

The gist of our problem is to recommend which particular video a person should watch next, given previous videos watched. More specifically, we want to know *What* to recommend, *When* to recommend it and *Where* to place the recommendation(s). We shall call this problem the **Joint** recommendation problem (figure 1).

What to recommend, poses a challenge. Imagine Jane, a user of an online video website. Despite using the website on a daily basis, our knowledge about her is at best partial. We may know which videos she has watched, but little about how much she liked them. Rating videos has never really been her thing. But to predict Jane's next view, it is exactly these likes and dislikes that we need to understand. Previous models have provided a good foundation [Adomavicius and Tuzhilin 2005][Rajaraman and Ullman 2011][Bell et al. 2007b][Salakhutdinov et al. 2007]. Yet, there is ample room for improvement. While some of the recommender systems acknowledge dislikes, they may tackle it either implicitly through broad assumptions [Balazs Hidasi 2016]: popular + unwatched = disliked. Or in a way [Hu et al. 2008] that we can not differentiate the likes from the dislikes. Most of the approaches ignore the relevance of dislikes in the prediction of a user's next view. But as our paper argues, dislikes matter for at least two reasons. First, we learn from psychologists and social scientists that a person's preferences and actions are influenced by both, positive and negative experiences [Elster 1989]. Therefore, knowing what a person dislikes will provide us with a good negative signal for separating positive and negative videos. Second, a negative experience can be far more influential despite it being of equal intensity to a positive experience [Kensinger 2009][Rozin and Royzman 2001]. Think, for example, do you appreciate every moment the road is clear during a morning commute to work, or are you more likely to only complain once you are stuck in a traffic jam?

Once we have learned what a person likes to watch, we need to understand ***When*** that person wants to watch it. It is likely that the preferences of a user change over time. [Wu et al. 2017] has contributed on that subject matter, by separating recommendation problems into two categories: Those that are time dependent like RNNs[Hidasi et al. 2015] and TimeSVD++ [Koren 2009], also called sequential recommendation problems. The other category is time independent, the so-called spatial recommendation problems as the matrix completion methods[Salakhutdinov and Mnih 2007] and Restricted Bolzman Machines[Salakhutdinov et al. 2007]. In this light, extra attention shall be paid to new users and the problem of cold start [Schein et al. 2002]. If a user is new to a platform, no history exists that could help us recommend him or her when to watch which video. But does that mean we need to leave him or her alone in the jungle of videos and TV shows?

Lastly, it is not enough to know when to recommend what, but also ***Where*** to place that recommendation. Often for a service provider it makes sense to provide more than one option. Yet, space is limited; a screen can only present to a user few recommendations.

Moreover, recent studies in behavioral economics show that the amount of options and the variety of options provided, influence the decisions a user makes. For example, placing a comedy next to two tragedies may increase the chances of watching that comedy. Hence, for a user to find its way and have a positive experience, a service provider has to organize its products smartly [Anderson 2004]. Our paper aims to fill those research gaps and address the **Joint** recommendation problem by using data from Britain's Channel 4 database.

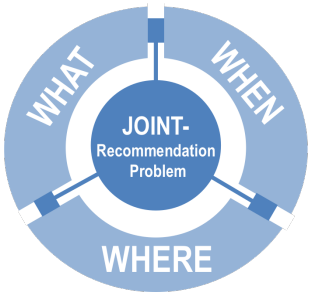

Fig. 1. The **joint** recommendation problem diagram; Every recommendation problem is the joint of *What* to present, *When* to present it and *Where* to place it?. The *What* further breaks down to likes and dislikes

1.2 Data

So often with problems grounded in the real world, their complexity exceeds that of the initial generalized problems. In the field of video recommendation that additional layer of complexity is the availability of data. Currently, the majority of data collected is implicit feedback data. That means preferences can only be inferred indirectly from user behavior. For example, if Jane watched the video "Sliding doors" we will find a "1" in our data, while otherwise we would see a "0" or a blank. While implicit data is much easier to collect, it is also more difficult to interpret. In other words, could we infer that Jane must have liked the video just because she watched it?

Currently most implicit feedback recommender systems are similar to the "Sliding Doors" video; once you have picked a video, it is considered as a positive experience and your recommendation path is determined accordingly. Such are most content based techniques [Adomavicius and Tuzhilin 2005][Rajaraman and Ullman 2011] that construct user profile according to any observed activity, collaborative filtering methods such as matrix factorization [Salakhutdinov and Mnih 2007][Bell et al. 2007b], Restricted Bolzman Machines [Salakhutdinov et al. 2007] and even the newer hybrid methods as RNNs [Balazs Hidasi 2016] and multi-networks [Wu et al. 2016].

The limits of such a "Sliding Door" approach become especially apparent if viewed through the lens of our **Joint** recommendation problem. Imagine the case where Jane has just finished watching the comedy "Hangover", even though not having liked the movie. Based on her implicit data track record, the system now recommends her to watch "Hangover 2". So, how will Jane react? Will she watch "Hangover 2", or will she not? Addressing the limits of the 'Sliding Door' and spare Jane from a negative experience in the first place is one of the main objectives of this paper.

1.3 Model

After examining our **Joint** recommendation problem and its complexities, we can specify the solution we are searching for:

• *What* interpretable: a solution that can argue for implicit feedback datasets why a certain video was watched, i.e. did the user like or dislike it?

• *When* temporal: our solution needs a) to predict videos in a sequential manner and b) use the sequential nature of human choice to provide predictions.

• *Where* ranked: our solution needs to optimize towards an evaluation metric, i.e. taking into account where a certain recommendation was presented to the user.

Besides these three main features, we want our solution to be:

• *Comprehensive*: our solution needs to produce a prediction regardless whether an item or user is new (cold start) as well as it takes into account the contextual information and the user's/item's explicit features.

• *Real-time*: our solution needs to be scalable and fast enough to produce recommendations in a real world platform such as Britain's Channel 4 database.

