Movie Recommendation based on Collaborative Topic Modeling

Abhishek Bhowmick
Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
abhowmi1

Udbhav Prasad
Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
udbhavp

Satwik Kottur
Dept. of Electrical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
skottur

Abstract

Traditional collaborative filtering relies on ratings provided by viewers in the movie-watching community to make recommendations to the user. In this project, we attempt to combine this approach with probabilistic topic modeling techniques to make recommendations that consist not only of movies that are popular in the community, but also those that are similar in content to movies that the user has enjoyed in the past.

1 Introduction

Recommender systems are an important technology for TV/movie streaming services like Netflix, HBO, audio/music streaming sites like Spotify, Pandora, news article feeds like Pulse, online retailers such as Amazon, Walmart etc. Indeed, any service provider or content management system that has large quantities of information (or the ability to extract such information) such as usage patterns, browsing and click history, natural text descriptions etc can and should make use of recommendation methods to help find items of interest. Among various information sources, data in the form of natural text is a particularly rich and expressive source of information, however it is highly unstructured in general. Topic models are used to extract latent structures from large volumes of unlabeled text, that can be used for analysis of content and in turn, aid end goals such as making recommendations. In particular, textual information such as movie plot summaries can be very helpful to improve the prediction performance of traditional collaborative filtering. In the remaining part of this report, we limit ourselves to the study of how topic modeling of large text corpora can help in the task of movie recommendation, however most of the discussion/analysis can be applied to other domains.

1.1 Collaborative Filtering and its shortcoming

Traditional collaborative filtering makes use of interactions between users and items. They may be broadly classified into two categories - neighbourhood methods and latent factor models. Neighbourhood models explicitly capture relationships between items (or users) and predict a user's liking for a particular item based on ratings of neighbouring items by the same user. The other approach, latent factor models, directly characterize both users and items by latent factors. We focus on factor based models as they are more accurate than neighbourhood based methods. However, all collaborative filtering methods suffer from the 'cold start' problem, that is, they are unable to recommend movies in the absence of rating patterns. In fact, in the domain of music, it has been observed [1] that the distribution of available rating information for music artists has a very long tail, which means that most of the music items have little rating data available. We believe the same is true of movies as well and hence, would like to be able to recommend movies that are in this long tail.

1.2 Content Based Recommendation

Content-based recommendation addresses the 'cold start' problem associated with collaborative filtering, where certain items do not have any rating information and hence the corresponding item vectors consist of all zeroes (we use zeroes to represent missing ratings in the rating matrix). One approach is to use topic modeling of plot summaries to identify latent themes/topics. We can learn topic representations for each item (a vector of topic proportions) and add them to the item vectors in the latent-factor model. Such topic representations of movie items are also useful outside the domain of movie recommendation. Interpretability of topics may help in explaining recommendations to users, effective content programming and ad targeting based on user profiles [2].

2 Problem Definition

Briefly, the problem we are trying to solve is predict how highly a user will rate certain movies based on all users' rating histories and plot summaries for all movies. Making use of these predicted ratings, we come up with movie recommendations for a user. The problem can be formalized as follows [3] :

We are given a list of users *U* = $\{u_1, u_2...u_m\}$ and a list of items *V* = $\{v_1, v_2...v_n\}$, where each user u_i has a list of items I_{u_i} which he/she has given ratings for. For a given user $u_a \in U$, we need to solve the following two tasks:

Prediction: Estimate the predicted rating P_{a_j} of an item $v_j \notin V_{u_a}$. The prediction task can be further split into two types: *in-matrix prediciton* and *out-of-matrix prediction*. *In-matrix prediction* is the problem of predicting ratings for items that have already been rated by atleast twenty other users, whereas *out-of-matrix prediction* makes predictions about those items that have very few or no ratings (less than 20).

Recommendation: Return a list of N items $I_r \in I \ \& \ I_r \cap I_a = \phi$, that the user will like most. This is simply a problem of returning the items with highest predicted rating values.

Specifically, we are interested in observing how incorporation of item topic representations increases the prediction accuracy of factor models. We would also like to analyze the interpretability of the latent topics that are captured by the topic model, however this will be just be a qualitative analysis.

