Cold-Start Recommendations for Audio News Stories Using Matrix Factorization

Ehsan Mohammady Ardehaly and **Aron Culotta**
Illinois Institute of Technology, Chicago, IL
emohamm1@hawk.iit.edu, aculotta@iit.edu

Vivek Sundararaman and **Alwar Narayanan**
Rivet Radio, Inc., Chicago, IL
{vivek.sundararaman,alwar.narayanan}@rivetnewsradio.com

Abstract

We investigate a suite of recommendation algorithms for audio news listening applications. This domain presents several challenges that distinguish it from more commonly studied applications such as movie recommendations: (1) we do not receive explicit rating feedback, instead only observing when a user skips a story; (2) new stories arrive continuously, increasing the importance of making recommendations for items with few observations (the *cold start* problem); (3) story attributes have high dimensionality, making it challenging to identify similar stories. To address the first challenge, we formulate the problem as predicting the percentage of a story a user will listen to; to address the remaining challenges, we propose several matrix factorization algorithms that cluster users, n-grams, and stories simultaneously, while optimizing prediction accuracy. We empirically evaluate our approach on a dataset of 50K users, 26K stories, and 975K interactions collected over a five month period. We find that while simple models work well for stories with many observations, our proposed approach performs best for stories with few ratings, which is critical for the real-world deployment of such an application.

1 Introduction

Personalized news recommendation is of growing importance due to the immediacy and diversity of online news. While most prior work in news recommendation engines assumes the user selects from a proposed list of stories (e.g., Google News [Liu *et al.*, 2010]), there is an increasing demand for applications in which content is streamed continuously (e.g., Pandora and Spotify for music).

Rivet Radio is a Chicago based start up that creates, organizes and distributes personalized audio news and information. The audio content is created throughout the day and is tagged with a rich set of attributes that describe the quantitative and qualitative aspects of the story. One of the distribution channels to the consumer is Rivet's iOS and Android mobile app. The app allows users to pause or skip to the next story at any time while listening to a story. This paper presents algorithms to deliver a predictive sequence of stories using the meta data and the user actions to maximize the listening by the user.

This domain presents several challenges that distinguish it from more commonly studied recommendation problems. First, we do not receive explicit user ratings for each item. Instead, we receive implicit feedback when the user decides to skip to the next story in the stream. Second, news stories are by definition new, which means that we will typically observe few user interactions for a story when making recommendations. Thus, the *cold start* problem [Schein *et al.*, 2002] is of critical importance in this domain. Finally, the representation of each story is a high-dimensional term vector, making it difficult to identify similar stories for recommendation.

Our proposed approach introduces new variants to recommendation systems to address these challenges. First, to incorporate the available implicit feedback, we formulate the task as a prediction problem to infer the percentage of a news story that a user will listen to before skipping. Thus, stories skipped in the first few seconds are deemed less relevant than those skipped near the end. Second, we propose a matrix factorization algorithm that jointly clusters users, n-grams, and stories simultaneously, optimizing the clusters for prediction accuracy. By reducing the dimensionality of content features, we can more accurately make recommendations for stories with few observed interactions.

We empirically evaluate our approach on data collected from the Rivet Radio news, consisting of 50K users, 26K stories, and 975K interactions collected over a five month period. We find that while simple content-based models work well for stories with many observations, our matrix factorization approach performs best for stories with few observations. We additionally find that combining models into an ensemble can further improve overall accuracy.

The remained of the paper is organized as follows: Section 2 summarizes related work, and Section 3 formalizes our problem setting. Section 4 presents our proposed model, as well as a number of baseline models, and Section 5 describes our empirical results. Section 6 concludes and outlines future directions.

2 Related work

Recommendation systems are widely used to recommend products to users. Many of the most successful recommendation systems are based on collaborative filtering (CF) [Sarwar *et al.*, 2001; Adomavicius and Tuzhilin, 2005], which leverages the ratings of similar users to make predictions. In many domains, such as movie or product recommendation, it is assumed that users provide explicit feedback for items through ratings or purchase history [Bennett and Lanning, 2007]. This type of explicit feedback is not available in our domain.