A solution that is interpretable, temporal, ranked, as well as comprehensive and practical in real-time, currently does not exist and requires a novel approach. We, therefore, propose our novel **Like** model. The **Like** model incorporates a vanilla RNN - temporal and practical (real-time) - into a novel architecture that a) introduces a Funnel Neural Network to learn about the video features a user dislikes, which is then combined with the RNN at the output layer b) introduces a new Subtitle2Vec feature for comprehensive video representation, and c) uses an appropriate loss to rank videos. Accordingly, our hypotheses are: 1. Incorporating disliked videos to recommendation systems can lead to a better prediction accuracy. 2. Incorporating explicitly disliked video features can produce an interpretable prediction. i.e. one can see which videos are driven more by the dislike portion of the network.

1.4 Results

Our results suggest great potential for our **Like** model: The **Like** model produces around 53 percent for top 15 accuracy over 7724 labels, which is competitive with state-of-the-art results[Hidasi et al. 2015]. Furthermore, to our knowledge, this is the first model allowing clear interpretation for whether a video is driven by likes dislikes.

The results stand out as we compare our Like model with our implementation for other prominent solutions from the field of recommendation systems. Comparing it with content-based item-to-item and collaborative filtering item-to-item methods, we find an increase of 2-50 percent points to the best results achieved by these models.

1.5 Paper structure

This paper comprises of four sections. The first section discusses the most prominent literature currently available in the video recommendation systems field with some broader look on recommendation systems in general. The second section presents our model design where we introduce our **Like** multi-network model that enables the interpretation of likes versus dislikes. In the third section we test the robustness of our model and discuss the results of our findings. Finally, the last section will outline our conclusion and provide suggestions for future research.

2 RELATED WORK

Based on comprehensive works by [Adomavicius and Tuzhilin 2005] [Rajaraman and Ullman 2011], we can organize the solutions for recommendation systems, as presented in Figure 2. Solutions are most commonly divided to one of content based, collaborative filtering (CF) [Herlocker et al. 2000], or hybrid methods [Adomavicius and Tuzhilin 2005]; Each one of them can then be applied using either a model based approach or a memory/heuristic based approach. Modern video recommendations are mainly concerned with Collaborative filtering and Hybrid methods and hence content based are not detailed below.

Using explicit dislikes, which is our main contribution, in video recommendation literature is scarce. Taking into account negative feedback, which we term dislikes (video watched less than 25 percent of its total duration) has not been incorporated explicitly in the implicit video recommendation field. i.e. Finding characterization of disliked features. The idea appeared as well in different forms in other fields. However, it is argued by [Gauch et al. 2007] that it is hard to get negative feedback in an implicit way, since users mainly interact with content they are interested with, and hence it is more rare in literature.

In this section we will look at Collaborative filtering and Hybrid approaches. After short explanation of the approach, we survey video recommendations using implicit datasets with both positive and negative feedback. Then we will look at video recommendations using implicit datasets with only positive feedback, video recommendations using explicit datasets and lastly general recommendation systems.

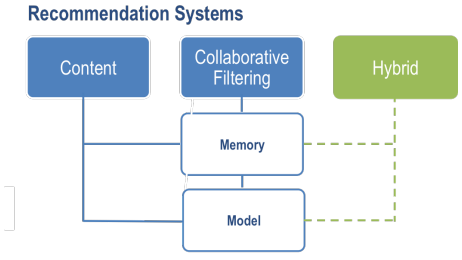

Fig. 2. The recommendation solution framework. Solutions are either content based, collaborative filtering based or a hybrid combination. Each one of these can then be applied using a memory based method or a model based method

2.1 Collaborative filtering

Collaborative filtering[Adomavicius and Tuzhilin 2015] is the umbrella name for all methods that use similarity among users' activity or items' ratings to offer recommendations. The main idea at its core is the construction of the utility matrix. The aim is to fill each partially missing matrix row (representing a user) either in full or with significant K values.

But what do we do when we only have implicit datasets? since even if we can reconstruct the perfect matching of the utility matrix it might still be useless in practice - did we predict a 1 representing what the user likes or is it a 1 representing the user's boredom.

Video recommendations using Implicit dataset with negative feedback The closest to our video recommendation idea of decomposing the *What* to likes and dislikes is [Hu et al. 2008] and its extension by [He et al. 2016]: which offer a matrix factorization technique. Their idea is to incorporate positive and negative feedback by the confidence it has in that video. So that, if a video is watched for 25 percent of its total duration it can be weighted by 0.25. In this way it accounts for user's dislikes.

The search for a good explanation to an implicit recommendation is very important [Herlocker et al. 2000] and although the latter method suggests a linear model to explain how the score of one prediction is comprised of other actions, it is still unable to say if a video was liked or disliked. Such is the case when a similar video contributed most of the prediction to another, both have been watched but both disliked. i.e. it can be inferred that one video is the driver of another but whether they are liked or not is vague - can only be implicitly inferred when driving videos are with confidence lower than 1. In contrast, we would like to learn an explicit feature representation indicating the disliked video characteristics of a user separated from those liked by her. We try to simplify what was offered by [Hu et al. 2008] by simply looking which portion of our Like multi-network contributed more to the general score of the item. Further, we will consider explicit features and treat the problem in a sequential manner.

Video recommendations using Implicit dataset with only positive feedback some of the prominent state of the art results are offered by that offer a video recommendation system based on co-clustering. This can be viewed as an extension of matrix factorization techniques. The paper tries to offer an interpretable recommendation (which we are interested in as well). The interpretation is very helpful to explore the space of what users like but still do not put emphasis on what the users dislike. Aside from the fact that the authors do not consider negative feedback, the method is also time independent and is made as a B2B where we concentrate on B2C (end client focused).