3 Proposed Method

We use Probabilistic Matrix Factorization (PMF) for collaborative filtering on movie ratings and Latent Dirichlet Allocation (LDA) for topic modeling of the corpus of movie plot summaries. We then combine the latent factor model learned through PMF and the topic model learned through LDA into a single collaborative topic regression model (CTR). A CTR model essentially uses the latent topic space (latent variables) to explain observed ratings and observed documents (observed variables), thus incorporating content information into a collaborative filtering framework [4].

3.1 Probabilistic Matrix Factorization

One of the most popular methods approaches to collaborative filtering is based on low-dimensional factor models. The idea behind such models is that the preferences of a user are based on a small number of unobserved factors. For example, if there are N users and M movies, the $N \times M$ preference matrix R is given by the product of an $N \times D$ user coefficient matrix U^T and a $D \times M$ factor matrix V. Training such a model amounts to finding the best rank-D approximation of the observed $N \times M$ target matrix R under a given loss function.

We adopt a probabilistic linear model with Gaussian observation noise. The graphical model is shown in figure 1. We define the conditional probability distribution over the observed ratings as

$$p(R \mid U, V) = \prod_{i=1}^{N} \prod_{j=1}^{M} [\mathcal{N}(R_{ij} \mid U_i^T V_j, \sigma^2)]^{I_{ij}} \tag{1}$$

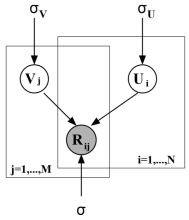


Figure 1: Graphical model for Probabilistic Matrix Factorization

where $\mathcal{N}(x \mid \mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is the indicator function that is equal to 1 if the user i rated movie j, and is 0 otherwise. We also place zero-mean spherical Gaussian priors on movie and user feature vectors:

$$p(U \mid \sigma_U^2) = \prod_{i=1}^{N} \mathcal{N}(U_i \mid 0, \sigma_U^2 I) \tag{2}$$

$$p(V \mid \sigma_V^2) = \prod_{j=1}^{M} \mathcal{N}(U_i \mid 0, \sigma_V^2 I) \tag{3}$$

Maximizing the log-posterior over movie and user feature vectors with hyper-parameters (the observation noise variances and prior variances) kept fixed is equivalent to minimizing the sum-of-squared errors objective function with quadratic regularization terms.

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2} ||U_i||^2 + \frac{\lambda}{2} ||V_j||^2 \tag{4}$$

where $\lambda_U = \sigma^2 / \sigma_U^2$ and $\lambda_V = \sigma^2 / \sigma_V^2$. A local minimum of the objective function can be found using gradient descent in U and V.

3.1.1 Gradient descent

Since we have more than 10 million ratings, we use stochastic gradient descent as our learning algorithm. For each given training case, the algorithm predicts R_{ij} and computes the associated prediction error:

$$e_{ij} = R_{ij} - U_i^T V_j \tag{5}$$

It then modifies the model parameters by a magnitude proportional to γ (the "learning rate") in the opposite direction of the gradient, yielding

$$U_i \leftarrow U_i + \gamma (e_{ij} V_j - \lambda_U U_i) \tag{6}$$

$$V_j \leftarrow V_j + \gamma (e_{ij} U_i - \lambda_V V_j) \tag{7}$$

Additionally, instead of taking each rating one by one, we took them in chunks of 100 ratings at a time, and calculated the updates over all those 100 ratings. This made the algorithm faster, and also made the updates more stable, and immune to any outlier gradient values. Convergence check was done by taking the average over a fixed number of gradients from previous iterations. If the average was smaller than a particular threshold, we considered the algorithm to have converged. The algorithm generally converged in less than 3 iterations over all the ratings. The learning rate was chosen to be an function that decayed with the number of iterations. In particular, the learning rate was formulated as :

$$\gamma = (\tau + i)^{-\kappa} \tag{8}$$

where τ is for normalizing, κ is the "forgetting" rate, and i is the iteration number.