Because explicit feedback is often not available, systems built on implicit feedback (clicks, purchases, etc.) have also been studied [Das *et al.*, 2007; Liu *et al.*, 2010; Breese *et al.*, 1998; Hu *et al.*, 2008; Johnson, 2014]. These approaches are most appropriate when the user feedback indicates a positive preference – e.g., clicking on a news story or clicking on a "thumbs up" to indicate a positive preference for a song. However, in our domain, the only interaction we observe is a negative one – the user presses a button to skip to the next news story. We use information about *when* this occurs to guide learning.

In this sense, our domain is most similar to streaming music recommendation systems [Mcfee *et al.*, 2012; Vandenoord *et al.*, 2013], in which personalized playlists are streamed to the user. However, in music recommendation, implicit ratings are often constructed based on the number of times a user listens to a track [Vandenoord *et al.*, 2013]; this is less useful in the news domains, in which stories are typically only listened to once by a user. Furthermore, while the cold start problem does affect music recommendation, it is even more paramount in news recommendation (since many more stories are generated each day, and their relevance degrades more rapidly with age).

The most common approach to the cold start problem is to create a hybrid system that combines content features with collaborative filtering to make recommendations for items (or users) with little history [Claypool *et al.*, 1999; Schein *et al.*, 2002]. In this paper we combine matrix factorization with regression to address the cold start problem. Matrix factorization is commonly used in recommendation systems due to its ability to identify clusters of users and items, thereby improving generalizability from sparse data [Zhang *et al.*, 2006; Takacs *et al.*, 2008; Salakhutdinov and Mnih, 2008; Rendle and Schmidt-Thieme, 2008]. When a user or item has few observations, it can be difficult to determine its latent factors. To address this, prior work has proposed regression-based approaches, which fit a regression model of item and user attributes to infer latent attributes, the output of which are then used to make recommendations [Saveski and Mantrach, 2014; Gantner *et al.*, 2010]. Agarwal and Chen [2009] further proposes a model that performs these steps jointly, rather than in a pipeline.

Local Collective Embeddings (LCE) [Saveski and Mantrach, 2014] is a matrix factorization approach that identifies a latent space common to both item features and the rating matrix. Using this common latent space, items can effectively be mapped to the appropriate latent factor even with few ratings. While LCE has shown to be effective in cold-start settings, it has several limitations that make it ineffective in our domain: it assumes non-rated items have a zero rating, and it also does not capture well a user's overall bias and an item's popularity. Furthermore, prior use of LCE for news recommendation used explicit user feedback in the form of comments on stories; here, we use only the implicit feedback of user's skipping stories, which is a continuous, rather than discrete, variable. In this paper, we extend this approach and propose a non-linear LCE that addresses these limitations. In our approach, we decompose an (implicit) ratings matrix as a product of three matrices: user hidden concepts, term hidden concepts, and item to term matrices, where terms are textual or categorical features from each news story.

Type	Top features
Concept 1	associated press, technology, welcome
Concept 2	sports, welcome
Concept 3	marketplace, science
Concept 4	marketplace, associated press, business crimes & courts, government & politics
Concept 5	art & entertainment, lifestyle, technology
Bias	art & entertainment, lifestyle, TV & film, weekend

Table 1: Top features for each concept.

One advantage of our non-linear LCE model is that it finds hidden factors for each item feature (term) and maps it to the conceptual space. Table 1 displays the five hidden concepts produced by one run of our model, showing top features for each hidden concept as well as the overall bias term. For example, Concept 3 relates to news from marketplace.org and science news. The bias in this table shows overall item popularity; the top features are similar to Concept 5, but have more TV & film terms.

3 Problem formulation

In mobile radio applications, a news playlist is generated periodically and sent to users. The goal is to stream to users only the news stories that they would like to listen to, and to do so with minimal effort from the user. Thus, when a story is streamed, two outcomes are possible: the users may listen to the entire story, or the user may choose to skip to the next story at any point. The skip action is the primary negative feedback received by the system.