Video recommendations using explicit datasets There are far more research around explicit datasets for video recommendations which boomed with the Netflix prize and its top performers [Paterek 2007][Bell et al. 2007a][Koren et al. 2008]. But as mentioned by the methods above, matrix factorization techniques are mostly time independent, while humans' preferences are time dependent [Kolter and Maloof 2003]. Even the notable Netflix prize winner TimeSVD++ [Koren 2009], which does take time into account needs to account for time in a manual manner - time window is hand picked.

In the recent years with the rise of Neural Networks and its family, many other methods were presented for matrix completion among them Restricted Bolzman Machines (RBM) [Salakhutdinov et al. 2007][Hinton et al. 2006; Salakhutdinov and Mnih 2007] and auto-encoders [Sedhain et al. 2015]. The latter produced state of the art results but was only applied on explicit videos' rating data sets.

General recommendation systems an older renowned method which is still being used today in many recommendation systems due to its simplicity and great results is the item-to-item memory based method by Amazon [Linden et al. 2003]; where we look at items' similarity in the utility matrix. We use this method as one of our baselines.

Takeaways From the collaborative filtering method we would like to take user's activity and its relation to other users. In addition to these features, we would like to add explicit features as well and take into account the time dimension. The former is complementary to latent factors found in the data while the latter is accounting for the nature of people's choice and predictions. All of the above leads us to the search for a hybrid method.

2.2 Hybrid methods

Hybrid methods combine characteristics both from content based methods and collaborative filtering methods. The hybrid combination helps us circumvent issues[Balabanović and Shoham 1997][Basu et al. 1998][Claypool et al. 1999][Schein et al. 2002] presented either in content based or in collaborative filtering approaches or in both, some of which mentioned above.

Video recommendations using Implicit dataset with negative feedback [Hidasi et al. 2015] introduces an RNN model which both treat the the problem in sequential manner and is applied to implicit data set and experimented on Youtube like dataset. - which we use as our top baseline model and which we are able to show competitive results. It does note the importance of dislikes, but only incorporates it through the loss. It assumes that if a video is popular and was not watched, then a dislike can be established. While this assumption might produce good results, it is reasonable to doubt its generalization. Furthermore, it applies the solution on session based history, in contrast to our long term user history.

offer a deep learning hybrid approach for Youtube video recommendations. As in [Hidasi et al. 2015] the incorporation of negative feedback is done through the loss with "candidate sampling" and importance weighting correction, and not through explicit representation of disliked video features. Also, historical user views are averaged and used as part of the input for the neural network; in such a way some temporal information can be lost. We hope to capture more information by using a Recurrent Neural Network.

Video recommendations using Implicit dataset with positive feedback only Despite the rise of deep learning hybrid models, the strength and popularity of Dimensionality reduction techniques still persist in hybrid approaches. Such example is inductive matrix completion; enabling to include also features like age, gender and postcode for the user, and features like genres and actors for the video [Jain and Dhillon 2013]

Video recommendations using explicit datasets Another method in the same sphere of dimensionality reduction but with more general properties is Tensor decomposition. This method is promising since unlike matrix decomposition methods, the initial matrix (Tensor) can be in high dimensional space and thus much richer in features and even include temporal dynamics [Anandkumar et al. 2014] [Frolov and Oseledets 2017]. However, to date, tensor decomposition is only applicable to a very specific space of problems and does not yet have much research background.

like our model, [Frolov and Oseledets 2017] tensor decomposition method suggests the importance of considering negative feedback towards prediction. But it is motivated by the cold start problem, and as a by product improves accuracy using negative feedback. This work concerns with an explicit data set and also does not offer a solution to the prediction problem but more to interpolation problem similar to matrix decomposition techniques.

In the deep learning arena, [Wu et al. 2017] presented state of the art results using two RNN networks meeting at the output layer to predict explicit ratings. The latter network provided inspiration to the construction of our **Like** model (see more details below).

General recommendation systems Another example of hybrid methods using dimensionality reduction techniques can be viewed through the paper by [Shin et al. 2014]; which uses inductive matrix completion to recommend tumblr blogs.

[Tan et al. 2016] offers an RNN model with data augmentation to predict users' purchase behavior; It does not specify negative feedback and like [Hidasi et al. 2015] it deals only with session based recommendations.

Takeaways In our solution we choose to employ a hybrid method, since this, as mentioned in the introduction, answers the requirements of the requested solution - the joint of *What*? *Where*? *When*? under the limitation of implicit data. Most importantly, it enables us to model the negative feedback (dislikes) and thus argue for its affect on implicit data set predictions.

3 THE LIKE MODEL

3.1 Baseline model vanilla RNN

The vanilla RNN model for recommendations suggested by [Hidasi et al. 2015] is a natural choice for a baseline model to compare with and a building block to our **Like** model; since aside for its state of the art results, it is comprehensive in the way we can include both user and video features. It is practical as a real time recommendation system since the weights are fixed for one forward prediction. Additionally, it can answer the *Where* question by choosing to optimize towards one of the BPR/cross-entropy losses which we use or any other differentiable ranking loss. The temporal property of RNNs helps us to answer the When question - not only we predict in a temporal manner, but also take into account the temporal behavior of users and videos as inherent to prediction. Our prediction is the What, but the inability to use dislikes as a prediction driver and interpret predictions has driven us to search for the missing portion.

we use the popular LSTM [Hochreiter and Schmidhuber 1997] model which tries to avoid the gradient instability and thus enables to learn long term dependencies. Note that other variations like Gated Recurrent Units (GRU) [Cho et al. 2014] can as easily be applied.

The LSTM model serves as our baseline and the building block for our **Like** model. We refer the reader to [Graves 2013] for further details. In our baseline model we apply the latter which gives a score for each of the video labels. Formally,

$$Output = K * LSTM(x_l) + b \tag{1}$$

Where x_l is the concatenation of the user's age, gender, average taste, contextual features as weekday, daytime, used device and one-hot encoder of video genre and the Subtitle2Vec feature vector (see details below).