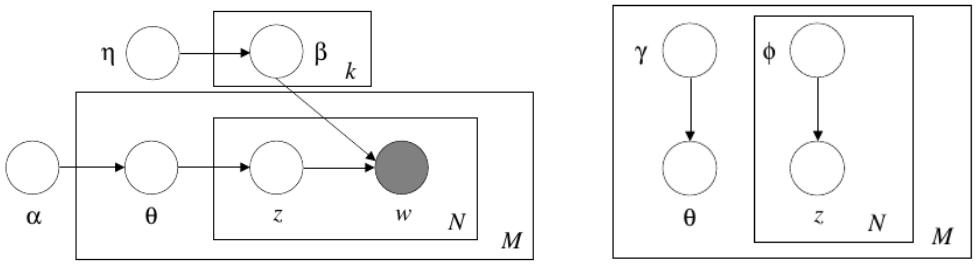
3.2 Latent Dirichlet Allocation

For a collection of text documents, a topic modeling algorithm extracts a set of topics, where each topic is a distribution over words that occur in the documents. Words belonging to a topic are biased around a single theme. The topic model that we use for representation of documents is Latent Dirichlet Allocation (LDA), which is a generative probabilistic graphical model for collections of discrete data [5]. Each document is modeled as a finite mixture over a set of underlying topics.

LDA has the underlying assumptions that the words in a document and documents in a corpus are exchangeable - i.e., the specific ordering of words and documents can be neglected. De Finetti's representation theorem states that a collection of infinitely exhangeable random variables are conditionally independent and identically distributed, if they are conditioned on a random parameter that is drawn from some probability distribution. Now, the generative process of the LDA model is:

initialize vocabulary from corpus, the size of which is V;
for *each of the K topics k* **do**
    Choose $\beta_{k,1:V} \sim$ Exhangeable Dirichlet(η) // Draw topic distributions;
end
for *each of the M documents* **w** *in corpus D* **do**
    Choose $\theta \sim$ Dirichlet(α);
    for *each of the N words* w_n *in document* **w do**
        Choose a topic $z_n \sim$ Multinomial(θ);
        Choose word w_n from $p(w_n|z_n, \beta_{z_n})$, a multinomial probability conditioned on topic z_n
    end
end

Algorithm 1: Generative process for LDA



(a) True posterior distribution

(b) Variational posterior distribution

Figure 2: Graphical models of LDA, before and after variational approximation

Figure 2a shows the graphical model representation of LDA, which is a three-level hierarchical model. We assume that the number of topic vectors k is fixed and the vocabulary size of the corpus to be modeled is V. The word probabilities are parametrized by a k X V random matrix β, each row of which represents the distribution of topics over words in the vocabulary. Each row of β is independently drawn from an exchangeable Dirichlet distribution with parameter η. Also, α is a k-dimensional vector which is a parameter for the Dirichlet random variable θ. Given the parameters α and β (which itself is a random matrix parametrized by η), the joint distribution of the topic mixture θ, the set of N topics **z** and the set of N words **w** is:

$$p(\theta, \mathbf{w}, \mathbf{z}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{9}$$

We make use of the representation theorem which states that the set of topics $\mathbf{z}$ are independent conditioned on θ which is a random parameter of a multinomial distribution. Next, we describe two important tasks for LDA, namely inference and estimation:

Inference: The inference task is to compute the posterior distribution of the hidden variables given the observed variable $\mathbf{w}$ (the document), assuming we know the parameters α and β. (Note that we treat β as a fixed parameter for the following discussion)

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)} \tag{10}$$

Computing this distribution is intractable and hence we use variational approximate inference, as described by Blei et al. [5]. Simple modifications to the LDA graphical model such as dropping edges between θ, $\mathbf{z}$ and $\mathbf{w}$ and adding variational parameters lead us to the variational model in Figure 2b, which has the following variational distribution:

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \tag{11}$$

The optimal values of the variational parameters (γ^*, ϕ^*) are obtained by minimizing the Kullback-Leibler (KL) divergence between the variational and true posterior distribution. By placing a Dirichlet prior on β, we get a separable variational distribution, which yields the same expressions for γ^*, ϕ^* and introduces a new variational parameter λ which has a similar expression as γ.

