Combining Personalized Agents to Improve Content-Based Recommendations

Jason M. Adams, Paul N. Bennett, Anthony Tomasic

CMU-LTI-07-015

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Combining Personalized Agents to Improve Content-Based Recommendations

Jason M. Adams
Carnegie Mellon University
jmadams@cs.cmu.edu

Paul N. Bennett*
Microsoft Research
paul.n.bennett@microsoft.com

Anthony Tomasic
Carnegie Mellon University
tomasic@cs.cmu.edu

December 12, 2007

Abstract

Ratings-based recommender systems typically predict user preferences for items based on the user's preference history, information about items, and the preferences of similar users. In content-based recommending, the similarities between items the user has previously expressed interest in form the basis for recommending new items. There are a number of practical reasons why users may not rate all of the items they have experience with, a fact that indicates ratings are not missing at random. We introduce a missing data model that takes this observation into account. By combining the personalized content models with missing data models, we build classifier agents for each user using the predicted ratings of the first two models. These stacked agents use collaborative filtering to construct a hybrid recommender system that improves upon the baseline scores produced by the content-based recommender on a popular movie ratings data set.

1 Introduction

Ratings-based recommender systems serve many purposes for e-commerce. For customers, they help overcome information overload by providing new product recommendations they were unaware of. Often, these recommendations provide them options similar to those enjoyed in the past, but the real benefit comes when the recommendations point out high value items that would have gone unnoticed. In turn, these systems help companies by boosting sales and building customer loyalty as they provide a more valuable service.

The two primary approaches to computing recommendations are *content-based* and *collaborative filtering*. The former seeks to find similarities between items by comparing item characteristics. In the domain of movie ratings, these profiles may consist of actors, directors, genres, production companies, etc. Collaborative filtering exploits the similarity in preferences between different users to predict preferences on new items. These two approaches are often combined into a *hybrid* approach. A *unified hybrid model* combines both content-based and collaborative characteristics into a single model, rather than incorporating one approach into another [1].

In this paper, we describe a framework that combines missing data scores with content-based recommendations to produce a hybrid recommendation system. In the first stage, personalized user agents produce recommendations for items with a content-based approach. Next, a second agent models the likelihood that the user already knows this item to be interesting. This model of the missing data is combined with the personalized content agents to form a stacked agents model using collaborative filtering. With this technique, we show improved results over

*This author's research contributions were made while a postdoctoral fellow at Carnegie Mellon.

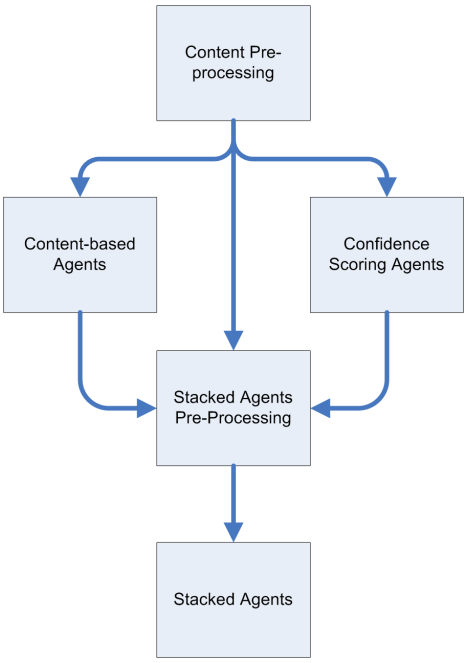

Figure 1: The Stacked Agents Model.

a baseline content model.

In Section 2, we begin with a description of the stacked agents model, with the details of our implementation in Section 3. In that Section, we also describe how support vector machines serve as the learning mechanism for providing item ratings and missing data predictions. In Section 4, we describe our experimental setup and report on our results using the MovieLens data set in Section 5. In Section 6, we discuss related work followed by possibilities for future work are given in Section 7. We end the paper with our conclusions in Section 8.