3.2 Full model

The Vanilla RNN model (LSTM) helps us answer the general questions of *What?*, *Where?* and *When?* rising in any video recommendation system. But, since we are dealing with an implicit feedback data set, our answers to these questions still do not help us practically to recommend to users; we do not know if the video was liked or not - watching a video does not imply immediately on its liking. This issue affects also our answer to the *Where* question. Since even if we have predicted correctly everything the user will watch next and ranked it accordingly, we still might present a disliked video in this list.

For that reason we construct a multi-network, that from one hand predicts views and from the other hand accounts for how well these views are liked. This is achieved by our architecture of two networks: an RNN and a funnel Neural Network meeting at the output prediction layer. - see figure 3 for the **Like** model; The Neural Network is constructed to learn a pseudo lower dimensional feature representation for disliked videos, while the RNN's aim is to predict the next video given previous fully watched videos.

Our funnel Neural Network part (dislike portion) is comprised of two layers shrinking in size. Formally, we have:

$$h_1 = RELU(Wx_d + b_{hidden}) \tag{2}$$

$$h_2 = RELU(Uh_1 + b_{hidden2}) \tag{3}$$

Where $RELU(x) = max(0, x)$, W, U and b are the Neural Network parameters and the input x_d is the concatenation of the Subtitle2Vec (detailed below) feature vector of each of the disliked videos for a particular user U (disliked := watched less than 25 percent of total video duration).

The RNN part is constructed the same way as in Vanilla RNN explained above. The output layer of the **Like** model is then comprised of:

$$dislike = Vh_2 + bias_{dislike} \tag{4}$$

$$like = K * LSTM(x_l) + bias_{like} \tag{5}$$

$$Output = like + dislike \tag{6}$$

With x_l as the concatenated input of: *viewer features* two gender features (one-hot encoding, male and female) seven age features (one-hot) for seven age bins which are uniformly distributed and 300 average user's taste features calculated by the averaging of the Subtitle2Vec embedding of the first three videos watched by each user. *video features* 300 Subtitle2Vec features (see details below) 18 genre one-hot encoded features and *contextual features* three features for part of the day, (daytime, evening, night), seven features for day of the week and four features for type of device used. Equation 6 is using "+" since we treat the dislike portion as an additional information culminating in prediction of the next view.

An extra layer of either softmax for the cross entropy loss or the sigmoid for the BPR-max loss is added during loss calculations for stable implementation. We further add a L_2 regularization term to the loss to avoid over-fitting.

Since we have modeled the dislike portion in a non temporal manner, its output is 1 layer deep and the same layer is added to the "like" portion upon all its layers.

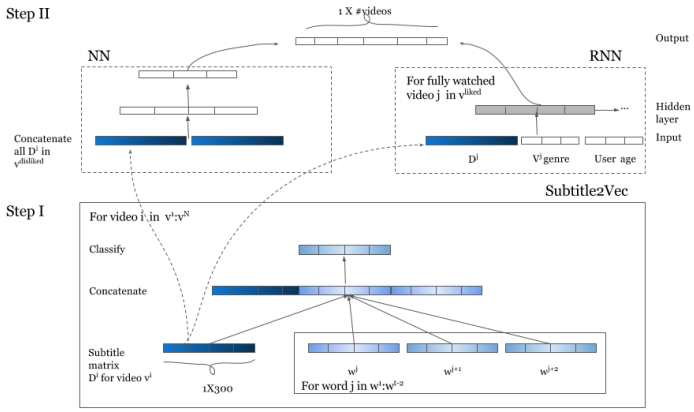

Fig. 3. The **Like** model: consists of two steps: 1. bottom graph illustrates the construction of Subtitle2Vec network for each video. After the first step is complete, its results are sent as part of the input for the second step. 2. The left network is a funnel Neural Network learning to represent hidden features of disliked videos, with its input being the concatenation of the learned subtitle feature vector of all disliked videos. The right LSTM network with its input being concatenation of the subtitle feature vector learned in the first step for a fully watched video at time t, video's genre, user's age and the device with which the video was watched. Both networks meet the at the output layer to produce a combined prediction for the next video the user will watch.

The most similar to our Like model architecture is [Wu et al. 2016] which proposes a Neural Network composed of RNN and Neural Network meeting at the output. However, it is trying to solve the heterogeneous recommendation problem of implicit and explicit data sets combined and is unable to say if a video was liked or disliked.

Though different, our inspiration comes from the paper by [Wu et al. 2017] where two RNN networks meet at each output unit in the sequence. In the **Like** model we replace one of the RNN networks with a funnel Neural Network; where the inspiration comes from [Sedhain et al. 2015] and generally from the ideas introduced by Auto-encoders [Rumelhart et al. 2013] and lower dimensional feature representation such as the one in Word2Vec and its variants.

Relating to our problem, with enough examples we would like to learn a low dimensional representation of features characterizing user's dislikes. In the same way as Auto-encoders can de-construct to lower feature space that holds enough information to characterize the higher dimensional input space, we show by experiment that our network is able to learn a predictively meaningful dislike feature space.

We choose to employ a multi-learning, meaning we construct the two networks and learn them together - in this specific case with a combined loss (This can be viewed loosely as a multi-task network). Since the two networks are learned together, we can not clearly say during training that one network is solely representing dislikes and the other solely representing likes, since the gradient with respect to the weights of the dislike network will be affected from the loss generated also from the "like" LSTM network. During testing though, after all weights are fixed, we can still see where the bigger contribution comes from.

Aside from the concept of likes versus dislikes and the model architecture, the input and loss to the **Like** model is tailored.

3.3 Model components

The input, loss function and optimization procedure are as important to the model as the architecture of the network and/or the chosen algorithm.

Our **Like** model consists of our novel feature such as the Doc2Vec video subtitle representation (Subtitle2Vec) with more common features like video genre and user's age. Moreover, we train and test against two loss functions as the cross entropy and BPR-max to provide appropriate ranking. For optimization we employ the ADAM optimizer.