Estimation: We wish to find parameters α and η that maximize the log-likelihood of observed data, however computing the likelihood is intractable. So, we use a variational EM procedure in which we alternatingly maximize a lower bound on the log-likelihood of data with respect to variational parameters γ, ϕ and λ. This is the E-step. We then maximize this lower bound with respect to the parameters α and η, which comprises the M-step. The updates for both the parameters α and η are obtained using an efficient Newton-Raphson method in which the Hessian is inverted in linear time.

3.3 Collaborative Topic Regression

The Collaborative Topic Regression (CTR) model combines a topic model such as LDA with traditional collaborative filtering [4]. This is done by combining both the latent topic vector and observed ratings to describe the item latent vector in the factor model of Section 3.1. The graphical model and generative process of CTR are given below:

Parameter Estimation: First, we train our LDA implementation on a separate training corpus and learn the model parameters α and β. Now, the log likelihood of the data is given by:

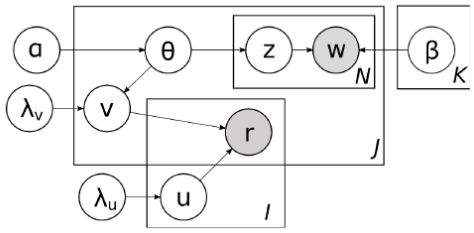
$$\mathcal{L} = \frac{-\lambda_u}{2} \sum_i u_i^T u_i - \frac{\lambda_v}{2} \sum_j (v_j - \theta_j)^T (v_j - \theta_j) + \sum_j \sum_n log(\sum_k \theta_{jk} \beta_{k,w_{jn}}) - \sum_{i,j} \frac{1}{2\sigma^2} (r_{ij} - u_i^T v_j)^2$$

Computing the full posterior of u_i, v_j and θ_j given parameter β is intractable. Our approach is to maximize the likelihood function by co-ordinate ascent, iteratively optimizing the collaborative filterings u_i, v_j and topic proportions θ_j. Given topic estimate θ_j, setting the gradient of the log likelihood with respect to u_i and v_j equal to zero gives us closed form update rules for the collaborative filtering variables. Similarly, in the next phase, we can optimize the topic estimate θ_j given u_i and v_j. However, similar to the approach taken by Blei et al. [4], we simply set θ_j equal to the topic estimate obtained from our LDA implementation, to save computation time without significant performance loss.[1]

[1]We use the movies for which we have both ratings and plot summaries to learn the optimal parameters $u^*_{i=1:I}$, $v^*_{j=1:J}$, $\theta^*_{1:J}$, β^*.

for *each user i* **do**
  Draw user latent vector $u_i \sim \mathcal{N}(0, \sigma^2 \lambda_u^{-1} I_K)$;
end
for *each item j* **do**
  Draw topic proportions $\theta_j \sim$ Dirichlet(α);
  Draw item latent offset $\epsilon_j \sim \mathcal{N}(0, \sigma^2 \lambda_v^{-1} I_K)$;
  Set item latent vector as $v_j = \epsilon_j + \theta_j$;
  for *each word* w_{jn} **do**
    Draw a topic assignment $z_{jn} \sim$ Mult(θ);
    Draw word $w_{jn} \sim$ Multinomial($\beta_{z_{jn}}$);
  end
end
for *each user-item pair (i,j)* **do**
  Draw the rating $r_{ij} \sim \mathcal{N}(u_i^T v_j, \sigma^2 I)$;
end

(a) Generative process



(b) Graphical model

Figure 3: Generative process and model of CTR

Prediction : We then use the learned parameters for prediction of movie ratings. For in-matrix prediction, we use the following approximation:

$$r_{ij}^* \approx (u_i^*)^T (\theta_j^* + \epsilon_j^*) = (u_i^*)^T v_j^* \tag{12}$$

For, out-of-matrix prediction, where the movie has no ratings available, we use the following approximation:

$$r_{ij}^* \approx (u_i^*)^T (\theta_j^*) \tag{13}$$

4 Experiments

We present the results of our evaluations and analysis in this section. We denote the plain collaborative filtering model as CF ($v_j = \epsilon_j$), the collaborative topic regression model as CTR ($v_j = \epsilon_j + \theta_j$) and the prediction model with just topic estimates as CTR-LDA ($v_j = \theta_j$), where v_j, ϵ_j and θ_j are the latent vector, offset vector and topic estimate vector respectively for item j.