2 Stacked Agents Model

In a recommendation setting, the user is not able to rate every item (otherwise, providing recommendations would be pointless). In most common interfaces and situations, the system is only able to present a small subset of items to the user. For various reasons, most users will not rate every item they have had experience with and could provide a rating for. We propose this assumption as one aspect of the *missing data mechanism* for recommender systems. The research done to date has largely assumed that data is *missing at random* [13], with the exception of [14]. Since it is not possible to determine empirically, we make the assumption that unrated items are *not* missing at random. From this assumption, it follows that features of items that have been rated correlate with features of items that have not been rated.

We can use content-based models to attempt to predict a user's rating for any item based on the characteristics of the item and previous items the user has rated. These models operate under the assumption that the user will like items similar to other items they are interested in. Likewise, we can apply similar content-based models to the task of determining whether a user is likely to already have found an item to be interesting. Content-based models are often weak when it comes to predicting items that a user likes but are dissimilar to other items they have rated highly. Collaborative filtering overcomes this approach by looking at similarities in user profiles. By combining the content-based approaches as the input to a collaborative filtering system, it was our intent to mitigate the effect of the weakness of content-based models and exploit the advantage of collaborative filtering.

The stacked agents model (presented in Figure 1) combines content-based recommendations and scores on missing data with collaborative filtering to produce a unified hybrid recommender system. In the content pre-processing phase, data is gathered about the items being recommended, which is then used to train the learning algorithm for the personalized *content-based* and *missing data* agents. Content-based agents predict the user's rating for each item using only information gathered about that item. Known-interest agents assign scores to those predictions representing the level of certainty the system has in them. Together these models can identify where we are confident about a user's interest from a content perspective. These predictions and missing data scores are carried forward into a pre-processing phase, where content information may be optionally recombined as additional features. The learner is

then trained on the predictions and content information to make final predictions of user ratings for items.

The missing data model is a classifier that is equivalent to a confidence score of how likely it is the user will rate an item in the limit. Another way of thinking of this score is as how likely it would be for the user to choose the item. Having a score for each item could help many applications. In a recommender system, it is often beneficial to have ratings for items that offer the most discriminatory power. The items that partition the space most effectively can be chosen from the list of items the user is likely to have rated but actually has not. These items can then be presented to the user to rate so that high quality predictions can be produced more efficiently.

Our system combines personalized recommender agents for each user into a unified *stacked model* through the blending of a missing data score with content-based recommendations. Each user's agent predicts ratings for all items with a standard content-based method. A similar content-based approach is used to build a classifier to predict which items the user would be likely to rate. The missing data value and the predicted rating are then combined as a single feature in the stacked model, representing both the rating and missing data for that user and item. This creates an $n \times m$ collaborative filtering matrix, where n is the number of users and m is the number of items. The value in each cell $a_{i,j}$ is the weighted, predicted rating of user i for item j.

2.1 Personalized Content Model

The first component of the stacked agents model is the construction of personalized, content-based recommender agents for each user. In content-based recommending, features are collected for the items being recommended to produce a set of item characteristics stored in a feature vector, $\vec{x}$. A label y for each vector indicates the actual rating of the user for the item, and $\hat{y}$ indicates the predicted rating. Figure 2 shows the configuration of the feature vector.

In general, content features can be produced from any sort of metadata or descriptions associated with an item. In our case, we crawled data from the Internet Movie Database (IMDB)[1] to obtain semi-structured textual descriptions of each item.

As is common in content-based recommender systems as well as information retrieval systems, we create a feature for each term in the descriptions and calculate that feature's value by using *term frequency/inverse document frequency* (TFIDF) [1, 5]. For a discussion of how TFIDF is computed in our system, please see the Appendix. For our content-based models, we use TFIDF to compute weights for the features that describe each item. Further details of the content-based representation are presented in Section 3.

