Rating Prediction for Amazon Movies

A Comparison of Text-Mining and Recommender System Algorithms

Rajiv Pasricha, A11286283

December 1, 2015

1 Introduction

With the proliferation of ecommerce, social, and media consumption websites in recent years, the task of providing accurate recommendations has become essential to maintaining user engagement and improving user satisfaction. There are a wide variety of recommender system algorithms intended to solve these tasks, including content-based, collaborative filtering, and latent factor algorithms. While these algorithms perform well given adequate data and appropriate rating density, their performance often suffers on smaller and sparser datasets.

This is known as the cold-start problem for recommender systems, and there have been many recommendation algorithms proposed to address this issue. One potential approach is to utilize the text of the review along with the observed ratings. This takes advantage of text-mining techniques to extract user sentiment from relatively few observations, resulting in improved rating predictions for new users and items.

In this project, we analyze a dataset consisting of 50,000 movie reviews on Amazon. The dataset includes the observed rating on a 1 - 5 scale, along with the text of the user's review. As the dataset is quite sparse, these reviews prove to be valuable in predicting unobserved ratings.

We train two supervised learning algorithms on the data, linear regression and logistic regression. These algorithms used a bag-of-words model of the observed reviews to predict movie ratings. We also train a latent factor recommender system on the data, to predict ratings based on the observed user and item rating behavior.

After analyzing the results of these algorithms, we se that the simple linear regression model outperforms all other models, including the latent factor algorithms which are optimized for this particular task. This matches our expectations, as we would expect textual reviews to provide richer and more predictive features than the observed ratings, especially given the size and sparsity of our dataset.

In Section 2, we present relevant information and statistics about the Amazon movies dataset. We describe the predictive tasks and evaluation metrics we used in Section 3, and the corresponding models in Section 4. Section 5 discusses the results of our algorithms on the Amazon movies data. Finally, we present related literature in Section 6, and conclude in Section 7.

2 Data

In this section, we describe the Amazon movies dataset, the dataset used for our predictive tasks. This dataset was collected from the Stanford Network Analysis Project (SNAP) dataset repository [4]. The original dataset consists of 7,911,684 reviews, between 889,176 users and 253,059 items. For this project, we used the first 50,000 reviews from this dataset, for faster and more efficient computation and to analyze the performance of our algorithms on a small, sparse review dataset.

Each example in the dataset contains the following information:

- User ID and Username
- Product ID
- Rating

- Review Helpfulness
- Review Time
- Review Text

2.1 Relevant Statistics

The subset of the movies used for this project has a total of 50,000 reviews, written by 36409 users for 1539 items. Some statistics about the observed reviews are summarized in Table 1.

Number of 1-star reviews	3868
Number of 2-star reviews	2746
Number of 3-star reviews	4922
Number of 4-star reviews	10507
Number of 5-star reviews	27957
Average review length (words)	168.51
Average reviews per user	1.373
Average reviews per item	32.489

Table 1: Movies Dataset Statistics

From the data above, we see that the data is highly biased towards 4 and 5 star reviews. This is expected as intuitively, people are more likely to provide feedback for an item if they feel positively about it.

However, this unbalanced dataset has the potential to bias our models towards predicting high ratings. For example, a naive algorithm that always predicts a rating of 4 or 5 stars would achieve a prediction accuracy of 0.769. In order to improve prediction performance, two possible approaches would be to train models with evenly balanced training sets, or to assign weights such that all rating values have approximately equal weight.

Also, we see from Table 1 that the average review length of the reviews in the dataset is approximately 170 words. In Figure 1, we include a histogram of the number of words in each review. We see that the highest number of movie reviews have lengths around 20 - 30 words. In particular, the count increases from 0 to 30 words, and then slowly declines as the number of words increases. However, there are enough reviews with large numbers of words to significantly influence the mean, shifting it up to a value of around 170 words. If the average review has this length, this means that the reviews should be long enough to extract meaningful information from the words that were used, enabling successful rating prediction using the review text.