4.1 Dataset

For Collaborative Filtering, we use the MovieLens 10M dataset [2], which is a collection of 10 million movie ratings on 10,000 movies by 72,000 users. The dataset has been pre-processed so that each user has rated at least 20 movies. This makes the rating matrix very sparse (98.6% sparse). The data is very well structured and fits into memory. The range of the rating values is 1 - 5, in steps of 0.5.

We use the CMU Movie Summary Corpus[3] for generative topic modeling of the movie summaries. This corpus has plot summaries for 42,306 movies and associated metadata such as genre, year of release, cast etc. Each movie is indexed by a Wikipedia Movie ID. We use only the plot summary text and none of the other metadata.

The datasets are used in the following manner: we first find the set of movies for which we have both plot summaries and user ratings - this common subset has approximately 5000 movies in total. Out of these, we choose 4400 movies and collect their ratings, 80% of which we use for training our CTR and CTR-LDA models and 20% for in-matrix prediction test. We use the remaining 600 movies from the common subset for out-matrix testing of CTR and CTR-LDA. We separately train the topic model using nearly 37,000 movie summaries for which we do not have any user ratings.

[2] http://grouplens.org/datasets/movielens/

[3] http://www.ark.cs.cmu.edu/personas/

Since the movie summaries are in the form of natural text, some preprocessing steps were necessary - namely tokenizing, upper-case to lower-case conversion, punctuation and stop-word removal and stemming. We used the python NLTK toolkit [6] for this. We extracted all words from the processed summaries and built our vocabulary after sorting them lexicographically. Each document was then represented as a vector of integers, each integer being an index into the vocabulary. For ease of combining the summary and rating datasets, we created a map from the Wikipedia IDs to the Movielens IDs (by linking movie IDs that have the same movie names).

4.2 Evaluation of LDA implementation

Interpretability of latent topics discovered by a topic model can be evaluated using two human evaluation tasks namely, *word intrusion* and *topic intrusion*, proposed by Chang, Blei et al [7]. These tasks evaluate the quality of topics inferred by the model and the assignment of topics to documents. We briefly describe these tests and evaluate our LDA using similar notions below:

4.2.1 Word Intrusion

This test measures whether the topics inferred by the topic model correspond to natural interpretations and contain words that are semantically close. An 'intruder' is defined as a word that doesn't belong with the the others in a group. The original task proposed in [7] consists of building a set of words from a randomly chosen topic and injecting an 'intruder'from some other topic. This set is then presented to a human subject to see if the 'intruder' word is correctly identified by the subject. For our evaluation, we just observe the proportion of 'intruder' words per topic. In the following table, we list out some of the topics from our 15-topic LDA (each topic is color-coded).

Topic	Terms after stemming	Intruders
1	film, movi, stori, play, show, perform, charact, scene, music, star, follow,includ, peopl, around, end, song, first, band, act, role	follow, around, end, includ, first
2	polic, kill, murder, offic, prison, arrestm investig, man, gang, case, death, crime, one, drug, shoot, killer, crimin, detect, escap, suspect	offic, one, man
3	war, american, forc, state, soldier, armi, unit, order, offic, german, govern, british, world, command, general, men, captain, group, agent, militari	men
4	tell, go, ask, see, say, leav, back, day, get, call, goe, come, home, next, want, find, night, one, know, time	tell, go, ask ...
5	use, ship, destroy, island, human, earth, crew, attack, discov, world, control, escap, power, one, alien, monster, caus, find, time, rescu	use, one, caus, find
6	king, power, kill, find, use, one, evil, return, take, villag, save, princ, magic, howev, queen, order, fight, name, world, death	find, use, one, take, howev

Table 1: Topics identified by LDA and 'intruders' per topic

We see that our LDA is able to identify important movie 'themes' such as theatre/drama (topic 1), crime/violence (topic 2), war movies (topic 3), adventure/fiction (topic 5), history/drama (topic 6). These topics have a few number of 'intruders' (3 per topic on average). However, there are certain topics like topic 4, that are too general to be useful, such topics have a large number of 'intruders'.