2.2 Missing Data Model

The second component of our model is the *missing data model*. This model assigns a score to each prediction $q_{u,v}$ produced by the content-based agents, for user u and item v. The purpose of this model is to produce a value corresponding to the user's interest in the item. To construct this classifier, we use feature vectors identical to those used by the content-based recommender agents. The labels assigned to each item correspond to whether the user has rated the item or not. The output of the classifier is a continuous value in the interval $(-\infty, \infty)$ and has been shown to correspond to a poorly calibrated log-odds estimate of the missing value $r_{missing}$ given the user u and item v [4, 17]:

$$\hat{y} \approx \log \frac{P(r_{missing}|u,v)}{1 - P(r_{missing}|u,v)}. \quad (1)$$

By applying the sigmoid function in Equation 2 to the predicted value $\hat{y}$, we obtain Equation 3, an estimate of the probability of the item being missing given the user and item.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \quad (2)$$

$$P(r|u,v) \approx \text{sigmoid}(\hat{y}) \quad (3)$$

We then assign a missing data score based on the estimated probability by considering values near 1 to

[1] http://www.imdb.com

be *missing* and values near 0 to be *not missing*:

$$q_{u,v} = \text{sigmoid}(-\hat{y}). \quad (4)$$

2.3 Delta Space

We perform a pre-processing step on the data by converting it into *delta space*. The average rating for each item v is found without considering the ratings by user u. This produces a delta score, δ. Each rating for that item is then transformed according to this delta value to produce the rating $r_{u,v}$ used by our system. Specifically, where $r'_{u,v}$ is the initial rating by user $u \in U$ for item v, the delta rating $r_{u,v}$ is computed as

$$r_{u,v} = r'_{u,v} - \frac{1}{\|U - u\|} \sum_{u' \in U-u} r'_{u',v} \quad (5)$$

2.4 Stacked Agents Model

The final component of our model is the *stacked model*. We combine the rating predictions produced by the content-based agents with their missing data scores to produce a single, combined prediction for each user and item. Given a predicted rating $p_{u,v}$ for user u and item v, the combined prediction is defined as

$$\hat{r}_{u,v} = p_{u,v} \times q_{u,v}. \quad (6)$$

The feature vector $\vec{r}_{u,v}$ consists of the combined predictions of all other users for item v. This feature vector may optionally contain the item characteristics feature vector $\vec{x}$ used by the other two types of agents to produce the final feature vector $\vec{z}_{u,v}$ for user u and item v (see Figure 3). The matrix of combined predictions reflects the standard setup for collaborative filtering algorithms [9]. However, since the collaborative filtering matrix is boosted by content-based predictions, it constitutes a unified hybrid recommendation system. A machine learning algorithm may then be trained on the feature vectors for a given user to produce predicted ratings.

Stage	Description
1	Produce predicted ratings $p_{u,v}$ for each user u and item v using the content model
2	Produce missing data scores $q_{u,v}$ for each user and item under the same content model
3	Combine scores to produce stacked feature vectors $\vec{z}_{u,v}$ for each user and item
4	Train the learning algorithm to produce final predictions $\hat{y}_{u,v}$

Table 1: Stages used in our approach.

3 Implementation

Our system was implemented to produce movie ratings using a four stage approach (see Table 1). The first stage consists of pre-processing the data and creating the feature vectors for each user as input to the leaner. This processing step is done for both the content-based agents and the missing data agents. The second stage trains SVMs for each user using the output of the first stage and produces predictions for the next stage. The third stage processes this output and creates the feature vectors for the stacked agents model. The final stage trains the learner on this output and produces predictions for each user and item.

