# Recommending Items using collectively trained models

Rohan Passi
ABV-Indian Institute of Information Technology and Management
Gwalior, India
rohanpassi94@gmail.com
Anupam Shukla
ABV-Indian Institute of Information Technology and Management
Gwalior, India
dranupamshukla@gmail.com
Joydip Dhar
ABV-Indian Institute of Information Technology and Management
Gwalior, India
jdhar@iiitm.ac.in

***ABSTRACT:*** *Linear models with nonlinear attribute modeling are popularly selected for massive regression and categorization tasks with sparse inputs. Memorization of attribute interactions through a broad batch of cross-product attribute conversion is compelling and explainable, while generalization needs extra feature engineering exercise. With limited effort, deep neural networks (DNN) can be generalized exceptionally to undiscovered feature sequences. However, DNN can over-generalize and recommend lesser suited items. In this paper, we propose an ensemble of various collectively trained linear models and DNNs to blend the advantage of memorization and generalization for a recommender system (RS). We evaluated the system on MovieLens 100K Dataset, a stable benchmark dataset with 100,000 ratings on 1682 movies from 943 users. Our experiment results show that ensembling of wide & deep models significantly increased recommendation accuracy and decreased mean absolute error (MAE) and root mean square error (RMSE) in comparison with collaborative filtering techniques, deep-only and wide-only models.*

**Keywords**: Collective training, Linear Regression, Deep Neural Networks, Recommender System

**Received:** 19 April 2017, Revised 22 May 2017, Accepted 26 May 2017



## 1. Introduction

There are many applications for data mining techniques in e-commerce such as Recommender Systems (RSs). The useful information obtained from the RS can be used directly by the companies or manufacturers of the product to recommend other products based on user's previous transactions. The main objective of an RS is to analyze the user's preferences and tastes and then provide him with some items that are not known to him. RS allows personalization for e-commerce by exploiting similarities and dissimilarities among user's preferences. A variety of algorithms are being used for providing recommendations, like Collaborative Filtering, Association Rule Mining (ARM)[1] [2], Matrix Factorization(MF)[3], Neural Networks(NN), Regression

Analysis[4] etc. While all the existing RSs have good characteristics, they are inefficient in providing recommendations which take into consideration both user's preferences and general population trends.

RSs are employed to provide users with a richer involvement and help them make the selection process easier. With the use of some standard recommender algorithms, a user could be provided with accurate recommendations. In this work, we propose one such RS which uses ans ensemble of different Wide and Deep models [5] i.e. collectively trained deep neural networks and linear models to find a particular set of products to be recommended by achieving both generalization and memorization. RSs suffer from various issues but we would target few of them like cold-start problem and gray sheep problem.

Cold-start is an inherent problem in recommendation frameworks which especially targets that the system can't sketch any deductions for which it has not yet assembled sufficient data. The cold-start issue is most common in RSs. RSs forms a particular sort of data separating method that endeavors to present things that are fairly important to the user. Normally, an RS looks at the user's profile to reference some attributes. These attributes might be from the user's data or the item's data.

In the collaborative filtering approach, the RS would analyze users who have similar tastes, and come up with things which the similar users favor. Because of the cold-start problem, this approach would neglect the things which nobody in the group has rated beforehand.

Gray sheep points to the users whose preferences don't constantly comply or contradict with a set of individuals and hence don't gain assistance from CF. Black sheep are another set of individuals whose particular tastes make suggestions practically absurd. For instance, let there be 3 individuals *A, B* and *C* in the Database and there be six things *I1, I2, I3, I4, I5* and *I6* in the database. Let's suppose that user *A* lean towards *I1, I2, I3* user B inclines toward *I2, I3, I4* and user *C* favors item I6, then there is a clear chance that user *A* would lean toward item I4 since user *A* and user *B* share some common items. User *A* cannot suggest item I6 as there are no shared preferences between user *A* and *C*.

Nonetheless, not all users in the database can be presented with some item suggestions. In the above case, user *C* can't be recommended with any of the items out of *I1, I2, I3, I4, I5* and *I6*. This is on account of user *C*'s profile regarding the things is not like both of the users *A* and *B*. Such a user in the database for whom any helpful suggestion can't be made is named as gray sheep.