4.2.2 Topic Intrusion

The topic intrusion test measures how well a topic model assigns topics to documents. Similar to the word intrusion test, the topic intrusion test consists of building a set of topics that are assigned to a particular document along with an 'intruder' topic. This set of topics is presented along with the document to a human subject to see if the 'intruder' topic is correctly identified by the subject. For our evaluation, we present a sample processed movie plot summary and highlight the topic mixture of this document by coloring the words that belong to the topics assigned by LDA.

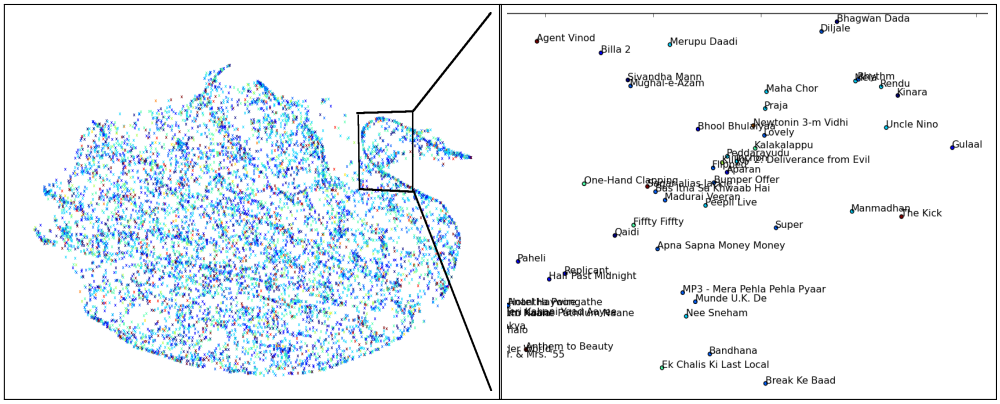
film courage struck reckless freewheel arthur show round table symbol life service brotherhood guinevere subsequent kidnap malagant men shoot crossbow battle malagant soldier citizen ensure lancelot malagant face disarm lancelot seize arthur bodies float sea set aflame

Figure 4: Topic distribution for summary of 'First Knight' (1995)

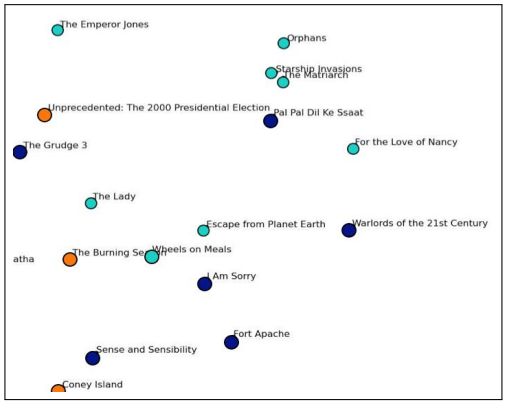
We see that the summary for the movie 'First Knight' contains a large proportion of words belonging to topic 6 - history/drama(brown). The other significant topics contained in this are topic 2 - crime/violence (green) and topic 3 - war movies (cyan). We thus see that our LDA correctly identifies the topics in the plot summary of 'First Knight', a historical period war drama based on the life of King Arthur.

4.2.3 Visualization of movie corpus

LDA provides feature vectors for movie plot summaries in the form of topic proportions, and models the latent structure in the items belonging to the corpus. We try to identify this latent structure by visualizing the corpus using t-SNE, a dimensionality reduction tool that is used for the visualization of high dimensional datasets [8].