There are many ways one can encode this feedback. For example, one can simply use a binary preference matrix, where 0 indicates a skip and 1 indicates that the story was listened to completion. However, skipping in the first 5 seconds often means something different than skipping in the final 5 seconds of a story. An arbitrary cutoff could be used, but there are drawbacks to discretizing continuous variables, such as reduced statistical efficiency [Royston *et al.*, 2006]. For these reasons, in this work we define the user's rating for a story as the percentage of the story that she listens to. Let this value be $R_{i,u}$. So, if user u listens to 75% of story i, $R_{i,u} = .75$.

Each story contains the text of the story, some categorical information (e.g., tags), as well as a timestamp. In preprocessing, we convert these content features into a tf-idf vector

S_i for story i.

Let T be a list of (item, user) pairs, T_i be the list of users that rate item i, and T_u be the list of items rated by user u. The problem is to predict rating $R_{i,u}$ for unexpired stories (that are possibly new) for each user. If $y_{i,u}$ is a predicted rating, then the error is $e_{i,u} = y_{i,u} - R_{i,u}$. Our objective is to minimize Root Mean Square Error (RMSE) for all rated values (i, u) in T: $RMSE = \sqrt{\frac{1}{|T|}\sum_{(i,u)\in T} e_{i,u}^2}$

Given predictions for unobserved ratings in T, the application constructs a playlist for a user by selecting the highest rated stories for that user.

4 Models

We begin with a simple content-based model, then extend it to include collaborative filtering and latent variables.

4.1 Content-only model (NLR)

Given a historical ratings matrix T, our goal is to fit a regression model to produce a prediction for $y_{i,u}$, representing the rating given to item i by user u. We introduce a parameter vector α that represents the importance of each item feature in S_i, as well as a user-specific bias term β_u which represents the typical rating given by user u. Since the ratings are all in $[0, 1]$, we use the logistic function to ensure our predicted rating is in the same range. The resulting hypothesis is as follows:

$$y_{i,u} = \sigma(S_i\alpha + \beta_u)$$

To fit parameters, we minimize the KL-divergence between y and R using gradient descent, using L2 regularization, described in more detail in the next section. Note that this model uses no collaborative filtering. In the experiments below, we refer to this as the non-linear regression model (**NLR**); as an additional baseline, we also report results of linear version of this model (**REG**), which simply removes the logistic link function and uses least square error in cost function. Both of these methods can be considered standard text regression models (the real-valued counterpart to traditional text classification).

4.2 Non-linear Local Collective Embedding (NLLCE)

Matrix factorization is widely used in recommendation systems [Takacs *et al.*, 2008]. The idea of matrix factorization is to decompose a rating matrix R as a product of two smaller matrices representing clusters of items and users. Often, this is done without considering item attributes, relying solely the ratings. The main challenge of using matrix factorization in our domain is identifying latent factors for new stories, which have few or no ratings. In such cases, we wish to use the item attributes S_i to infer the latent factors for item i. While prior work has proposed fitting a separate model to S_i to predict the latent factors [Vandenoord *et al.*, 2013], it is desirable to combine these two steps into a single objective, as in Agarwal and Chen [2009]. Additionally, when the item feature vector has high-dimensionality, we may also want to jointly decompose S with the R matrix for better generalizability [Saveski and Mantrach, 2014]. Thus, our goal is to jointly decompose the ratings matrix R and the item feature matrix S, while simultaneously using the decomposition of the item feature matrix to inform the latent factors for items with few ratings. Our resulting model combines ideas from this prior work, while also extending them to the audio news setting.

We call the resulting approach non-linear local collective embedding (NLLCE), due to its relation to the Local Collective Embedding approach of Saveski and Mantrach [2014]. The hypothesis is:

$$y_{i,u} = \sigma(S_i\alpha + \beta_u + \mu + U_u^T V S_i^T)$$

where α and β_u are defined as in the prior section, μ is the overall rating mean, U is the latent factor matrix for users, with U_u as the row for user u, and V is the latent factor matrix that links the user factors with the item features S_i. It is V that allows us to make predictions for items with few ratings.