3.3.1 Input feature - Subtitle2Vec. An important part of a recommender system is to find items' similarity - videos in our case. Some of the common video features to use are genres[Adomavicius and Tuzhilin 2005], which we use as well.

Another feature, which is less common for video recommendation is to use the script/subtitles of the video as a characteristic of it. This is a novel feature for video recommendation. The use case of it being: suppose a stand-up movie is tagged as a comedy genre. However, if the movie's content is about drug abuse, and your next view is of the movie 'Trainspotting' (movie about drug abuse), the subtitles would be more telling than the genre tags. Although this idea is not very common in the video recommendation industry, the idea is prevalent in written documents classification to topics[Adomavicius and Tuzhilin 2005]. i.e. as one of the features, use the words that characterize the document topics.

In order to find distributed representation of the subtitles in the video, we choose the Word2Vec [Bengio et al. 2003] method from the Natural language processing field or more appropriately its Doc2Vec extension [Le and Mikolov 2014] - see figure 4.

What is Doc2Vec? given a document (subtitles) and words $\{w_1, w_2, ..., w_n\}$ we would like to maximize the average log probability:
$\frac{1}{K} \sum_{l=k} logP(w_l|w_{l-k}, ..., w_{l+k})$.

With K being the window around each word. In simple words, we would like to find the most likely word given the previous K words. In figure 4, given the words "the", "cat" and "sat", and paragraph P we would like to predict the word "on". The Doc2Vec idea is derived from the assumption that words with similar meaning are mentioned frequently in similar contexts and thus sentences containing these words will have similar semantics.

In the Doc2Vec model, word vectors are randomly initialized and a matrix C is constructed. Furthermore, a matrix $\mathcal{D}$ is constructed to represent a unique document (subtitles) identification; The randomized vector for word i is represented by C_i and positioned in row i. This model then applies a Neural Network over C and $\mathcal{D}$ with the goal to maximize the probability of seeing the next word in the window given document and previous words in that window - usually with a fixed window size (Referring once more to figure 4). Meaning, we have a X vector which is the concatenation of the words in our fixed window and the document. This is formally then calculated by:

$$y = bias + UX \tag{7}$$

$$P(w_l|\mathcal{D}_\rangle, w_{l-k}, ..., w_{l+k}) = softmax(y) \tag{8}$$

Where C, $\mathcal{D}$ are free feature vector parameters and U, bias are the Neural Network parameters. Since for example the word cat and the word dog can interchange and expected to appear in overlapping contexts, their feature vectors will be close in Euclidean space. The feature vectors are learned simultaneously to the probability of a given word conditioned on previous words. all learning can be done with stochastic gradient descent via backpropogation.

We would like our lower distributed representation to capture ideas from the video; such can be for example the location of scenes, the underlying concepts or just the repetition of certain words. The Doc2Vec approach was chosen since it is a natural part of the Neural Network methods that are used in our architecture. Moreover, according to [Mikolov et al. 2013] it has promising results and is well scaled.

3.3.2 Loss. The loss function helps us to answer the *Where* question. By choosing the appropriate loss we can optimize towards our goal: ranking video predictions such that they are ordered; prevent disliked videos to appear on the user's screen while showing more desired videos in higher positions. This is aside from the overall need in all machine learning applications to choose an appropriate loss so that weights will learn to find an optimum and provide meaningful results.

The first loss we are using is the common cross entropy loss - which is a list-wise loss enabling ranking of the videos. Since a computational hurdles might rise with cross entropy use, we search for an alternatives. such is the Bayesian Personalized Ranking [Rendle et al. 2009] (BPR) and its BPR-max [Hidasi et al. 2015] variant loss which we apply. Formally, we define the BPR function to be:

$$BPR = -Mean(\sum_j \log \sigma(y_i - y_j)) \tag{9}$$

where $j \in negativesamples$: negative samples are considered as videos watched later than current prediction or not watched at all. i

is the ground truth position of the label and

$$\sigma(y_i - y_j) = \frac{1}{1 + \exp^{-(y_i - y_j)}} \tag{10}$$

The BPR-max loss adds to each loss term a weight corresponding to its importance; discount the importance of a negative example if it does not contribute much to the gradient update.

4 EXPERIMENTS

4.1 infrastructure

goal: for each user U_i that is not in the training set we would like to predict the video that U_i has watched next. i.e. we get all the videos U_i has watched by order $[V_i1, V_i2, ...]$ and all disliked videos with no ordering $D_i = [D_i7, D_i3, ...]$. Our funnel network (dislike portion) gets D_i as input - Concatenation of all D_i where each D_{ij} is 300 length vector representing the Subtitle2Vec of the video, and the RNN network gets at each time t the appropriate V_it which is represented by the 300 length vector representing the encoding of video subtitles to Word2Vec, concatenated with 1-hot encoding of age and 1-hot encoding representing the video genre (see the Like model section for further input details). Combining the two networks at the output using the weights learned at training time we try to predict $V_{i,t+1}$ and to indicate whether this prediction is driven by the liked or the disliked portion. Successful prediction is measured according to either Top-K or Mean Reciprocal Rank metrics - see details below.

4.1.1 Dataset. For our experiments we use Britain's Channel 4 dataset which consists of around two million users activity over approximately 5 years duration. We use the following setting:
top 2000 users from the last 2 years that fully watched at most 500 videos and 10 partially watched videos. This setting serves to show the strength and ease of interpretation of the **Like** model even on a bigger scale data. Since as observed at Britain's Channel 4, the ratio of fully watched videos to partially watched ones is around 50:1.

4.1.2 Protocol. The collected data is divided to 25 percent testing and 75 percent training; testing with the RNN baseline model and **Like** model on new users only. Since the other models (see *Baseline models* below) suffer from the cold start problem (item and/or user), we re-divide the data into 25/75 ratio such that all users are part of both training and testing.

Choosing 2000 users as a subset is limited, and can be considered as a prediction network for frequent users. We note that the same construction might produce different results if we introduce less frequent users. However, in order to introduce preliminary results for our hypotheses, these amounts of users are adequate. Even with such a subset the training procedure lasted around a week for the RNN baseline model and for the **Like** model.