3.1 Support Vector Machines

For our learner, we used the support vector machine implementation SVM^{light} [11]. SVMs partition a complex feature space by finding the maximum margin hyperplane that separates the data. One advantage of an SVM is its use of a kernel function. The kernel space with an appropriate non-linear kernel allows the SVM to solve the problem as if it were linearly separable. A test example x is classified depending on the sign of the side of the hyperplane in which it lies. By calculating the distance of the example from the hyperplane, the SVM essentially provides a built-in estimate of the reliability of the

Item	Feature Vector	Labels	Predictions
1	$\vec{x}_1$	y_1	$\hat{y}_1$
2	$\vec{x}_2$	y_2	$\hat{y}_2$
⋮	⋮	⋮	⋮
m-1	$\vec{x}_{m-1}$	y_{m-1}	$\hat{y}_{m-1}$
m	$\vec{x}_m$	y_m	$\hat{y}_m$

Figure 2: User profile matrix for personalized content and missing data agents.

Actors	Actresses
Directors	Genres
Keywords	MPAA Ratings
Plot Summaries	Producers
Production Companies	Release Dates
Movie Length	

Table 2: Content features extracted from IMDB.

prediction, and it corresponds to a poorly calibrated log-odds estimate (see Section 2.2) [4, 17].

The weights for each feature in the content model feature vector are computed using TFIDF as described in Section 2.1. Each feature listed in Table 2 is weighted according to the number of times it occurs and the number of movies it occurs in. Duplicate terms and people were treated as single features. For example, if Tom Cruise were both a producer and an actor in some movie, he would be treated as a single feature with a term frequency of two. The total number of movies in which he was either an actor, director or producer would constitute the document frequency for his feature.

The test set is held out as a means of validating the output of the SVM for all three models. Feature computations ignore the test set and use only the features that appear in the training set. The prediction set (for the content and missing data models) consists of all items in the system using only the features that were present in the training data.

After the personalized content and missing data models have been assembled from the data, the SVM is trained on the corresponding training set to produce the predictions that will be fed into the stacked model. Validation is done prior to stacking using a held out test set for these two models. For the personalized content model, we used the linear kernel for the SVM, a standard practice in text classification. After training, the SVM produces a model file for each user that can be used for classifying new examples. The SVM is then tested on the test set as a means of verifying how well it has learned the user's profile. This verification is done using the NMAE score described in Equation 8 in Section 5. Next, the SVM classifies the prediction sets to produce predictions of the user's rating for every item as well as a missing data score for each item. For the content-based recommender, we chose the linear kernel for the SVM. We used regression mode to produce output as a rating value, rather than as a classification. We chose the Gaussian kernel to model the missing data since it allows for a non-linear decision surface that has neighborhoods and matches our intuition that rated data will tend to clump into neighborhoods. For the Gaussian kernel, there is a γ parameter that can be tuned for each user. We used 4-fold cross validation on the training set to tune each value of γ. For both kernels, it is possible to tune the cost factor by which positive and negative examples are weighted. We found that tuning this parameter tended to yield worse results than leaving it at the default value of an even trade-off.

The output of the SVMs trained for the personalized content and missing data models is the input to the stacked agents model. Figure 3 shows the configuration of the new user profile matrix in this model. The vector $\vec{x}$ is the feature vector from Figure 2, with weights readjusted to reflect the movies present in the new training set. A test set is held out to measure results. The new feature vector $\vec{r}$ is the set of combined predictions calculated as described in Section 2.4.

The final stage consists of training and testing the SVM for each user using these new feature vectors. In our preliminary experiments, the Gaussian kernel performed the best for the stacked agents model. In this case, we used 4-fold cross validation on the training set to tune each value of γ and we ignored the cost factor due to its detrimental effect.