Since we have restricted both $R_{i,u}$ and $y_{i,u} \in [0, 1]$, we can construct an optimization problem to minimize the KL-divergence between R and y:

$$D_{KL}(R_{i,u}||y_{i,u}) = R_{i,u}\log\frac{R_{i,u}}{y_{i,u}} + (1 - R_{i,u})\log\frac{1 - R_{i,u}}{1 - y_{i,u}}$$

$$= C - R_{i,u}\log y_{i,u} - (1 - R_{i,u})\log(1 - y_{i,u})$$

where C is a constant. To reduce overfitting, we use L2 regularization, resulting in the following final objective:

$$J = -\sum_{(i,u)\in T} R_{i,u}\log y_{i,u} + (1 - R_{i,u})\log(1 - y_{i,u}) + \frac{\lambda}{2}(||\alpha||^2 + ||\beta||^2 + ||U||^2 + ||V||^2)$$

We perform gradient descent to optimize this objective. The gradients are as follows:

$$\frac{\partial J}{\partial V_h} = \sum_i S_i \sum_{u\in T_i} e_{i,u}U_{u,h} + \lambda V_h$$

$$\frac{\partial J}{\partial U_u} = \sum_{i\in T_u} e_{i,u} V {S_i}^T + \lambda U_u$$

$$\frac{\partial J}{\partial \alpha} = \sum_u \sum_{i\in T_u} e_{i,u}S_i + \lambda\alpha; \frac{\partial J}{\partial \beta_u} = \sum_{i\in T_u} e_{i,u} + \lambda\beta_u$$

The main advantages of non-linear LCE model over prior work is as follows:

- It does not treat missing values as zero ratings, as in LCE.
- It contains additional terms for the user bias (β_u) and the item popularity ($S_i\alpha$).
- It does not need a separate regression task to estimate latent factors.
- It uses a non-linear (sigmoid) kernel to account for the real-valued expected output.

4.3 Other baselines

In addition to the content-only models described above (**REG** and **NLR**), here we describe a number of competing baselines. While these are based on prior work, we have had to adapt each method to work with real-valued outputs, to handle cold-start problems, and we have also added terms to some models to handle item and user bias.

1. Baseline (BL): We use a baseline common in recommendation systems, which is simply a function of item bias b_i (the deviation between overall item ratings and μ), user bias b_u (the deviation between overall user ratings and μ) and overall rating average μ: $y_{i,u} = b_i + b_u + \mu$. For new item j, we use ridge regression to find b_j, where the independent variable is item feature vector S_i and the dependent variable is b_i.

2. Matrix Factorization with Regression (MFR): Inspired by Gantner *et al.* [2010], we perform a two-stage estimation, one to infer latent factors, and a second to predict the latent factors from item features. The hypothesis is:

$$y_{i,u} = \sigma(S_i\alpha + b_i + \beta_u + \mu + U_u^T M_i)$$

where U and M are latent factors for users and items, and b_j is an additional term for overall item popularity. As in NLLCE, we use a KL-divergence objective optimized with gradient descent. For new item i, we predict values for b_i and M_i using ridge regression fit on item attributes S in the training data.

3. Local Collective Embedding (LCE): As in **MFR**, we perform a two-stage estimation; however, latent factors for R and S are learned simultaneously, as in Saveski and Mantrach [2014]. The unified objective for factorization is:

$$J = c||S - M^T H||^2 + (1-c)||R - U^T M||^2$$

where M and H are latent factor matrices, and $c \in [0, 1]$ is a hyper-parameter that controls the importance of each factorization. After minimizing the cost function and finding coefficients (U, M, and H), again another regression task is required to predict the rating of a new story (as in **MFR**). Let $V = (HH^T)^{-1}H$, then for the the new item i with the feature vector S_i, the inferred rating for the user u can be computed as:

$$y_{i,u} = {U_u}^T V {S_i}^T = {U_u}^T (HH^T)^{-1} H {S_i}^T$$

In practice, this model has very poor results; we believe that is mainly because it assumes zero ratings for missing ratings. As a result, we do not include this model in the evaluation and in ensemble model.