(a) 2D plot of feature representations of movies (b) Indian movies are clustered together



(c) Movies similar in content spaced together

Figure 5: t-SNE visualizations

Figure 5a is a plot of the feature representations of the movies when mapped to 2D using t-SNE. Colors indicate the genres of the movies. From the plot, we can observe that similar genres (colors) aren't necessarily clustered together, implying that there is no on-to-one mapping from the topic representation of a movie to its genre. However, we do observe some structure inferred from the corpus of plot summaries. For instance, all Indian movies are clustered in the boxed region of Figure 5a, a close-up view of this region is shown in Figure 5b. Similarly, movies that have the same color (genre) and are clustered together usually tend to have very similar content. For instance, as seen in

Figure 5c, the movies 'Coney Island', 'Burning Season' and 'Unprecedented: The 2000 Presidential Elections' all have the same color (orange) and are placed close by, indicating that they should have similar content. A study of their plot summaries indicates that these are all documentaries based on various issues in America. Similar observations are recorded in other parts of the graph as well.

4.3 Prediction of Ratings

Topics	Train Error				Test Error			
	CF	CTR-LDA-in	CTR-in	CTR-out	CF	CTR-LDA-in	CTR-in	CTR-out
15	0.915	0.912	0.912	0.912	1.140	1.370	1.029	1.167
25	0.915	0.913	0.913	0.912	1.140	1.303	1.039	1.136
40	0.914	0.914	0.913	0.913	1.138	1.271	1.052	1.148

Table 2: Train and test error of different models for different number of latent topics

Table 2 lists the train and test errors of the different models as we vary the number of topics in our topic model. On average, for any number of topics, we observe that in-matrix prediction of the CTR model does better than the CF model. This is expected as the CTR model incorporates content analysis along with rating information, whereas the CF model relies on user rating data alone. We get a reduction of nearly 9.7% in prediction rating RMSE test error, when the number of topics is 15. As the number of topics increases, we find that the test RMSE error of CTR-in slightly increases, hence we fix the final number of topics to 15 for CTR for further evaluations. Also, we evaluate the out-of-matrix prediction of CTR by removing certain ratings from the rating matrix, remembering them as true ratings, computing the predicted rating values using CTR and comparing them against the true values. We observe that CTR is able to perform out-of-matrix prediction with almost as good accuracy as that of CF in-matrix predictions (note that out-of-matrix prediction is a feature of CTR that CF does not possess). CTR-LDA-in gives poorer prediction than CF, implying that just content-based prediction performs worse than ratings-based predictions (we do not list the values for CTR-LDA-out as it performs nearly the same as CTR-LDA-in).

Figure 6 plots the performance of the various models as a function of λ_v, which is a collabrative filtering variable in CTR. In the above plots, the blue curve is the train error while the green curve represents the test error. We observe that

4.4 Recommendation

5 Conclusions

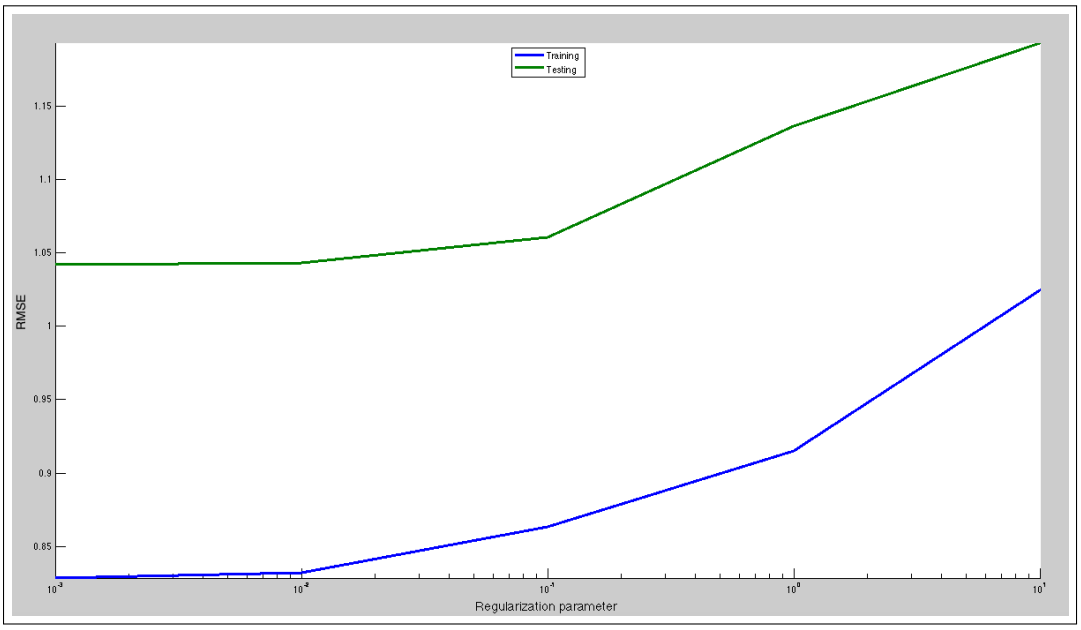
We used a Collaborative Topic Regression model, which combines topic modeling with traditional collaborative filtering, to recommend movies based on both item content and user ratings from the community. We implement our own version of Latent Dirichlet Allocation for topic modeling of the corpus of movie summaries and combine . We see that the CTR model