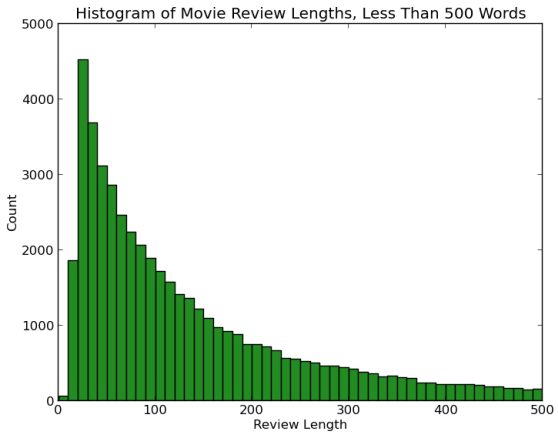

Figure 1: Movie Review Lengths

Figures 2 and 3 show the number of reviews written by each user, and for each item, respectively. For figure 3, the x-axis was truncated to 20 reviews for ease of visualization. From these figures, we can see that the vast majority of users have rated only 1 or 2 items, and very few have rated 3 or more. On the other hand, while most items have not been rated many times, there distribution of the number of observed reviews diminishes much more slowly than for users' reviews.

From the distribution of user and item reviews, we see that there are likely not enough reviews from each user for a latent factor model to accurately determine the latent dimensions behind the observed rating behavior. As a result, we would expect latent factor models with fewer dimensions to be less likely to overfit to the training set, and have improved generalization behavior.

Finally, we include two plots of the average rating given by each user (Figure 4), and associated with each item (Figure 5). To create these figures, we calculate the average rating associated with each user and item, and plot

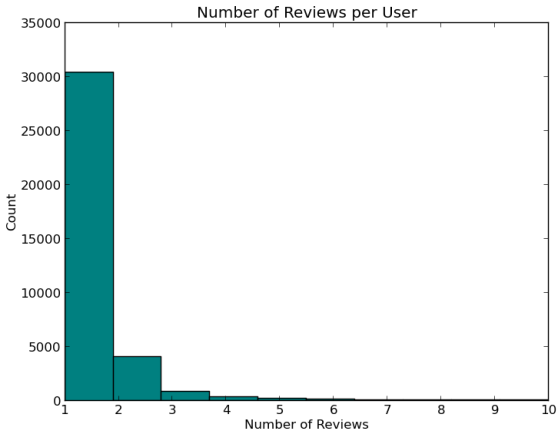

Figure 2: User Review Frequency

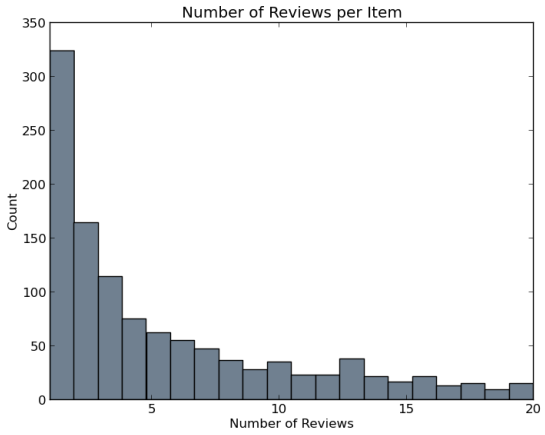

Figure 3: Item Review Frequency

these sorted values. From these figures, we see that over half of all users have given only five star ratings. We also see that there are very few users whose average rating values are between whole numbers. This is due to the sparsity of our dataset, and that the vast majority of users have only rated a single item.

This is contrasted by Figure 5, which shows the average rating for each item. We see that the values in this plot increase at a much slower rate, showing that there are many items that have been rated numerous times. Also, the concavity of this graph demonstrates that the majority of the ratings are 4 or 5 star ratings. The user graph should display this behavior as well, if we had created our dataset such that most users had rated many items.