4. Clustered Non-linear Regression (CNLR): The drawback of the content-only models (**REG**, **NLR**) is that they do not distinguish well between different types of users. This can result in poor accuracy for users with many historical ratings. CNLR attempts to address this by first clustering users, then introducing separate parameter vectors for each cluster. To do so, we compute the singular value decomposition of S, then map the ratings of a user into the resulting conceptual space:

$$S \approx U\Sigma V^T; R_u^* = \frac{1}{|T_u|} R_u U$$

where R_u^* is the map of user u's activities in the conceptual space, normalized by the number of ratings for the user. We next run k-means clustering on the R_u^* vectors to get user clusters. To find k clusters, we truncate the SVD to the top $3k$ singular values. Suppose $C(u)$ is the cluster of user u, then by adding separate coefficients for each cluster, our hypothesis becomes:

$$y_{i,u} = \sigma(S_i(\alpha + \alpha_{C(u)}) + \beta_u + \beta_{C(u)})$$

We again use gradient descent to minimize the KL-divergence of the true and predicted ratings. To deal with selecting k, we run the model with varying numbers of clusters and use the average for a final prediction.

5. Regression-based Latent Factor Models (RLFM). This is the approach of Agarwal and Chen [2009], which aims to deal with both cold and warm start. However, this model requires user attributes (such as demographics), which are not available in our data, so we simplified the model to work without user attributes. The main idea of this generative model is very similar to the **MFR** model, except that it assumes model error has a normal distribution. And then, with expectation maximization (EM) approach, it tries to estimate the error's variance. In fact, this model treats the error's variance as a regularization term (similar to λ), but it also estimates it in the maximization step. The hypothesis of simplified RLFM is:

$$y_{i,u} = \alpha_i + \beta_u + \mu + U_u^T M_i$$

In this model we assume that the error has a zero mean Gaussian distribution:

$$\begin{aligned} e_{i,u} =& y_{i,u} - R_{i,u} \sim N(0, \sigma) \\ \epsilon_i =& \alpha_i - S_i\gamma \sim N(0, b^2) \\ \delta_i =& M_i - S_i V_i \sim N(0, a^2 I) \end{aligned}$$

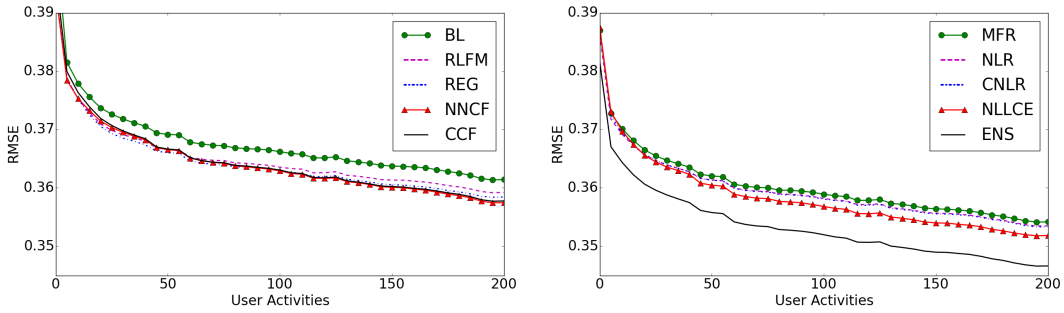
where δ_i has a multivariate Gaussian distribution. In the expectation step, we minimize the negative complete data log-likelihood:

$$\begin{aligned} -\log(P[y, \alpha, \beta, U, M | S, V, \gamma, \delta, \sigma, b, a]) = C \\ + \sum_{i,u} \frac{e_{i,u}^2}{2\sigma^2} + \frac{||\epsilon||^2}{2b^2} + \frac{||\delta||^2}{2a^2} \end{aligned}$$

where C is a constant. We start with initial guess for V, δ, σ, b, a and use the gradient descent method to find α, β, U, M. In the maximization step, the ridge regression can be used to update the V and the δ. Then, the variances is updated as following:

$$\sigma^2 \leftarrow \mathbb{E}[e_{i,u}^2]; b^2 \leftarrow \mathbb{E}[\alpha_i^2]; a^2 \leftarrow \mathbb{E}[M_i^T M_i]$$