The primary goal of this paper is to propose a new RS that would be able to achieve both generalization and memorization using collectively trained wide and deep models. Another closely related objectives of the are:

- Using collective training to reduce training time of individual model.
- Proposing solution to Cold-start problem and Gray sheep problem.

The first part of the paper gives an analysis of related work including the advantages and disadvantages of existing RSs. The next part focuses on the proposed RS. Finally, we discuss how our proposed RS outperforms the existing RSs and conclude with results and future work.

## 2. Related work

The Collaborative Filtering (CF) RS is one of the first and the remarkably efficient technology for RSs [6] [7]. This RS is extensively used in many e-commerce companies such as Amazon and Netflix [8]. The CF RS recommends items to an individual user based on the items that are not rated by that user but have been rated by some other users similar to that individual user [6]. A CF RS as person to person correlation was introduced later. The process of recommending items to an individual is based on the extent of the interrelationship between that person and other people who have those items in their purchase history[9]. CF systems are implemented in various domains such as in: news, music, movies, jokes, books and other product domains. One of the shortcomings of the CF recommendation approach is that it must be initialized with user's tastes in order to make meaningful recommendations [10]. As a consequence, this approach experiences cold start problem in which an RS is incapable of making useful recommendations because of the absence of initial ratings when new items or new users enter the system [6] [11].

Other problems like Gray sheep user [12] and recommending alike items only [7] also persists in the RS. The content-based RS

as an item to item correlation system was introduced. It recommends items based on items with similar content items to those that a user liked before [9]. The content based approach has its origin in information filtering and information retrieval [6] [13]. Examples of such systems are: The newsgroup filtering system: NewsWeeder [14]; The web page RS: Fab [15] and Syskill & Webert [16]; The book RS: Libra [17]; The funding RS: ELFI [18]. One of the drawbacks of the content based recommender approach is a new user problem [6] [13] [19]. Another drawback is over-specialization that is, a content-based approach tends to favor items that are identical to the items rated by that user in the past [6].

A model based on deep belief network (DBN) and probabilistic graphical model bring together the two stages into a process that concurrently reads features from audio and generates personalized recommendations [20]. In comparison to existing deep learning based models, their model outperforms them in both the warm-start and cold-start stages without relying on collaborative filtering (CF).

Another approach like Back Propagation Neural Network (BPNN) can quickly drop into the local minimum point in time series prediction. A mixed approach that couples the Adaptive Differential Evolution (ADE) algorithm with BPNN, called ADE–BPNN, devised to increase the recommendation efficiency of BPNN. ADE is initially used to examine the global weights and thresholds of BPNN. Then, BPNN is employed to explore the search space for best weights and thresholds comprehensively. The ADE is used as a preliminary quest for the global best weights and thresholds of BPNN [21].

A ranked Bayesian model named as collaborative deep learning (CDL), collectively uses deep learning for the content and CF for the rating matrix. CDL can concurrently obtain competent deep attributes from content and acquire the connection and implicit bond between items. CF is used as an additional complicated target. CDL is the first ranked Bayesian model to unite the divergence between various deep learning models and RS [22].

An another novel idea like Wide & Deep learning — collectively trained deep neural networks and linear models - to blend the advantages of generalization and memorization for RSs. The wide part is a generalized linear regression model, while the deep part is a neural network with feed forward algorithm. The idea is to use a warm-starting arrangement which starts a new model with embeddings and the linear regression model weights from the prior model [5].

[23] proposed the problem of time heterogeneous feedback recommendation. They remodeled this problem to calculating the probability of a user choosing an item in the forthcoming time given the previous responses. They used a neural network which consists of two parts: recurrent and nonrecurrent part. The recurrent part recalls the impact of the previous responses. And the non-recurrent part depicts the fundamental user choice.

Tag information for RSs can be used to boost the execution of conventional recommendation approach. However, individuals defined tags will frequently experience lots of issues, such as redundancy, ambiguity, and sparsity [24]. To target issues as mentioned earlier, they suggest a brand-new recommendation approach using DNNs where user's profiles are originally characterized by tags, and then a DNN model is employed to obtain the in-depth attributes from tag field piece by piece. Hence, depictions of the data would turn further abstract and progressive, and therefore the particular arrangement of tag field will be affirmed naturally. Based on those obtained conceptual attributes, user's profiles are amended and used for aggregating recommendations.