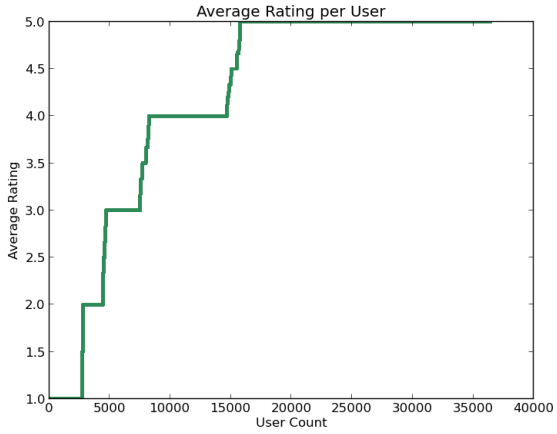

Figure 4: User Average Rating

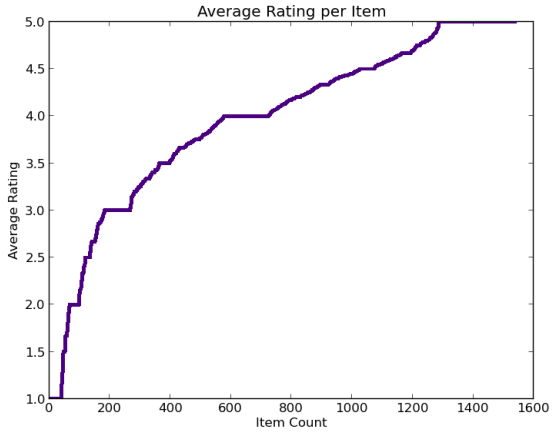

Figure 5: Item Average Rating

3 Predictive Tasks

Given the Amazon movie review dataset, our main predictive task is to predict a user's rating on a particular movie. This is typically a task performed best by dedicated recommender system algorithms. However, due to the extreme sparsity of the data, we would like to incorporate a user's review text into the rating prediction.

As a result, we decided to solve the rating prediction problem with a variety of supervised

learning algorithms, which use features from the text of a user's review. In particular, we decided to use linear regression, logistic regression, and SVMs. Logistic regression and SVMs are both classification algorithms, so we use the multiclass classification versions of these algorithms, treating each rating value as an individual class.

While the above supervised learning algorithms, both regression and classification, are able to utilize the text of the review to predict the ensuing rating, they ignore the interactions and similarities between individual users and items. As a result, we compare these supervised learning algorithms with latent factor recommender systems. Latent factor algorithms uncover latent representations of users and items, and make recommendations based on the proximity between similar users and items in this low-dimensional space.

As we are using the review text to predict rating values, one additional predictive task we are interested in is gaining a deeper understanding of the review text itself. This can be done with Principal Component Analysis (PCA), which is a dimensionality reduction algorithm that extracts the most salient dimensions from the review text. These dimensions can then be analyzed to determine the most important aspects, or topics, in the observed reviews.

3.1 Evaluation

In order to evaluate these methods, we use the Mean Squared Error (MSE). The MSE is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \qquad (1)$$

where $\hat{y}_i$ is the predicted value for a particular example, and y_i is its true value. There are many advantages to using the MSE as an evaluation metric. First, it is easy to compute and differentiate, greatly simplifying many models which rely on derivative-based algorithms such as gradient descent.

Also, when compared compared to linear evaluation metrics such as the Mean Absolute Error (MAE), the MSE penalizes large magnitude errors more severely. As a result, optimal values of the MSE are obtained when errors of small magnitude are distributed evenly across all training examples.

3.2 Baselines

We can derive appropraite baselines for these predictive tasks from analyzing the algorithms detailed above. In particular, the linear regression and latent factor recommender system algorithms provide natural baselines to attempt.

For linear regression, one baseline is to perform a simple linear regression procedure with no regularization. This is one of the simplest machine learning algorithms, but has a high risk of overfitting to the training data.

Latent factor recommender systems include an offset, bias, and latent factor terms for the observed users and items. A natural latent factor baseline is to set the dimensionality of the latent factor terms to 0. This results in a latent factor model consisting of just an offset and bias terms. The offset term corresponds to the global mean rating, and the bias terms correspond to the average deviation from the mean for each user and item.

3.3 Data Preprocessing

The original dataset consists of a list of ratings and their corresponding reviews. While this is sufficient to train a latent factor recommender system algorithm, which only uses user and product IDs, additional processing is required to train models based on the review text.