6. Clustered Collaborative Filtering (CCF): We can use traditional collaborative filtering with clustering to deal with the cold-start problem. In this memory-based approach, we cluster stories with the k-means algorithm and take an average of ratings for each cluster for each user. Then we can apply the traditional collaborative filtering to find ratings for clusters that do not have rating for some users. We can use a heuristic by creating different number of clusters in each run to create multiple predictors, then we take the average of all these predictors to compute the final predictor.



(a) Linear and memory-based models.

(b) Non-linear models.

Figure 1: RMSE for all models as number of observations per user increases.

	User-cold		User-warm	
Item-cold	Size:	23%	Size:	25.7%
	ENS:	.4106	ENS:	.3518
	NLR:	.4142	NLLCE:	.3566
	MFR:	.4156	CNLR:	.3600
	CLNR:	.4167	NLR:	.3604
	NLLCE:	.4169	MFR:	.3613
Item-warm	Size:	27.1%	Size:	24.2%
	ENS:	.4109	ENS:	.3482
	NLR:	.413	NLR:	.3525
	MFR:	.4135	CNLR:	.3526
	CNLR:	.4160	MFR:	.3531
	NLLCE:	.4172	NLLCE:	.3533

Table 2: Comparison of models in different item-user and cold-warm quartiles. Users with fewer than 130 ratings are user-cold, and items with fewer than 30 ratings are item-cold.

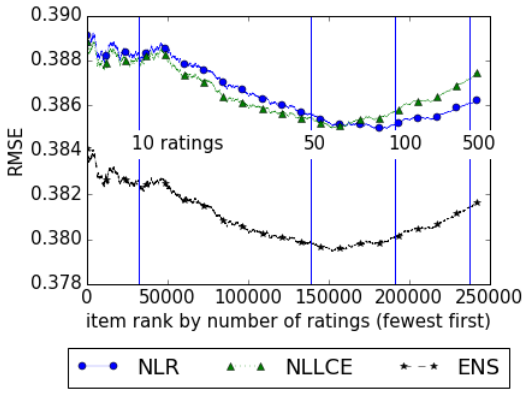


Figure 2: RMSE as the number of ratings per item increases.

7. Nearest Neighbor Collaborative Filtering (NNCF): Another way to deal with cold-start is to use the item-item similarity between the feature vectors instead of using the item-item similarity based on ratings. We use Latent Semantic Analysis (LSA) to reduce the dimension, and after the transformation to the lower dimension, Σ_i is the reduced dimension of S_i. To estimate the rating $R_{i,u}$, we can estimate it as the average ratings of all item j for user u, weighted by Pearson's correlation between the reduced dimensions: $Sim(i,j) = 1 + \text{Pearson}(\Sigma_i, \Sigma_j)$. The resulting hypothesis is:

$$y_{i,u} = \frac{1}{\sum_{j \in T_u} Sim(i,j)} \sum_{j \in T_u} Sim(i,j) R_{j,u}$$

We found better results by defining different thresholds to prune from the weighted average ratings with low similarities. We compute the average result over set of fixed thresholds.

4.4 Adding temporal dynamic

The user's taste can change over time. For example, users who prefer technology stories might at some point begin to prefer sports stories. In order to address this, we assume an exponential decaying window, and add it in the cost function. For example, for linear regression model, we can update cost function as following:

$$J(\alpha, \beta) = \frac{1}{2} \sum_{(i,u) \in T} e^{-ct(i,u)} e_{i,u}^2 + \frac{\lambda}{2}(||\alpha||^2 + ||\beta||^2)$$

Where c is a small number that control decaying strength, and the $t(i,u)$ is the seconds between training time, and when the user u listens to the story i. We add temporal dynamic to all model-based models (except **RLFM**).