4.2 Metrics

Since we are interested in a ranking which will answer *Where* to place videos on the platform, we resort to two common metrics:
• Top K - this metric evaluates whether the positive video is in one of K top predicted positions, if so we give our model 1 otherwise 0. We check for each model for $K \in \{5, 10, 15\}$. In video recommendation, the justification for using top K is that any platform suggests more than one video to watch in one screen. This metric is sometimes referred as *recall@k*.
• Mean Reciprocal Rank (MRR) - This metric is position specific and allows us to evaluate how well we ranked our positive video. If a the predicted video fall on position 15, we get the reciprocal $\frac{1}{15}$ as our MRR score [Balazs Hidasi 2016].

4.3 Baseline models

In order to evaluate how well our model is performing, we compare it with some baselines:
• Popularity - our own implementation which recommends to users the most popular items - see popularity section below for why did we check this model.
• Content based item-to-item - this model is a nearest neighbour model which was quite popular in the past. We provide here our own implementation based on the algorithm proposed in [Adomavicius and Tuzhilin 2005]
• Collaborative filtering item-to-item - this model is also a nearest neighbour model where similarity is measured through similarity of users/items in the utility matrix. Although it is already old, up until today it is a very popular model. We implement our own version of it based on the algorithm proposed by Amazon at [Linden et al. 2003]
• Hybrid RNN - one of the more recent state of the art video recommendation models. This model allows us to use all our features and hence the best comparison for our novel model - we implement a similar model to [Balazs Hidasi 2016] but we include our novel Subtitle2Vec representation which is not part of their model.

We use all these baseline models in a temporal manner, so that they are comparable to the temporal manner in which our **Like** model makes predictions. For example, for content based item-to-item, during testing:
1. pick a user U
2. for t in ordered videos watched by U:
3. observe video t, update U's profile
by a weighted average of training user profile
and t's profile.
4. calculate user's profile similarity to videos
and produce a recommendation.

4.4 Results

Results indicate that our **Like** model outperforms renowned methods like collaborative filtering item-to-item and as competitive as the Vanilla RNN with bpr-max loss model: with Mean Reciprocal Rank of 0.36 and 0.523 accuracy for top 15 videos in compare to 0.37 Mean Reciprocal Rank (MRR) and 0.538 for top 15 for the RNN with bpr-max loss - see paired T-test. All accuracy results for testing are presented in table 1.

4.4.1 Paired T-test. We have conducted paired T-tests (see [Rice 1995] for details) upon our MRR results per user; To validate that our **Like** model is significantly better not only on average. Intuitively, if the paired difference standard deviation is too large relative to the mean difference, we would not be able to conclude that one model

Metric - testing				
Model	Accuracy 15	Accuracy 10	Accuracy 5	MRR
Popularity	0.015	0.010	0.005	0
Content based	0.001	0.001	0.0005	0
Collaborative filtering	0.510	0.482	0.414	0.300
baseline RNN (xent)	0.212	0.188	0.151	0.110
baseline RNN (bpr-max)	0.538	0.506	0.452	0.371
Like model (bpr-max)	0.523	0.488	0.43	0.361

Table 1. Table presenting testing accuracy of model predictions. From left to right: share of videos appearing in top 15 predictions, top 10, top 5 and the mean reciprocal rank of a video.

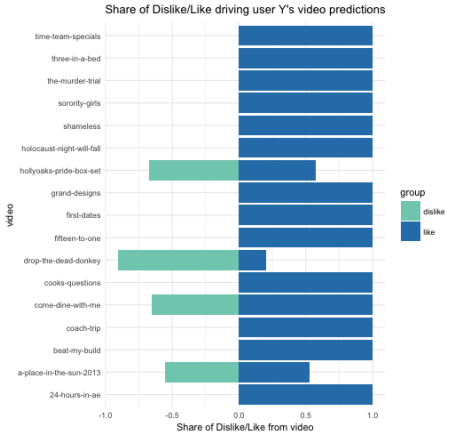

Fig. 4. Horizontal bar chart comparing the contribution share of the like portion versus the dislike portion for randomly selected user y

is better than the other. Results indicate that our **Like** model is as competitive as state of the art RNN model:
for $\mathbb{H}_0$: MRR of the RNN is equal to the MRR of the Like model, we get P = 0.3519, t = 0.9320 with 363 df - which indicate that we can not reject $\mathbb{H}_0$.

4.5 Interpretation

4.5.1 Likes vs. Dislikes. In this subsection we capture the influence of the dislike portion (Neural Network) of the network on the prediction. We present the interpretation of the results through three questions: does negative prediction contribute to prediction? do correct predictions can also be tagged as a dislike? and does the dislike network learn to contribute to the original disliked videos?

• Does negative feedback help prediction?
From the accuracy results we see that the **Like** model can make quite good prediction. If we further look at figure 6, we can see that the dislike portion pushes some of the scores towards its direction. We can say with that which videos are pseudo disliked and which correct predictions result from a dislike.

• Do the **Like** network correct predictions are driven also by dislikes?
we note that 12.5 percent from total predictions were driven more by dislikes than likes. Furthermore, 2.4 percent of correct predictions which are 2101 videos, were driven by dislikes rather than likes.

• Do we learn a dislike score for the true disliked videos?
By this we are trying to check if the input disliked videos affected their corresponding label in making a prediction. We report that on average the network learns a contribution of %30 of these labels as having a disliked portion, with the maximum being %50.

4.5.2 Popularity. Since we are picking the top users, and after results are given. We can see that item-to-item collaborative filtering is doing very well (51 ,48 and 41 percent accuracy for top 15, 10 and 5 respectively). This implies that similar users watched similar items. We can ask whether these watched videos were very popular and hence viewed by most users? firstly we note the portion which popular videos comprise from the total - around 13 percent of videos contribute around 50 percent of all views. A simple model to check this hypothesis would be to just recommend the most popular videos. We construct a simple model which uses this idea. Popularity is simply the sum of views by the users. Results for this model are poor compared to the other models but are much better than random: around 1.5, 1 and 0.5 percent accuracy for top 15, 10 and 5 respectively - see table 1.