References

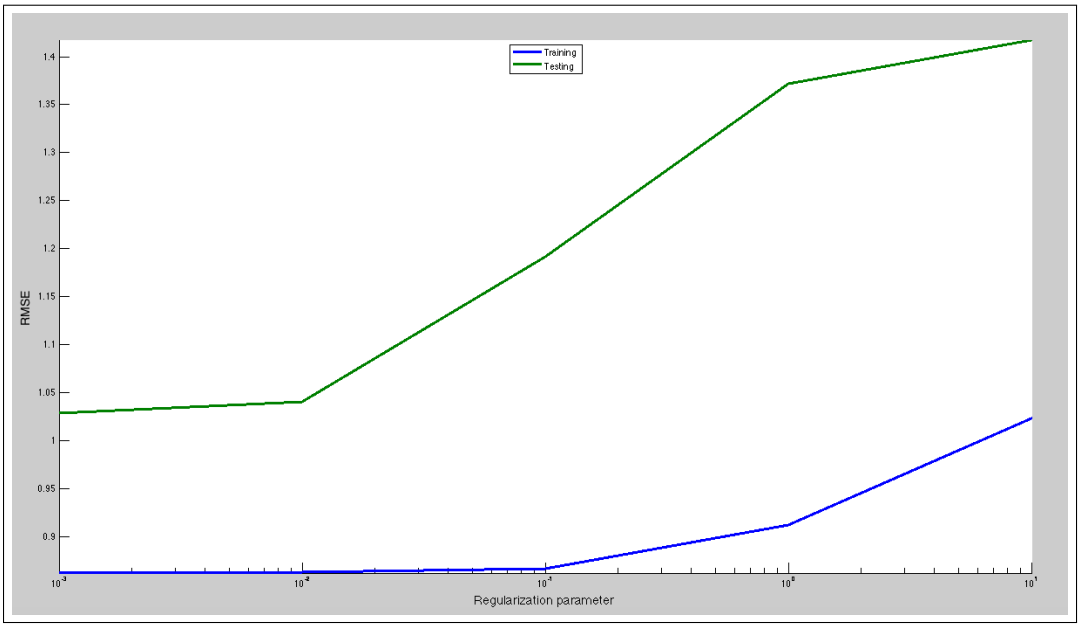
[1] O. Celma, "Music recommendation and recovery in the long tail," *PhD thesis, Universitat Pompeu Fabra, Barcelona*, 2008.

[2] D. Agarwal and B.-C. Chen, "flda: matrix factorization through latent dirichlet allocation," in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 91–100, ACM, 2010.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, ACM, 2001.



(a) Vanilla Collaborative filtering



(b) Vanilla Collaborative filtering

Figure 6: Regularization and RMSE errors for training and testing

[4] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 448–456, ACM, 2011.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[6] "Python Natural Language Toolkit." `http://www.nltk.org/`.

[7] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *Advances in neural information processing systems*, pp. 288–296, 2009.

[8] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.