First, we notice that many of the reviews include HTML formatting along with the review text. While this helps the review be displayed correctly on Amazon's website, it does not help when training a predictive text-based model. If these HTML tags, such as `<br />` are not removed, they can become additional words in

our vocabulary, skewing the observed results.

In order to remove these HTML tags, we use the BeautifulSoup Python library. In particular, we initialize a BeautifulSoup object with the text of the review, and then call its `get_text()` method to extract the text of the review with no markup or HTML tags. This procedure allows to easily process the review's text itself, without having to process additional formatting or markup text.

While there have been some approaches which endeavor to incorporate HTML and formatting tags into various predictive tasks, most algorithms ignore the formatting associated with the text of a review, which can differ significantly between different datasets.

After removing the HTML markup tags from the review text, we convert the review to lowercase and split it into words. We use a regular expression in order to reliably extract individual words, which may be accompanied by unpredictable spacing or punctuation characters. We use the regular expression `'\W+'` to split the string along groups of non-word characters. As a result of this regular expression, all sequences of characters will be correctly extracted as individual words, regardless of the punctuation or spacing between them. This regular expression also has the effect of removing the punctuation characters from the review. However, one downside of this approach is that it splits words that are meant to be joined together, such as hyphenated words. This could impact the predictive power of our models, as two words joined together may have a different meaning from when these words are expressed individually.

After performing the data processing described above, we have converted each review into a list of its constituent words. These words can then be used to create a bag-of-words representation of the review, used by the various supervised learning models.

4 Models

In this section, we describe the various models used to predict ratings of the Amazon movie reviews dataset. In particular, we detail the formulation, justification, and concerns associated with each model.

For all models except PCA, we randomly split the movies dataset into training, validation, and test sets. In particular, the training set contains 90% of the data, and the test set has the other 10%. In addition, the majority of models require tuning of various regulariation hyperparameters. In order to tune these hyperparameters, we take 10% of the original data from the training set to create a validation set. After determining optimal values for the hyperparameters, we evaluate the completed models on the test set, training using 90% of the original data.

4.1 PCA

In order to understand the approximate topics of the observed reviews, we run Principal Component Analysis (PCA) on the 50,000 reviews. PCA is a dimensionality reduction procedure which determines the dimensions in the observed data responsible for the highest observed variance. This can be done to create compressed representations of our data, while still retaining the information which is meaningful to the original data.

The dimensions which represent the directions of highest variance in the original data turn out to be the eigenvectors of the covariance matrix, Σ, of the original data. If we let φ represent the eigenvectors of the covariance matrix, we see that the PCA procedure transforms the original datapoints x into the processed data points $y = \varphi x$. However, as φ has the same dimensionality as the original data points, this does not reduce the dimensionality of our data. To do this, we simply keep the k dimensions corresponding to the largest eigenvalues of Σ. Using these reduced dimensions, represented by φ_k, we can calculate reduced dimensionality representations of our original

datapoints $y_k = \varphi_k x$.

For this project, we are less interested in using PCA to reduce the dimensionality of our data, instead we would like to analyze the derived PCA dimensions to determine their characteristics. We first create bag-of-words representations of the movie reviews in our dataset, using the top 5000 most commonly observed words. We then compute the tf-idf representations of the bag-of-words vectors, in order to estimate the relevance of each word to the document as a whole.

Tf-idf stands for term frequency-inverse document frequency, and is a methodology used to determine the relative importance of each word in our dictionary. In order to do this, we first calculate the term frequency and inverse document frequency. The term frequency (tf) is defined as the number of times a word occurs in a particular review. The inverse document frequency (idf) is defined as follows:

$$idf = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

where N is the total number of reviews in our dataset, D is our entire dataset of reviews, d is a particular review in D, and t is the term we are interested in analyzing. Intuitively, the idf represents how common term t is across all documents in our dataset D.