An innovative idea focused on characterizing environmental attributes as unsupervised low dimensional hidden attributes, selects info from a collection of mobile sensors to deduce unknown user attributes [25]. The hidden attribute patterns are modeled as arithmetic vectors which are conveniently obtained from raw sensor data. The hidden attributes are frequently studied for individual user exploiting unsupervised deep learning approaches and PCA on the info received from the mobile phone of the user. The data obtained from the mobile phone sensors is organized into a hidden context-aware attribute by which there is a boost in accuracy. [26] came up with an evolutionary technique, called Invenire, to automate the decision of approach utilized by bringing together outcomes of different recommendation approaches. The Genetic Algorithm(GA) is employed as a search algorithm to enhance the results sequence of CF-approaches. The intent of the GA is to procure a favorable result for the issue of picking up the suitable number of components over each of the particular ranking generated by the basic approaches. The preferred components are used to generate the last ranking. As a consequence, each location in a candidate solution depicts a chunk of a base ranking to be considered in the final ranking.

Features can be exploited in RNN based session models using deep learning to get better results. They show that distinct

approaches do not leverage these data sources. Thus they introduced many parallel RNN (p-RNN) architectures to model sessions based on the clicks and the features (images and text) of the clicked items. The algorithms in this work utilize deep learning techniques both to extract high-quality features from visual information and to model the sessions [27].

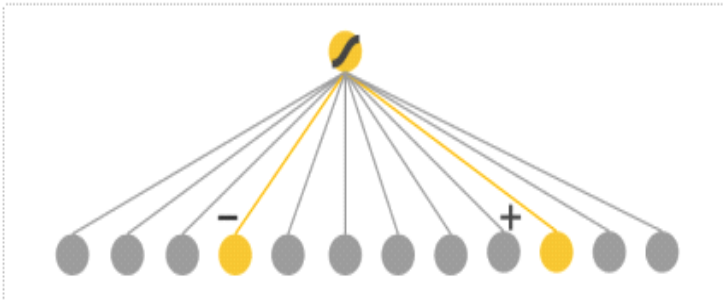

Figure 1. Wide Model

A different approach is to use the inherent features to explain the behaviors of users and capture the properties of items. As users interact with the various items over time, user and item features can influence each other, evolve and coevolve over time [28]. To accurately capture the fine-grained nonlinear co-evolution of these features, they use a recurrent co-evolutionary feature embedding process model, which combines RNN with a multidimensional point process model. To recommend items, [29] first learn a user-independent high-dimensional semantic space in which items are positioned according to their substitutable and then learn a user-specific transformation function to transform this space into a ranking according to the user's past preferences. An advantage of the proposed architecture is that it can be used to effectively recommend items using either content that describes the items or user-item ratings. They firstly structure items in a semantic space and then for a given user learns a function to transform this space into a ranked list of recommendations that matches the user's preferences.

**3. Proposed Recommender System Architecture**

One challenge in RSs is to accomplish both generalization and memorization. Memorization could generally be described as processing the persistent co-existence of elements or attributes and utilizing the interrelationships available in the actual data. Generalization, else ways depends on logical relation and examines new attribute sequences that have not at any time or hardly appeared previously. Memorization of attribute interplay over a broad batch of cross-product attribute conversions is compelling and explainable, while generalization needs additional attribute engineering exercise. With limited attribute engineering, DNNs can generalize exceptionally to undiscovered attribute sequences. However, DNNss can overgeneralize and recommend lesser suited things. Wide & Deep learning collectively trains linear models and deep neural networks can fuse the gains from memorization and generalization for RSs. The wide part is a linear regression model of the form as shown in equation 1 and model representation in Figure 1.