4.5 Ensemble (ENS)

Given that we expect some models to work relatively better under different conditions (e.g., cold/warm items, cold/warm users), in the last approach, all of models are combined to create an ensemble model. To avoid overfitting, we run models with different hyper-parameter values, for example with 5, 10, 15, and 20 latent factors, and let the ensemble model find weights of each model. In this model, we first split the results of all models based on number of user activities in training. For example, we group the output of all models for users that have less than 5 activities together, and between 5 to 10 activities together. After splitting, we can use the ridge regression on each split, where the independent variables are the output of the models, and the dependent variable is the truth value of ratings. In our implementation, we use 26 total model variants to create the ensemble model.

5 Evaluation and results

We use a dataset from an existing mobile radio application. In our experiments, we use 975K activities from March 2nd to August 3rd, 2015, containing 51K users and 26K news stories. The term vectors are derived from over 1.8M tokens and 123K terms (unigrams and bigrams); we remove terms on a stop list as well as those appearing in fewer than 3 documents. In the evaluation process, to reflect the real-world setting, we train on prior activities to infer the next hour of activity. In practice, we find that using only 7 prior weeks' activities is sufficient. For example, for testing on activities on July 1st between 10:00am and 11:00am, we train on the data from May 14th, 10:00am to July 1st, 10:00am. We run 26 model variants for activities from April 20th to June 7th, and save the model predictions. Then we use this output to train the ensemble model on activities from June 8th to August 3rd.

Figures 1(a) and Figure 1(b) compare results of different models on the test set. In these figures, the y-axis is RMSE and the x-axis is the the number of user ratings in the training set. As a result of user cold-start, the RMSE is higher when we have few user ratings, and it gradually decreases by increasing number of ratings.

Figure 1(a) compares linear and memory-based models. As expected, the baseline has highest error. According to this plot, memory-based models (CCF and NNCF) have slightly higher error than linear models (REG and RLFM) in the user cold-start state, but their error is lower in the user warm-start state.

Figure 1(b) compares non-linear models. According to this plot, non-linear models have lower error than both linear models and memory-based models. MFR has higher error than other non-linear models. The difference between the NLR and the CLNR models is not significant, and the clustering does decrease the error of NLR model (most likely because we do not have user demographics), but can generate playlists that are more customized to the specific user. Our proposed NLLCE method has the lowest error. In addition, as expected, the ensemble model significantly decreases the error compare to all other models. Thus, it appears that the ensemble approach is able to combine the best of each model.

In the next experiments, we followed prior work to partition the user-item space into four roughly equal quartiles [Park and Chu, 2009]. To create quartiles, we classify users that rate fewer than 130 items as the user-cold set, and remaining users as the user-warm set. Similarly, we classify items that have been rated by fewer than 30 users as the item-cold set, and remaining items as the item-warm set. Table 2 shows results of the top 5 models for each quartile. The ENS model is the top model in all quartiles.

According this table, the NLR model, which predicts almost the same playlist for all users, is the best stand-alone model for the user cold-start state. This makes sense, since collaborative filtering has little information to work with for cold users. For the item cold-start state, NLLCE works much better with the user warm-start state. In addition, this table shows that dealing with the user cold-start state is much harder than the item cold-start state (which we again attribute to lack of user attributes). When we have warm-start for both users and items, NLR and NLLCE have similar results. Thus, we find NLLCE to be the best model for users with at least 130 observations, and NLR to be the best model for users with fewer than 130 observations.