Finally, we calculate the tf-idf score as:

$$tfidf = tf * idf \tag{3}$$

In order to train PCA on our dataset, we use the `IncrementalPCA` class from the Scikit Learn library. When using the original `PCA` class, we ran into significant memory issues which prevented the model from training successfully. This was due to the intermediate computations in the `PCA` class which compute the Singular Value Decomposition of the entire covariance matrix Σ. In contrast, the `IncrementalPCA` implements the algorithm from [8] and only requires a constant memory overhead, regardless of the number of training examples.

4.2 Linear Regression

The basic linear regression model attempts to solve the matrix equation

$$y = X\theta \tag{4}$$

Solving for θ, we arrive at the analytical solution

$$\theta = (X^T X)^{-1} X^T y \tag{5}$$

In this project, y represents the observed movie ratings associated with each review. X represents the bag-of-words feature matrix generated from the data preprocessing step, and θ represents the model's parameters. We create the bag-of-words feature matrix using only the 1000 most commonly observed words. When running linear regression, we add an additional constant feature to the model, which serves as the model's offset term enabling it to fit hyperplanes which do not pass through the origin.

While this model serves as a good baseline, it has a tendency to overfit to the data. As a result, we add an l2-regularization term to limit the parameter magnitudes and reduce overfitting.

For this formulation of linear regression, an objective function is defined in terms of the MSE.

$$\arg\min_{\theta} \frac{1}{N} ||y - X\theta||_2^2 + \lambda ||\theta||_2^2 \tag{6}$$

This objective function is minimized using an iterative procedure such as gradient descent.

For this project, we use the `Ridge` class in the Scikit Learn library. This allows to train a linear regression model while specifying an arbitrary regularization parameter.

4.3 Logistic Regression

While linear regression directly predicts the rating given a particular review, logistic regression is a binary classification algorithm, predicting whether an example belongs to a particular label or not. In the case of rating prediction, there are many classes that must be

selected between, specifically each possible rating value. As a result, the typical logistic regression procedure must be adjusted to handle this multiclass case.

The multiclass logistic regression scheme used for this project is known as the "one vs rest" scheme. This involves training individual binary predictors for each class, which predict the probability that a particular example will be a member of a given class. In order to make a prediction for a new example, the example is run through each binary predictor, and the class that has the highest associated probability is returned. While this method easily allows us to apply logistic regression to multiclass problems, it is susceptible to discrepancies in the ratio of positive to negative examples. Given the uneven distribution of rating values in our dataset, this could impact the performance of the logistic regression classifier.

We used the `LogisticRegression` class in the Scikit Learn library to train a one vs rest logistic regression classifier on the Amazon dataset. The objective function associated with logistic regression is as follows:

$$l = \sum_i -\log\left(1 + e^{-X_i \cdot \theta}\right) + \sum_{y_i=0} -X_i \cdot \theta - \lambda ||\theta||_2^2 \tag{7}$$

Iteratively training the sklearn logistic regression classifier was very computation and memory intensive. As a result, in order to successfully complete the model training in a reasonable time, we only used the top 1000 words from our bag-of-words model as input features to the algorithm.

4.4 SVM

SVMs are traditionally known as one of the best supervised learning classification algorithms. We tried to train an SVM classifier to predict movie ratings from review text, using Scikit Learn's `SVC` class. The multiclass classification approach used by the `SVC` class is known as "one vs one". Unlike the "one vs rest" scheme described above for logistic regression, the one vs one scheme trains an individual classifier for every pair of output labels, and selects the label predicted by the most classifiers.

Due to the large number of classifiers that must be trained, and due to the time complexity required to train an SVM classifier, we were unable to compute rating predictions using the SVM algorithm.

4.5 Latent Factor Model

Latent factor models are one of the most commonly used and most successful recommender system algorithms. They have a high level of expressiveness, and can fit a wide variety of relationships and interdependencies between various users and items, solely from past observed rating data.

Intuitively, this problem makes predictions via dimensionality reduction. Specifically, the latent factors uncovered during the training process project the users and items down to a reduced dimensionality space. In this space, the items closest to a particular user are given as recommendations.