$$y = w^T x + b \tag{1}$$

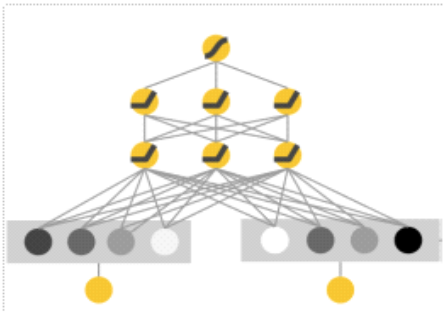

Figure 2. Deep Model

$y$ is the prediction vector, and $x$ is an input vector of $t$ features as shown in equation 2.

$$x = [x_1, x_2, \ldots, x_t] \tag{2}$$

*w* represents the parameters for the underlying model as shown in equation 3 and *b* is the bias parameter.

$$w = [w_1, w_2, \ldots, w_t] \tag{3}$$

The deep part is a neural network with feed-forward algorithm, as shown in Figure 2. Precisely, each hidden layer calculates activations according to the equation 4:

$$a^{(n+1)} = f(W^{(n)}a^{(n)} + b^{(n)}) \tag{4}$$

where *n* denotes the layer number and f denotes activation function. $b^{(n)}$, $a^{(n)}$, and $W^{(n)}$ are the bias, activations and model weights of nth layer. The wide part and deep part are connected by applying a weighted sum of their output logs as the prediction, which is input to one simple logistic loss function for collective training. Training of the proposed model is done by back propagation of the gradients computed from the result of wide and deep part using mini-batch stochastic optimization collective training of a wide & deep model, as shown in Figure 3.

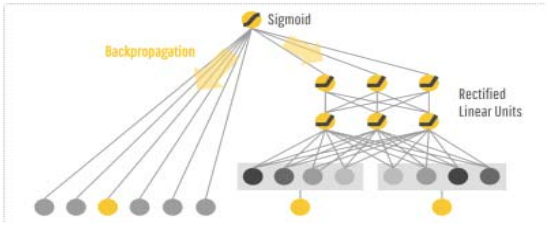


Figure 3. Wide and Deep Model

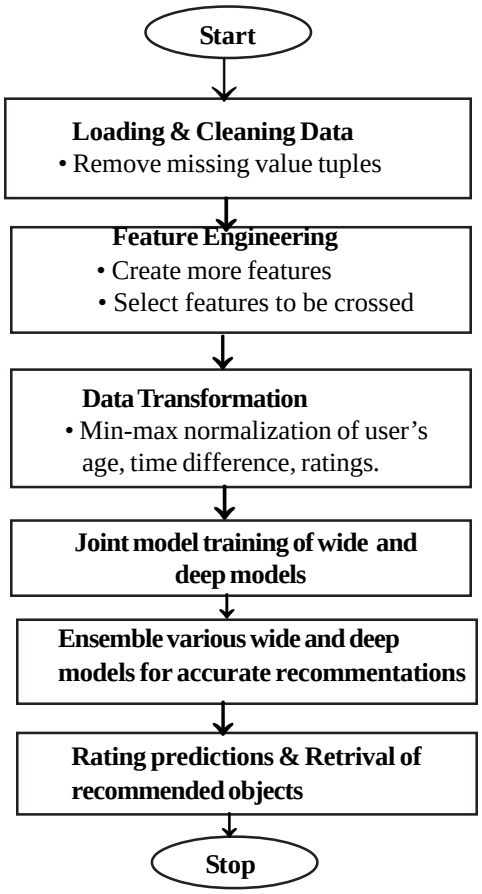


Figure 4. Work Flow diagram

The overview of the complete process of recommending the objects is depicted by the flow diagram in Fig 4.

### 3.1 Loading Data

In this work we use MovieLens 100K dataset for demonstration purpose, but proposed RS would work with any data containing information about users, items, and user-item ratings.

We first acquire information about every user from 'u.user' file, movies from 'u.item' file, and every user-item rating from 'u.data' file. Then we merge all the information from above the files to make a single data frame. We take into consideration only certain information about users and movies such as 'user id', 'age', 'occupation', 'movies id', 'movie release date', 'genre information', 'date when user rated a movie'. The above data is already split into 5-fold cross-validation data. We load this data one by one and perform computations.