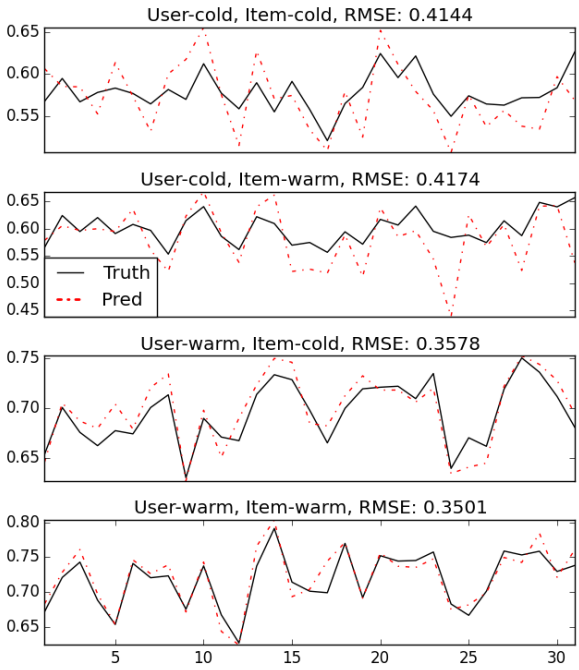


Figure 3: Warm and cold start comparison for the ensemble model over one month (x-axis is days, y-axis is the daily average of ratings).

To investigate item cold-start, Figure 2 plots RMSE as the number of ratings per item increases. We sort all predicted ratings by the number of ratings the target item has at prediction time. We then compute a sliding average RMSE, using a window of size 200. From this figure, we can see that most items have few ratings (the median is 31 ratings). Moreover, NLLCE has lower error than NLR up to items with around 50 ratings. At this point, we suspect that NLLCE begins to overfit in the collaborative filtering component.

Finally, Figure 3 compares ensemble prediction in chronological order for the different quartiles for July 2015. The x-axis shows the day of the month, the solid line shows the daily average of the truth ratings, and the dash line show the average of the ensemble model prediction for each day. According to this figure, while the ensemble model has some error in the user cold-start states, the error is much lower in the user warm-start states. In addition, the results of the item warm-start state is slightly better than the the item cold-start state, but the difference is small. This shows again that the model can handle the item cold-start challenge.

6 Conclusion and future work

In this paper, we propose several models to address the cold-start in recommendation systems. Our results suggest that the Non-Linear Local Collective Embedding (NLLCE) model has lower error than other models in the item cold-start states.

In addition, by simplifying the NLLCE model, we create the Non-Linear Regression (NLR) model that has better results in the user cold-start states. Finally, we find that blending different models with the ensemble method can significantly improve results in both the cold-start and the warm-start for users and items. In the future, we will investigate the effect of adding social media and demographic attributes of users, and create new models that deal better with the user cold-start state.

References

[Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[Agarwal and Chen, 2009] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[Bennett and Lanning, 2007] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[Breese *et al.*, 1998] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[Claypool *et al.*, 1999] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper, 1999.

[Das *et al.*, 2007] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[Gantner *et al.*, 2010] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 176–185, Dec 2010.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE, 2008.

[Johnson, 2014] Christopher C Johnson. Logistic matrix factorization for implicit feedback data. *NIPS*, 27, 2014.

[Liu *et al.*, 2010] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pages 31–40, New York, NY, USA, 2010. ACM.

[Mcfee *et al.*, 2012] Brian Mcfee, Student Member, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE TASLP*, 2012.

[Park and Chu, 2009] Seung-Taek Park and Wei Chu. Pairwise preference regression for cold-start recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 21–28, New York, NY, USA, 2009. ACM.

[Rendle and Schmidt-Thieme, 2008] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 251–258, New York, NY, USA, 2008. ACM.

[Royston *et al.*, 2006] Patrick Royston, Douglas G Altman, and Willi Sauerbrei. Dichotomizing continuous predictors in multiple regression: a bad idea. *Statistics in medicine*, 25(1):127–141, 2006.

[Salakhutdinov and Mnih, 2008] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[Saveski and Mantrach, 2014] Martin Saveski and Amin Mantrach. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 89–96, New York, NY, USA, 2014. ACM.

[Schein *et al.*, 2002] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[Takacs *et al.*, 2008] G. Takacs, I. Pilaszy, B. Nemeth, and Domonkos Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference on*, pages 553–562, Dec 2008.

[Vandenoord *et al.*, 2013] Aaron Vandenoord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.

[Zhang *et al.*, 2006] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *IN PROC. OF THE 6TH SIAM CONFERENCE ON DATA MINING (SDM*, pages 549–553, 2006.