Like linear regression, latent factor models also provide a natural baseline. While typical latent factor models which have offset, bias, and user and item factor terms, we can create a simplified model with just the offset and bias terms. The offset term can be interpreted as representing the average rating throughout the entire dataset, and the bias terms can be interpreted as encoding the deviation of each user and item's ratings from the global average. The objective function for this baseline model can be expressed as follows:

$$f = \sum_{u,i} (\alpha + \beta_u + \beta_i - R_{u,i})^2 + \lambda \left[\sum_u \beta_u^2 + \sum_i \beta_i^2\right] \tag{8}$$

In contrast, the full model incorporates two additional terms which express the relationships between each user and item. The objective function for this model can be written

mcclane	samurai	hulk	que	bella	wonka
connor	algren	thor	de	sal	charlie
terminator	mcclane	wolverine	la	edward	factory
marcus	katsumoto	season	y	jacob	chocolate
skynet	cruise	banner	en	spike	burton
die	japan	vs	es	twilight	willie
machines	japanese	episode	el	pinocchio	willy
salvation	emperor	x	lo	vampire	wilder
john	watanabe	jeannie	un	moon	season
2007	tom	zulu	las	radio	stuart

Table 2: Text dimensions derived from running PCA on Amazon movie reviews

as follows:

$$f = \sum_{u,i}(\alpha + \beta_u + \beta_i + \gamma_u\gamma_i - R_{u,i})^2 + \lambda\left[\sum_u \beta_u^2 + \sum_i \beta_i^2 + \sum_u ||\gamma_u||_2^2 + \sum_i ||\gamma_i||_2^2\right] \tag{9}$$

These latent factor models were trained using HLBFGS, an iterative gradient-based optimization procedure in C++. Unlike the other supervised learning models, they do not incorporate the review text, instead analyzing the observed users, items and ratings.

5 Results

In this section, we summarize the results of the algorithms described above.

5.1 PCA

We ran PCA on all reviews in the training set, using the 5000 most common words in the vocabulary to create bag-of-words representations for each example. Once the PCA model had been trained, we conducted an analysis of the derived dimensions. In order to determine an interpretation for each dimension, we created example "documents" consisting of just a single word in the dictionary. We calculated the PCA projections of these "documents" and determined which words have the highest values for each dimension. The results are summarized in Table 2.

From this analysis, we see that although many of the most common words appear in multiple dimensions, PCA was able to effectively extract dimensions that correspond to various movies, genres, and languages. This demonstrates that even though our dataset is relatively small at only 50,000 examples, and we only used the 5000 most common words, these words still represent important topics which can be extracted by the PCA procedure.

In order to improve the dimensions generated by PCA, it would be necessary to run the algorithm with an increased number of training examples and features. However, this was not possible with our dataset due to computation and memory limitations.

5.2 Linear Regression

The results for the remainder of the models are summarized in Table 3. From this table, we can see that surprisingly, linear regression performed the best out of all algorithms that were implemented. This simple algorithm outperformed the more complicated logistic regression and latent factor models. A possible interpretation of this result is that due to the small size and sparsity of our dataset, it is very difficult for complicated machine learning models to learn the underlying distribution without overfitting to the training data. Linear regression seems to not have this problem. It is a simple model that assigns weights to each word, and is able to generalize well to unseen instances as a result.

Model	Test MSE
Baseline Linear Regression	1.088
Regularized Linear Regression	1.088
Regularized Logistic Regression	1.306
Baseline Latent Factor Model	1.216
Latent Factor Model, 5 Dimensions	1.247
Latent Factor Model, 25 Dimensions	1.231
Latent Factor Model, 100 Dimensions	1.183
Latent Factor Model, 250 Dimensions	1.174

Table 3: Text dimensions derived from running PCA on Amazon movie reviews

One additional interesting observation is that there is no significant difference between the performance of the baseline and regularized linear regression models. One possible justification of this is the tf-idf processing of the bag-of-words representation of the movie reviews. This processing transformed the data to approximately the same scale, resulting in parameters that were of similar magnitudes. As a result, adding an l2-regularizer did not significantly change the final parameter vector.