4.5.3 Item similarity - qualitative analysis. Item similarity measures allow us to easily say which videos are similar to others. We examine this by picking few videos and checking what would be the top recommendation for a user who has watched them. We present it for similarity among users' activity in collaborative filtering item-to-item methods, and leave out content based similarity due to its poor results .

However, extracting any insight from this analysis is tricky and might imply in some examples that watching "black mirror" is a good predictor for watching "my mad fat diary" and "four in bed" and the other way around as well; this might suggest that people watch a wide variety of genres with emphasis on popular shows.

5 CONCLUSIONS

This paper has focused on the problem of whether a person likes or dislikes a video and how this can help in better recommendation and interpretation. This was seen from the perspective of the general recommendation problem of *What* to recommend? *When* to

recommend it? and *Where* to place our recommendation? which we termed as the **Joint** recommendation problem.

Accordingly, in order to solve this problem, we have constructed the novel **Like** model. This solution is competitive with recent state-of-the-art models and enables to interpret if a video is liked or disliked (What). It learns *When* should we recommend a video - achieved through the temporal nature of the RNN and *Where* to place it given by the chosen ranking loss.

The **Like** model has been measured against recent state of the art models as RNN[Hidasi et al. 2015] and was shown to outperform are be as competitive as these models.

With this solution, not only we can better predict videos, since it is argued that both negative and positive experience can direct actions, but also we can interpret if a video is going to be liked or not and thus avoid from recommending it.

This model holds few limitations which we can categorize under the **Joint** recommendation problem (figure 1) and map exactly to future research suggestions.

Firstly, regarding *What* to recommend, our **Like** model can not say in a clear cut that the portion attributed through the disliked network is a real dislike, since the loss is learned jointly to the like portion. Secondly, we limit our dislike portion by considering it as constant measure added to all time-stamps. However, we might benefit from a temporal dislike model (*When*) - as it is reasonable to consider preferences changing through time in both the negative and positive dimensions. Lastly, incorporating a more specific ranking loss could provide with a better order of *Where* to place each recommendation. These limitations lead us directly for future research opportunities.

6 FUTURE DEVELOPMENT

What - Our **Like** model could be considered as a sort of multi-task network. However, it would still be of interest to implement the network in a way such that the dislike portion is learned by an Auto-encoder prior to combining it with the RNN. A-priori dislike learning using an Auto-encoder guarantees a clear cut separation of dislikes and likes. Moreover, we can expect a richer representation of the dislike feature construction than in the **Like** model since the deconstruct-reconstruct procedure forces meaningful features.

When - Considering dislikes as changing over time, further works might benefit from two RNN networks. - architecturally close to [Wu et al. 2017]. Where one RNN network can represent the "likes", while the dislikes can be the represented by a second RNN-auto-encoder network similar to [Cho et al. 2014].

Where - we have emphasized during this work the importance of ranking videos and presenting to the user positive experiences. This can be done with more attention if the loss to which we optimize is a specialized ranking loss. One such loss is the mean reciprocal rank [Chapelle and Metlzer 2009; Shi et al. 2012] that is a list-wise loss that enforces ordering of rank, which we have used only as a metric.

Many other such losses exist[Le and Smola 2007], with many being non-differentiable. One optional solution is to use Reinforcement Learning methods. Thus, the great possibility to choose any metric that we would like, even if non differentiable and optimize accordingly.

ACKNOWLEDGMENTS

The authors would like to thank Britain's Channel 4 for providing the data.

REFERENCES

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. (2005), 734–749 pages. https://doi.org/10.1109/TKDE.2005.99 arXiv:3

G. Adomavicius and A. Tuzhilin. 2015. Context-aware recommender systems. In *Recommender Systems Handbook, Second Edition*. 191–226. https://doi.org/10.1007/978-1-4899-7637-6_6 arXiv:3

Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. 2014. Tensor Decompositions for Learning Latent Variable Models. *Journal of Machine Learning Research* 9355 (2014), 19–38. https://doi.org/10.1007/978-3-319-24486-0_2 arXiv:1210.7559

Chris Anderson. 2004. The Long Tail. *WIRED magazine* 12, 10 (2004), 170–177. https://doi.org/10.3359/oz0912041

Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (1997), 66–72. https://doi.org/10.1145/245108.245124 arXiv:10

Alexandros Karatzoglou Balazs Hidasi. 2016. Session-based Recommendation with Recurrent Neural Networks. *ICLR* (2016), 1–10. arXiv:arXiv:1511.06939v2

Chumki Basu, Haym Hirsh, and William Cohen. 1998. Recommendation as Classification : Using Social and Content-Based Information in Recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (1998), 714–720. https://doi.org/10.1.1.36.4620 arXiv:13

Robert Bell, Yehuda Koren, and Chris Volinsky. 2007a. Modeling relationships at multiple scales to improve accuracy of large recommender systems. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'07* 5, 1 (2007), 95–104. https://doi.org/10.1145/1281192.1281206

Robert M Bell, Yehuda Koren, and Chris Volinsky. 2007b. The BellKor solution to the Netflix Prize A factorization model. *KorBell Teams Report to Netflix* 2 (2007). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.142.9009

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research* 3 (2003), 1137–1155. https://doi.org/10.1162/153244303322533223 arXiv:arXiv:1301.3781v3

Olivier Chapelle and Donald Metlzer. 2009. Expected reciprocal rank for graded relevance. *Proceedings of the 18th ...* (2009), 621–630. https://doi.org/10.1145/1645953.1646033

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), 1724–1734. https://doi.org/10.3115/v1/D14-1179 arXiv:1406.1078