### 3.2 Feature Engineering

Next step is to create more features for better prediction. In wide linear models, we use crossed columns. There are lots of combinations of crossed columns, so only those columns which are highly associated can be useful. To find the degree association we use Pearson Correlation Coefficient. After careful analysis, we find various combinations of crossed columns. We have movie release date and date at which a particular user rated a movie, both of these features do not constitute as a useful feature on their own, but if we use the difference between the rating of a movie and its release date, it could be very helpful. So we used this difference as a feature. We also have users age which is useful, but it could be more useful if we also make it categorical. So we also use age bucket as a feature.

### 3.3 Data Preprocessing

In this step, we firstly remove all those rows which have missing values, which were very few. Then we apply minmax normalization on users age and difference between movie release date and date at which user rated a movie. After min-max normalization, we scale them down between 0 and 1 inclusive. We know that all users behave differently, so different users rate the same movie differently even if they all like or dislike that movie. For instance, there are two users A and B. They both like a movie M1, A rates this movie as 5 but B rates this movie as 4. There is another movie M2, they both dislike M1, so A rates this movie as 1 but B rates this movie as 2. Certain users refrain themselves from giving the highest or lowest ratings. So we should normalize ratings for those users. We have also taken care of this behavior of the users.

### 3.4 Model Training

All along the training, our input layer receives training data and produces sparse & dense attributes simultaneously along a label. For the deep component, an embedding vector is obtained for every categorical attribute, whereas wide part comprises of cross-product conversion of person's profile and movie's information. We integrate all embeddings collectively along the dense attributes. The finally chained vector is next used as an input to Rectified Linear Unit (ReLU) layers, & lastly to an output unit which uses the logistic function. The wide and deep part are trained on over approximately 80K tuples.

Observe that there is a difference between collective training and ensemble. In ensembling of models, each model is trained independently without insightful of one another, & their prediction probabilities are summed up entirely at interpretation point but not at training point. In comparison, collective training enhances complete parameters concurrently by considering together the wide & deep component along with their weights during training time. There are consequences upon model size too: Since in ensembling, the training is independent, so every particular model size generally needs to be bigger in order to accomplish acceptable prediction accuracy. In contrast, collectively training wide component particularly requires overcoming the shortcomings of deep component using a limited number of cross-product attribute conversions, alternatively than using only a wide model. Collective training of a wide and deep component is accomplished through backpropagating the gradients originating at the output of both wide and deep component of the final model concurrently by applying mini-batch stochastic gain. In our experiments, we chose follow the regularized leader(*FTRL*) algorithm along *L1* regularization as an optimizer for the wide component, and Adaptive Gradient Descent(*AdaGrad*) algorithm for the deep component.

After training various wide and deep models with a different set of hyperparameters, we use ensembling techniques such as model averaging to increase the prediction accuracy of recommendations further.

## 4. Experiments and results

For evaluation purpose, MovieLens 100K dataset[30] has been used which consists of *100,000* ratings (*1-5*) on *1682* films from *943* individuals. Each individual has appraised no less than *20* films. We have elementary statistical information for the individuals (*age, sex, profession, zip*). The information was gathered through the MovieLens site amid the sevenmonth time frame from *September 19th, 1997* through *April 22nd, 1998.*

### 4.1 Wide Model

In this experiment, we use wide linear model with *FTRL* as an optimizer, regularization parameters *L1 = 0.01* and *L2 = 0.01* and *Learning rate = 0.1*. Various feature columns like age, occupation, movie genres and their cross-product transformations are given as an input. Fig 5 depicts the Accuracy, MAE and RMSE for wide model.

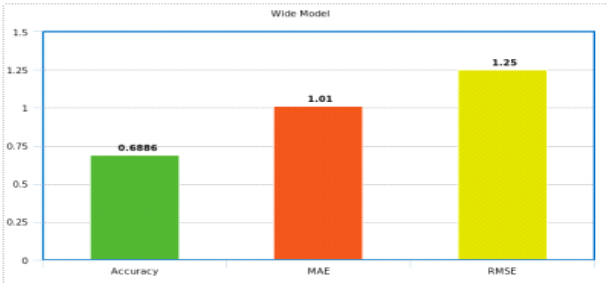


Figure 5. Wide Model Results

### 4.2 Deep Model

In this experiment, we use DNNs as a deep model with *AdaGrad* as an optimizer, regularization parameters *L1 = 0.001* and *L2 = 0.001*, *hidden layers = 3*, Units in *hidden layers = [64, 32, 16]* and Learning *rate = 0.1*. Various feature columns like age, occupation, movie genres and time difference between rating a movie and its release date are given as an input. Figure 6 shows the Accuracy, MAE and RMSE values computed for the deep model.