5.3 Logistic Regression

From the results summarized in Table 3, we see that Logistic Regression performed the most poorly out of all the models that were attempted. There are a few possible explanations for this result. First, logistic regression is a more complicated model than linear regression, so it has a higher chance of overfitting to the given training set.

Secondly, logistic regression is intended to be used in binary classification settings, whereas in this situation we trained a multiclass predictor. This predictor was trained using the "one vs rest" prediction scheme, where one model is trained per output value and the overall prediction is the model that returns the highest probability.

However, due to the significantly skewed ratings in the training set, most of these models had a significant imbalance between positive and negative examples. The model for predicting a rating of 2.0, for example, had 2746 positive examples and 47254 negative examples. This significant imbalance also leads to overfitting, further impairing the model's ability to generalize to new examples in the test set.

5.4 Latent Factor Models

The latent factor model also performed worse than the linear regression model. This turned out to be the case for both the low and high dimensional models. This is due to the increased complexity associated with latent factor models, which were not able to adequately generalize given a relatively small and sparse dataset.

Interestingly, the baseline latent factor model performed better than latent factor models with 5 and 25 dimensions. This can be attributed to the increased number of parameters leading to more significant overfitting of the training data. The simplest model is essentially a linear model with offset and bias terms, and thus generalizes the best to new user and item pairs.

However, the latent factor models with 100 and 250 dimensions outperformed the latent factor baseline and low dimensional models. This seems counterintuitive, given the significantly increased model complexity. We would have expected this increased complexity to lead to further overfitting, but we actually observed increased performance on the test examples. One possible explanation for this behavior is that the higher dimensional models encountered a local optimum during the training process, which happened to pexhibit bet-

ter performance on the test set. Also, the improved performance could also be a function of the particular random initialization of the latent factor parameters.

6 Related Work

The methods and applications described in this project are extensively studied topics. The linear regression, logistic regression, SVD, and latent factor models are commonly used machine learning models, and have been successfully applied to a wide variety of problems and disciplines.

With regards to recommender systems, there have been a wide variety of proposed algorithms to deal with the cold-start problem. These algorithms incorporate a variety of external information to help make predictions for users and items with few observed ratings. [1] provides an algorithm for making recommendations based on implicit feedback datasets, such as purchase or browsing history. It does so by recording and adjusting confidence parameters for each user and item pair.

In addition, there are recommender system algorithms which incorporate additional implicit information from users' social connections and community memberships. These additional data sources are often added as regularizers, enforcing certain restrictions on the learned parameters. [6] proposes an algorithm for socially regularized recommender systems, enforcing that a user's factors be similar to those of his friends. Similarly, [5] introduces algorithms for recommender systems that are regularized by community memberships, forcing a user's latent factors to be similar to those of other users in the same communities.

Applying supervised learning techniques to text classification problems is also a well-studied field. Many regression and classification algorithms have been applied to text problems, and online product reviews in particular. [2] and [3] train logistic regression models to highlight review spam, where individuals write false and misleading reviews in order to achieve some levels of personal gain and profit.

Finally, there has been significant research into incorporating review text into recommender system algorithms. As previously discussed, the review text associated with a particular rating provides useful information that can help improve rating predictions for new users and items. [7] demonstrates that combining the latent factors learned by recommender system algorithms and the topics in review text discovered by topic modeling and dimensionality reduction techniques results in significant improvements in performance and interpretability of results.

7 Conclusions

In this project we analyzed a sigment of the Amazon movie reviews data, and compared the performances of latent factor recommender system models with supervised learning algorithms that incorporated the text of the review itself. We created bag of words models from the observed reviews, and used PCA to understand the salient topics present. We trained linear regression, logistic regression, and latent factor algorithms, and observed that the best-performing algorithm was linear regression. This is likely due to the small size and sparsity of the movie dataset used for this task, demonstrating that simple algorithms that incorporate the text of each review are the most effective at handling cold-start users and items.

References

[1] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[2] N. Jindal and B. Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190. ACM, 2007.

[3] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[4] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[5] H. Li, D. Wu, W. Tang, and N. Mamoulis. Overlapping community regularization for rating prediction in social recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 27–34. ACM, 2015.

[6] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[7] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[8] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.