Item	Feature Vector							Labels	Predictions
1	$\vec{x}_1$	$r_{1,2}$	$\cdots$	$r_{1,i-1}$	$r_{1,i+1}$	$\cdots$	$r_{1,n}$	y_1	$\hat{y}_1$
2	$\vec{x}_2$	$r_{2,2}$	$\cdots$	$r_{2,i-1}$	$r_{2,i+1}$	$\cdots$	$r_{2,n}$	y_2	$\hat{y}_2$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
m-1	$\vec{x}_{m-1}$	$r_{m-1,2}$	$\cdots$	$r_{m-1,i-1}$	$r_{m-1,i+1}$	$\cdots$	$r_{m-1,n}$	y_{m-1}	$\hat{y}_{m-1}$
m	$\vec{x}_m$	$r_{m,2}$	$\cdots$	$r_{m,i-1}$	$r_{m,i+1}$	$\cdots$	$r_{m,n}$	y_m	$\hat{y}_m$
		$\overbrace{\quad\quad\quad\quad\quad\quad}^{\vec{r}_u}$							
	$\overbrace{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}^{\vec{z}_u}$								

Figure 3: User profile matrix for stacked agents model for user i. Element $r_{u,v}$ is the combined prediction (equation 6) of user u for item v. Note that the user himself is not included in his own feature vector.

4 Experiments

We report on experiments using the 1M MovieLens data set made available by the GroupLens project[2]. This data set consists of approximately one million ratings by 6040 users on 3800 movies. Ratings consist of the values $\{1, 2, \ldots, 5\}$, with each user having at least 20 ratings. Content features were derived from IMDB and any feature that appeared in only one item was discarded, a standard text classification practice. We extracted a total of 151,822 features from IMDB that corresponded to movies in the MovieLens data set. Many movie titles given in the MovieLens data set did not match the movie titles listed in IMDB, so those titles were manually changed to match IMDB. We base our experiments on the setup described by [13]. Our experiments follow Marlin's model of *weak generalization*, in which movies are held out for each user and the learner predicts the ratings of the held out set.

To perform our experiment, we selected 5000 users from the MovieLens data set. For each user, one movie was held out as the test set for *leave-one-out* and the rest were used as training data. The stacked agents model was trained on each user to produce predicted ratings for the test case. We compute the NMAE for each user and report the average NMAE across all users as described in the next section. All experiments were performed on a machine consisting of four 3.06 GHz Pentium 4 CPUs with 2 GB RAM.

[2]http://www.grouplens.org

5 Results

5.1 Methodology

To evaluate our results, we use the *normalized mean absolute error* (NMAE) described by [13]. The *mean absolute error* (MAE) is defined as

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \tag{7}$$

where N is the number of items, y_i is the actual rating for item v, and $\hat{y}_i$ is the predicted rating. This is calculated for each user and then averaged. The MAE is normalized with a normalization constant that depends on the rating scale. The normalization constant is the expected mean absolute error of uniformly distributed ratings with random predictions. For MovieLens, where ratings are defined as $\{1, 2, \ldots, 5\}$, the normalization constant is 1.6. Thus, the NMAE for our data set is defined as

$$NMAE = \frac{MAE}{1.6}. \tag{8}$$

The error for the missing data model is measured simply as the ratio of incorrect classifications to the total classifications. We also report the number of

missing data scores that occurred in the margin, that is, scores that the SVM did not indicate were reliable. We compare our results to the baseline NMAE given by our content model.

5.2 Discussion

The results of our experiments are summarized in Table 3. Performing the processing for the personalized content and missing data agents took approximately 70 hours, and the stacked agents model took another 17 hours. Of the nearly 152,000 features we extracted, an average of around 16,000 were used in the content-based agents for each user, while nearly all were used in the missing data agents. As can be seen in the table, the stacked agents model improves over the content model NMAE score of 0.4405 by 1.703% when content features are omitted in the stacked agents model. This result shows that it is possible to improve over a baseline content model using missing data scores and the stacked agents model. We attempted four experiments, in which the user and test sets were held constant and two parameters were modified. The first parameter was the inclusion of the content features as features in the stacked agents model. The second parameter was pruning of the feature space so that the combined predictions (equation 6) with the lowest absolute value were discarded. We chose to remove 50% of the features in this manner for this test. We found that the best results were obtained by not using the content features, while pruning the combined predictions had only a negligible effect.