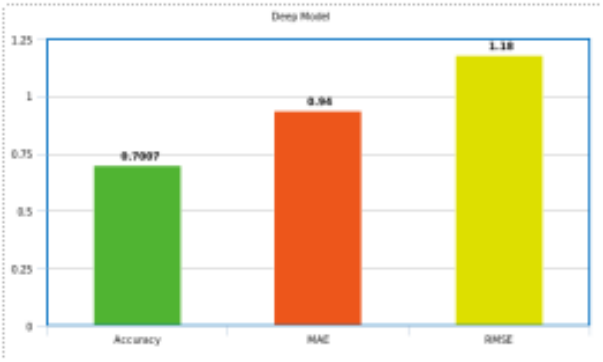


Figure 6. Deep Model Results

### 4.3 Wide and Deep Model

In this experiment, we use DNNs as a deep model with AdaGrad as an optimizer and wide linear model with FTRL as an optimizer, regularization parameters *L1 = 0.001* and *L2 = 0.001*, hidden layers = 3, Units in *hidden layers = [64,32, 16]* and Learning rate = 0.1. For the wide model we use linear regression with *FTRL* as an optimizer, regularization parameters *L1 = 0.01* and *L2 = 0.01* and *Learning rate = 0.1*. Various feature columns like age, occupation, movie genres, their cross-product to wide linear model. All feature column including age, occupation movie genre and time difference between rating a movie and its release date are given as an input to deep model. Fig 7 depicts the Accuracy, MAE and RMSE values computed for wide and deep model.

The objective of our analysis was to assess the quality and exhibition of the recommendations given by our RS Architecture. With a specific end goal to assess the diversity and nature of the top-N recommendations and to experimentally prove that our RS produces better recommendations, we contrast our system with some of the existing RS.

We use RMSE, MAE, and Accuracy of recommended items as the metrics to check how our proposed RS performs as compared various other RSs.

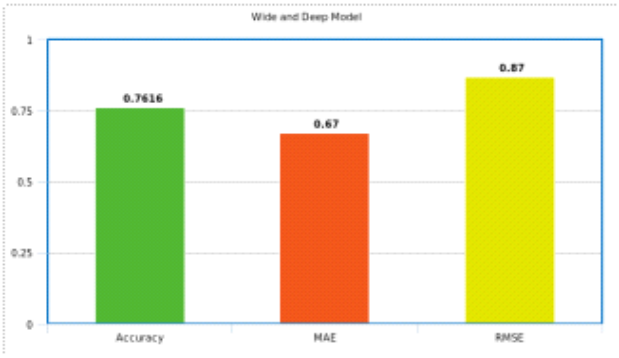


Figure 7. Wide and Deep Model Results

Fig 8 shows RMSE values of different models, we can clearly see that wide and deep model has a lesser value of RMSE as compared to different models.

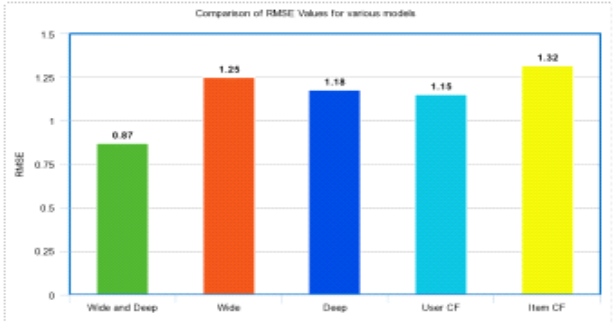


Figure 8. Root Mean Square Error Comparison of different models

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_i - \bar{c}_i)^2} \quad (5)$$