Mark Claypool, Tim Miranda, Anuja Gokhale, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. 1999. Combining content-based and collaborative filters in an online newspaper. *Proceedings of Recommender Systems Workshop at ACM SIGIR* (1999), 40–48. https://doi.org/10.1.1.46.3659

J Elster. 1989. *Nuts and bolts for the social sciences*. viii, 184 pages. http://www.loc.gov/catdir/description/cam023/89031449.html

Evgeny Frolov and Ivan Oseledets. 2017. Tensor methods and recommender systems. (2017). https://doi.org/10.1002/widm.1201 arXiv:1603.06038

Susan Gauch, Mirco Speretta, Aravind Chandramouli, and Alessandro Micarelli. 2007. User Profiles for Personalized Information Access. *The Adaptive Web* 4321 (2007), 54–89. https://doi.org/10.1007/978-3-540-72079-9_2

Alex Graves. 2013. Generating sequences with recurrent neural networks. preprint. *arXiv:1308.0850* (2013). https://doi.org/10.1145/2661829.2661935 arXiv:arXiv:1308.0850v5

Xiangnan He, Hanwang Zhang, Min-yen Kan, and Tat-seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16* (2016), 549–558. https://doi.org/10.1145/2911451.2911489 arXiv:1708.05024

Jonathan L Herlocker, Joseph a Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (2000), 241–250. https://doi.org/10.1145/358916.358995 arXiv:48

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *Iclr* (2015), 1–9. https://doi.org/10.1103/PhysRevLett.116.151105 arXiv:1511.06939

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 18, 7 (2006), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527 arXiv:1111.6189v1

S Hochreiter and J Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:1206.2944

Yifan Hu, Chris Volinsky, and Yehuda Koren. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings - IEEE International Conference on Data Mining, ICDM*. 263–272. https://doi.org/10.1109/ICDM.2008.22

Prateek Jain and IS Dhillon. 2013. Provable inductive matrix completion. *arXiv* (2013), 1–22. arXiv:1306.0626 http://arxiv.org/abs/1306.0626

Elizabeth A Kensinger. 2009. Remembering the details: Effects of emotion. *Emotion Review* 1, 2 (2009), 99–113. https://doi.org/10.1177/1754073908100432

J.Z. Kolter and M.a. Maloof. 2003. Dynamic weighted majority: a new ensemble method for tracking concept drift. *Third IEEE International Conference on Data Mining* (2003), 123–130. https://doi.org/10.1109/ICDM.2003.1250911

Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. *Proc. of KDD '09* (2009), 447–456. https://doi.org/10.1145/1557019.1557072

Yehuda Koren, Park Ave, Florham Park, H Database Management, and Database Applications. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), 426–434. https://doi.org/10.1145/1401890.1401944 arXiv:62

Qv Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014* 32 (2014), 1188–1196. https://doi.org/10.1145/2740908.2742760 arXiv:1405.4053

Quoc Le and Alexander Smola. 2007. Direct Optimization of Ranking Measures. *arXiv preprint arXiv:0704.3359* 1, 2999 (2007), 1–29. arXiv:0704.3359 http://arxiv.org/abs/0704.3359

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344 arXiv:69

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)* (2013), 1–12. https://doi.org/10.1162/153244303322533223 arXiv:arXiv:1301.3781v3

Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. *KDD Cup and Workshop* (2007), 2–5. https://doi.org/10.1145/1557019.1557072 arXiv:978-1-59593-834-3/07/0008

Anand Rajaraman and Jeffrey D Ullman. 2011. Mining of Massive Datasets. *Lecture Notes for Stanford CS345A Web Mining* 67 (2011), 328. https://doi.org/10.1017/CBO9781139058452 arXiv:arXiv:1011.1669v3

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (2009), 452–461. https://doi.org/10.1145/1772690.1772773 arXiv:1205.2618

John a Rice. 1995. *Mathematical Statistics and Data Analysis*. Vol. 72. 330 pages. https://doi.org/10.2307/3619963

Paul Rozin and Edward B. Royzman. 2001. Negativity Bias, Negativity Dominance, and Contagion Paul. *Personality and Social Psychology Review* 5, 4 (2001), 296–320. https://doi.org/10.1207/S15327957PSPR0504

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 2013. Learning Internal Representations by Error Propagation. In *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*. 399–421. https://doi.org/10.1016/B978-1-4832-1446-7.50035-2 arXiv:arXiv:1011.1669v3

R Salakhutdinov and A Mnih. 2007. Probabilistic Matrix Factorization. *Proc. Advances in Neural Information Processing Systems 20 (NIPS 07)* (2007), 1257–1264. https://doi.org/10.1145/1390156.1390267 arXiv:1705.05355

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th international conference on Machine learning - ICML '07* (2007), 791–798. https://doi.org/10.1145/1273496.1273596 arXiv:1606.07129

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR 02* 46, Sigir (2002), 253–260. https://doi.org/10.1145/564376.564421

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec : Autoencoders Meet Collaborative Filtering. *WWW 2015 Companion: Proceedings of the 24th International Conference on World Wide Web* (2015), 111–112. https://doi.org/10.1145/2740908.2742726

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2012. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. *Proceedings of the Sixth ACM Conference on Recommender Systems* (2012), 139–146. https://doi.org/10.1145/2365952.2365981

Donghyuk Shin, Suleyman Cetintas, and Kuang Chih Lee. 2014. Recommending tumblr blogs to follow with inductive matrix completion. In *CEUR Workshop Proceedings*, Vol. 1247.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. *arXiv* (2016), 0–5. https://doi.org/10.1145/2988450.2988452 arXiv:1606.08117

Caihua Wu, Junwei Wang, Juntao Liu, and Wenyu Liu. 2016. Recurrent neural network based recommendation for time heterogeneous feedback. *Knowledge-Based Systems* 109 (2016), 90–103. https://doi.org/10.1016/j.knosys.2016.06.028

Chao-yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*. 495–503. https://doi.org/10.1145/3018661.3018689