We further analyzed the results of our data set by comparing the actual and predicted results of the stacked agents model. Figure 4 shows the distribution of users per rating score in delta space. The valid values for delta space are $\{-4, -3, \ldots, 3, 4\}$ since the rating scale for MovieLens is $\{1, 2, \ldots, 5\}$. The predicted ratings are more heavily concentrated around the average rating, indicating that the SVM guesses the average score more often than it occurs in the data. One possible reason for this discrepancy is that if the data is not linearly separable, it minimizes the cost to guess the average rather than guessing a value that deviates from it too greatly.

6 Related Work

Hybrid recommendation systems have often been used to avoid weaknesses in standalone content-based and collaborative filtering approaches. The creators of *Fab*, an early hybrid recommendation system, pointed out that content-based systems alone suffer from *over-specialization* [2]. This occurs when the system is only able to recommend items scoring highly against a user's profile, and is therefore unable to recommend anything that is not similar. Collaborative techniques do not suffer from this shortcoming, so a combination of the two approaches is a natural extension. However, collaborative approaches suffer when new items are introduced. If there are no ratings for a new item, there is no way to recommend it, even if it is an item that is similar to others a user has chosen. Again, the combination of the two approaches seems obvious, since content-based approaches are ideally suited for this situation. In the case of the *Fab* system, and others, hybrid approaches improved on the results of standalone approaches [2, 21, 15]. These improved results were one motivation for our use of a hybrid recommendation system as the basis for the stacked agents model.

Other approaches, such as *latent semantic analysis* (LSA), have looked at ways of reducing the dimensionality of the collaborative filtering matrix, which

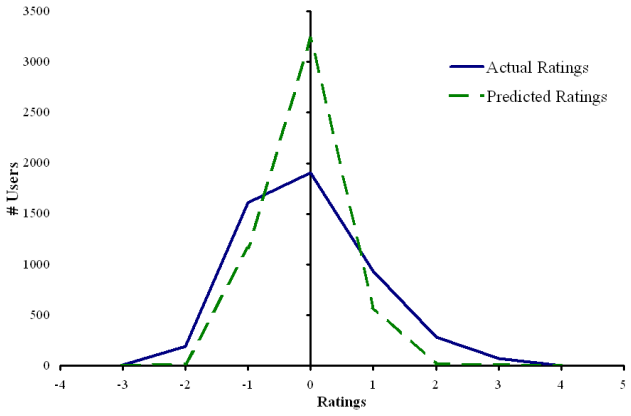

Figure 4: Comparison of actual and predicted ratings.

Content Features	Stacked Pruning	Content Model NMAE	Confidence Model Error	% Confidence in the Margin	Stacked Model NMAE	Improvement over Content
+	-	0.4405	0.0414	21.42%	0.456125	-3.547%
+	+	0.4405	0.0414	21.42%	0.455625	-3.434%
-	-	0.4405	0.0414	21.42%	0.433000	+1.703%
-	+	0.4405	0.0414	21.42%	0.433000	+1.703%

Table 3: Results for the MovieLens data set.

is sparse [21, 20]. LSA in this context seeks to find relationships between users by finding the low-rank approximation to the sparse collaborative filtering matrix. Singular Value Decomposition (SVD) has also been applied to collaborative filtering by finding the optimal user-item matrix decomposition according to squared error. SVD has trouble with sparse matrices with many missing values and will often over-fit, but a recently proposed Variational Bayesian approach somewhat overcomes this by introducing priors. The squared error objective function is replaced with a probabilistic model leading to significant improvements over regular SVD on the Netflix Prize[3] data set [12]. Maximum Margin Matrix Factorization, on the other hand, does not limit the dimensionality of the matrix, but instead limits the number of dimensions that are considered important [18]. MMMF has led to much improved results, outperforming the best results reported by Marlin, which used a variety of machine learning techniques [13]. MMMF has been extended using ensembles, producing the current state of the art results [6]. Rather than reducing dimensionality in the collaborative filtering matrix, our approach reduces the effect of each user's rating by the missing data score corresponding to that user and item.