The Fig 9 shows MAE values of different models; we can conclude from the values of bar graphs that wide and deep model performs better than any other model as MAE values for our model is lesser.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\text{rating}_{predicted} - \text{rating}_{actual}\right| \quad (6)$$

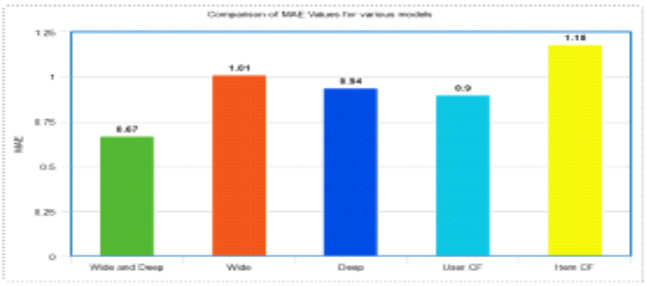


Figure 9. Mean Absolute Error Comparison of different models

The Fig 10 clearly shows an accuracy gain in prediction of recommended items for a user.

Our RS also focuses on the cold-start problem and gray sheep problem. To tackle the cold-start problem, we classify items into two groups cold-start items and non-cold start items. For recommending cold-start items, we calculate their similarity with non-cold start items. Based on similarity, we calculate the ratings of cold start items. After calculating the ratings, we recommend these cold start items together with non-cold start items.

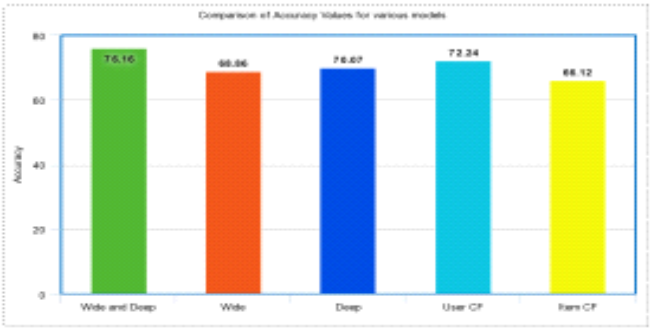


Figure 10. Accuracy Comparison of different models

As our model is a collectively trained wide and deep model, we achieve both generalization and memorization together. As gray sheep user's preferences do not match very closely to a single user but partially to a set of users, this partial similarity is taken into account by DNN, and the wide part helps for diversification of recommendations.

**5. Conclusion**

All the traditional RS algorithms are incapable and incomplete to the user's taste and preferences. We have seen that all the algorithms used in RS give ineffective recommendations. Also, they lack in interpreting general population trends. So we have built an RS Architecture that is adequate, as it deals with the problems incurred by all other models used for RS. Our proposed system collectively trains both deep neural networks and linear models to produce efficient RS which would help to solve the gray sheep problem and the cold-start problem. Our experiment reveals that our RS successfully overcomes the problems associated with the traditional RS algorithms. Specifically, the contribution of this paper is as follows:

- Propose a solution to the cold-start problem.
- Overcome the problem of false nearest neighbors.
- Achieve both generalization and memorization togetherby collective training.
- Propose a solution to the gray sheep problem.
- Avoid the problem of recommending only similarproducts.

**6. Future Work**

In the near future, the emphasis would be given to test our RS on different data sets to validate our hypothesis that our RS would work in any domain of data. Also, different fields of data may require various combinations of wide and deep models so that we would build more such combinations of wide and deep models.

**References**

[1] Agrawal, R., Srikant et al, R. (1994). Fast algorithms for mining association rules, *In:* Proc. 20th International Conference on Very Large Databases, VLDB, V. 1215, p. 487–499.

[2] Han, J., Pei, J., Yin, Y. (2000). Mining frequent patterns without candidate generation, ACM SIGMOD Record, 29 ( 2) ACM, p. 1–12.

[3] Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems, *Computer*, 42 ( 8).

[4] Harrington, P., (2012). *Machine learning in action*. Manning Greenwich, CT, 5.

[5] Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir et al, M. (2016). Wide & deep learning for recommender systems, a*rXiv preprint arXiv*:1606.07792.