Support Vector Machines have been applied to the problem of text categorization with good results [10]. Content-based recommendations fit well with the term-document paradigm in information retrieval and so their use as a machine learning technique in this arena seemed natural. Joachims also reports that SVMs are able to perform in high-dimensional feature spaces, which is ideal for our approach since we do not perform any sort of dimensionality reduction. Other findings have shown that SVMs perform worse than kNN for the collaborative filtering domain [8]. Additional work has been done on improving kNN using data from the Netflix Prize competition [3]. In that work, Bell and Koren show that substantial improvements to kNN for the task of collaborative filtering can be made by changing the approach to normalizing the input data and by altering the method for relating users (or items). The *Gravity Recommendation System* uses a variety of different approaches including kNN, neural networks, clustering and matrix factorization that are interpolated into a final model [22]. Their results on the Netflix Prize data set show that machine learning approaches are useful especially in conjunction with matrix factorization-based approaches.

The notion of user agents for recommender systems have been described by a number of researchers. Good et al describe the use of personal user agents that act as filterbots for creating a hybrid recommender system [7]. The filterbots are information filtering agents that seek to reduce the noise in the collaborative filtering results using content information. This idea was later extended to improve recommendations in situations when either new items have received few ratings or new users have provided few ratings [16]. Another early approach to agent-based recommendation systems used agents that formed shared interest groups between users in a collaborative filtering setting [23]. The interest groups could then dynamically update as user preferences evolved. Recently, CinemaScreen has used recommender agents that weight the output of collabora-

[3]More information available at http://www.netflixprize.com.

tive filtering system based on a content-based filtering system [19].

Our system takes a different approach by stacking predictions for movies that are missing in the data. We combine personalized content models with missing data models to produce item-based predictions of user ratings. The stacking model combines these scores and each user agent is trained using an SVM to produce recommendations. This approach yields a unified hybrid recommendation system capable of improving on content-based ratings.

7 Future Work

The combination of the content-based and collaborative filtering approaches has many advantages over pure collaborative filtering [1, 20]. One particular weakness of collaborative filtering is the *cold-start problem*, which occurs when either a user has rated too few items to build an accurate user profile or an item has too few ratings for collaborative filtering to be very useful. Because we combine content-based with collaborative filtering, we expect our method will perform very well on the cold-start problem. In future work, we intend to examine how well our approach handles situations when there are few users rating an item and when a user has rated only a few items. We believe we will find that our system performs well in these cases and outperforms approaches that rely on pure collaborative filtering.

Another area we would like to explore is what effect the machine learning algorithm has on the accuracy of the system. We used SVMs as our learning algorithm, but it may be that other algorithms, such as kNN or decision trees, will outperform SVMs. Other feature selection techniques could also lead to improved results. Under our model, we apply the sigmoid function (Equation. 2) to the output of the missing data model to produce a score in the interval $[0, 1]$. It may be possible to improve classification accuracy for the missing data model by using a held-out set to tune the parameters a and b in the parameterized version of the sigmoid function:

$$q_{u,v} = \frac{1}{1 + e^{ax+b}}. \tag{9}$$

We use a simple thresholding function where features that appear in only one item are removed. It may be beneficial to use mutual information or chi-squared to prune additional features to make the feature space easier to separate.

8 Conclusions

Producing high quality recommendations is an important tool for web-based businesses. Content-based recommenders use similarities between items to provide recommendations based on items the user has already rated. While often useful at overcoming cold-start problems, content-based recommenders suffer from being unable to recommend items the user may still enjoy but which are different from the ones he has rated. Another difficulty in recommendation systems lies in discriminating between information that is missing because the user has not provided enough ratings, and information that is missing because the user has no experience with the item. In our system, we have built classifiers that act as personalized content and missing data models. These scores are combined in a collaborative filtering framework to train personalized stacked agents for each user. By doing so, we show improved results over our baseline content-based recommender system. We also show one possible way of incorporating a missing data model into a hybrid recommender system. We believe future work will show that our system leads to improvements on the cold start problem.