[6] Adomavicius, G., and Tuzhilin, A. (2005). 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) p. 734–749.

[7] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms, *In:* Proceedings of the 10th International Conference on World Wide Web. ACM, p. 285– 295.

[8] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering, *In:* Proceedings of the 22nd Annual International ACM SIGIR conference on Research and Development in Information retrieval. ACM, p. 230–237.

[9] Schafer, J. B., Konstan, J., Riedl, J. (1999). Recommender systems in ecommerce, *In:* Proceedings of the 1st ACM conference on Electronic commerce. ACM, p. 158–166.

[10] Burke, R. (2002). Hybrid recommender systems: Survey and experiments, User modeling and user-adapted interaction, 12 (4) 331– 370.

[11] Lika, B., Kolomvatsos, K., Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems, *Expert Systems with Applications*, 41 (4) 2065–2073.

[12] Srivastava, A. (2016). Gray sheep, influential users, user modeling and recommender system adoption by startups, *In:* Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 6, p. 443–446.

[13] Wei, K., Huang, J., and Fu, S. (2007). A survey of e-commerce recommender systems, *In:* Service Systems and Service Management, 2007 International Conference on. IEEE, p. 1–5.

[14] Lang, K. (1995). Newsweeder: Learning to filter netnews, *In:* Proceedings of the 12th International Conference on Machine Learning, p. 331–339.

[15] Balabanovi´c, M., Shoham, Y. (1997). Fab: content-based, collaborative recommendation, *Communications of the ACM*, 40 (3) p. 66–72.

[16] Pazzani, M. J., Muramatsu, J., Billsus et al, D. (1996). Syskill & webert: Identifying interesting web sites, *In:* AAAI/IAAI, Vol. 1, p. 54– 61.

[17] Mooney R. J., Roy, L. (2000) . Content-based book recommending using learning for text categorization, *In:*Proceedings of the Fifth ACM conference on Digital libraries. ACM, p. 195–204.

[18] Schwab, I., Pohl, W., Koychev, I.K. (2000). Learning to recommend from positive evidence, *In:* Proceedings of the 5th international conference on Intelligent user interfaces. ACM, p. 241–247.

[19] Felfernig, A., Burke, R. (2008). Constraint-based recommender systems: technologies and research issues, *In:* Proceedings of the 10th international conference on Electronic commerce. ACM, p. 3.

[20] Wang, X., Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning, *In:* Proceedings of the 22nd ACM International Conference on Multimedia. ACM, p. 627–636.

[21] Wang, L., Zeng, Y., Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, *Expert Systems with Applications*, 42 (20) p. 855–863.

[22] Wang, H., Wang, N., Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems, *In:* Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, p. 1235–1244.

[23] Wu, C., Wang, J., Liu, J., and Liu, W. (2016). Recurrent neural network based recommendation for time heterogeneous feedback, *Knowledge-Based Systems*, V. 109, p. 90–103.

[24] Zuo, Y., Zeng, J., Gong, M., Jiao, L. (2016). Tag-aware recommender systems based on deep neural networks, *Neurocomputing*, vol. 204, p. 51–60.

[25] Unger, M., Bar, A., Shapira, B., Rokach, L. (2016). Towards latent contextaware recommendation systems," *Knowledge-Based Systems*, vol. 104, pp. 165–178.

[26] da Silva, E. Q., Camilo-Junior, C. G., Pascoal, L. M. L., Rosa, T. C. (2016). An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering, *Expert Systems with Applications*, V. 53, p. 204–218.

[27] Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D. (2016). Parallel recurrent neural network architectures for feature-rich session-based recommendations, *In:* Proceedings of the 10th ACM Conference on Recommender Systems. ACM, p. 241–248.

[28] Dai, H., Wang, Y., Trivedi, R., Song, L. (2016). Recurrent coevolutionary feature embedding processes for recommendation, *arXiv preprint arXiv*:1609.03675.

[29] Vuurens, J. B., Larson, M., de Vries, A. P. (2016). Exploring deep space: Learning personalized ranking in a semantic space," *arXiv preprint arXiv*:1608.00276.

[30] Hareper, F. M., Konstan, J. A. The movielens datasets: History and context.