References

[1] Adomavicius, M.-G., and Tuzhilin, M.-A. 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 (2005), 734–749.

[2] Balabanović, M., and Shoham, Y. Fab: content-based, collaborative recommendation. *Commun. ACM 40*, 3 (1997), 66–72.

[3] Bell, R., and Koren, Y. Improved Neighborhood-based Collaborative Filtering.

[4] Bennett, P. N. Using asymmetric distributions to improve text classifier probability estimates. In *SIGIR '03* (2003).

[5] Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference.* (1998), pp. 43–52.

[6] DeCoste, D. Collaborative prediction using ensembles of Maximum Margin Matrix Factorizations. *Proceedings of the 23rd international conference on Machine learning* (2006), 249–256.

[7] Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B. M., Herlocker, J. L., and Riedl, J. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI* (1999), pp. 439–446.

[8] Grcar, M., Fortuna, B., Mladenic, D., and Grobelnik, M. kNN versus SVM in the collaborative filtering framework. In *WebKDD 2005: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005)* (Aug. 2005).

[9] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1999), ACM, pp. 230–237.

[10] Joachims, T. Text categorization with suport vector machines: Learning with many relevant features. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning* (London, UK, 1998), Springer-Verlag, pp. 137–142.

[11] Joachims, T. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning* (1999), 169–184.

[12] Lim, Y., and Teh, Y. Variational Bayesian Approach to Movie Rating Prediction.

[13] Marlin, B. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.

[14] Marlin, B. M., Roweis, S. T., and Zemel, R. S. Unsupervised Learning with Non-Ignorable Missing Data. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS 2005)* (2005), pp. 222–229.

[15] Melville, P., Mooney, R. J., and Nagarajan, R. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence* (Menlo Park, CA, USA, 2002), American Association for Artificial Intelligence, pp. 187–192.

[16] Park, S.-T., Pennock, D., Madani, O., Good, N., and DeCoste, D. Naive filterbots for robust cold-start recommendations. In *KDD '06: proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (New York, NY, USA, 2006), ACM Press, pp. 699–705.

[17] Platt, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, Eds. MIT Press, 1999.

[18] Rennie, J. D. M., and Srebro, N. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning* (New York, NY, USA, 2005), ACM Press, pp. 713–719.

[19] Salter, J., and Antonopoulos, N. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems 21*, 1 (2006), 35–41.

[20] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2002), ACM Press, pp. 253–260.

[21] Soboroff, I., and Nicholas, C. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI '99 Workshop on Machine Learning in Information Filtering* (11 Aug. 1999), pp. 86–91.

[22] Takacs, G., Pilaszy, I., Nemeth, B., and Tikk, D. On the Gravity Recommendation System.

[23] Uchyigit, G., and Clark, K. Agents that model and learn user interests for dynamic collaborative filtering. In *Cooperative Information Agents VI : 6th International Workshop, CIA 2002, Madrid, Spain, September 18 - 20, 2002. Proceedings* (2002), vol. 2446/2002 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, p. 152.

Appendix

Let $t_{i,j}$ be the number of times that term i occurs in document j. The normalized term frequency is computed as follows:

$$tf_{i,j} = \frac{t_{i,j}}{max_k(t_{k,j})}, \tag{10}$$

where the normalization value is the number of occurrences of the most frequent term in document j. The inverse document frequency for term i is defined as

$$idf_i = log\frac{N}{n_i}, \tag{11}$$

where N is the total number of documents and n_i is the number of documents containing term i. The product of the term frequency and inverse document frequency is the TFIDF weight for term i and document j:

$$w_{i,j} = tf_{i,j} \times idf_i. \tag{12}$$