A Hybrid Variational Autoencoder for Collaborative Filtering

Kilol Gupta
Columbia University
New York, NY
kilol.gupta@columbia.edu

Mukund Y. Raghuprasad
Columbia University
New York, NY
my2541@columbia.edu

Pankhuri Kumar
Columbia University
New York, NY
pk2569@columbia.edu

ABSTRACT

In today's day and age when almost every industry has an online presence with users interacting in online marketplaces, personalized recommendations have become quite important. Traditionally, the problem of collaborative filtering has been tackled using Matrix Factorization which is linear in nature. We extend the work of [12] on using variational autoencoders (VAEs) for collaborative filtering with implicit feedback by proposing a hybrid, multi-modal approach. Our approach combines movie embeddings (learned from a sibling VAE network) with user ratings from the Movielens 20M dataset and applies it to the task of movie recommendation. We empirically show how the VAE network is empowered by incorporating movie embeddings. We also visualize movie and user embeddings by clustering their latent representations obtained from a VAE.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Personalization**; Clustering;

KEYWORDS

variational autoencoders, collaborative filtering, recommender systems, personalization, movie embeddings, deep learning

1 INTRODUCTION & RELATED WORK

Recommender systems have become important in today's landscape where social media and online interactions have grown in a significant way. People frequently make choices with regard to the news articles they read, products they buy, songs they listen to and the movies they watch with the help of recommender systems. All of these applications have potential new products to be discovered by users. When combined with personalized recommendations, it leads to increased user engagement and increased business profits. They also help users filter products from the plethora of options available to them. The task of generating personalized recommendations has historically been and continues to be challenging. Essentially, the task is to recommend items to users, based on user context such as view history, click-through rate, demographic information etc. and context information on items such as popularity, genre, description, reviews etc.

Collaborative Filtering (CF) has been one of the most widely used methods to obtain recommendations. CF recommends items to users based on the historical data of user-item interaction. The model-based CF includes techniques such as Latent Dirichlet Allocation (LDA) [3] and Matrix Factorization [9, 13]. However, these methods are linear in nature whereas the interaction between users and items is often non-linear.

Neural Networks have made remarkable progress in achieving encouraging results in tasks such as image processing [10], natural language processing [4], speech recognition [7] and autonomous driving [1]. The field of neural networks, more commonly referred to as deep learning now, has proven successful because of its ability to model complicated non-linear data representations [2]. The aforementioned CF algorithms try to generate a latent representation of the interaction between user and items. Better characterization of this interaction will supposedly lead to better recommender systems. There has been promising work of applying deep learning to the field of collaborative filtering [6, 11, 16, 17].

Variational Autoencoders (VAEs) have recently been adapted for the task of personalized recommendation [12], more specifically for movie recommendation using MovieLens dataset [5]. Our paper draws motivation from this work to empirically study if augmenting movie ratings with movie embeddings, result in a better characterization of the interaction between users and items (movies in this case). We do so by first using a VAE network to learn movie embeddings (in a latent low-dimensional space) and then augmenting the binarized user ratings with these movie embeddings. This mixed representation is then fed into a second VAE network that learns from a Collaborative Filtering model.

2 PROBLEM FORMULATION

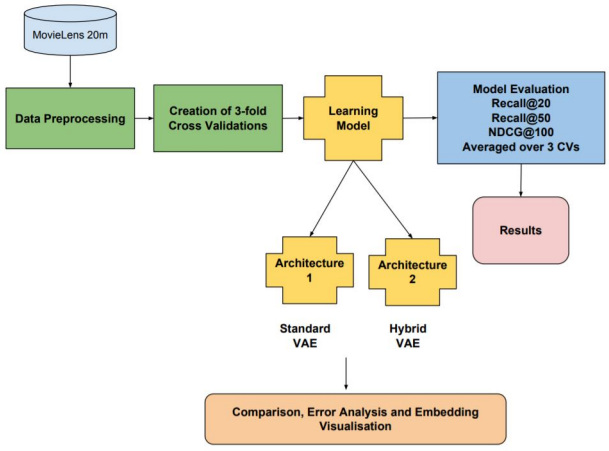

Figure 1: Project Pipeline

Our problem statement is to apply VAEs to the task of collaborative filtering for achieving a better representation of the user-item interaction. The overall project pipeline can be seen in Fig. 1

There are two versions of our implementation:

(1) Standard VAE implementation (Fig. 2)
(2) Hybrid VAE implementation (Fig. 4)

We then proceed to empirically show the improvements the Hybrid VAE implementation achieves over the Standard VAE implementation.

2.1 Dataset and Preprocessing

1. **MovieLens 20M dataset [5]**: This dataset contains 20,000,263 ratings across 27,278 movies as generated by 138,493 users. We randomly divide the set of users into training, test and validation sets with 10,000 users each in test and validation and the remaining (118,493) in training. To standardize our input, we discard those movies that did not have any information on IMDb[1], leaving a total of 26,621 movies to create a new rating dataset. The ratings are binarized with 1 for movies that the user had rated greater than 3.5, and 0 otherwise. The unseen movies are treated as being rated 0 by the user (implicit feedback).

2.2 Evaluation Methodology

We create 3-fold cross validations (CVs) of the dataset to ensure robustness of results. All the results reported in this paper are the averages over numbers obtained from the 3 CVs. We find the standard deviation across CVs to be in the order of 10^-3. The evaluation metrics are the following rank-based metrics:

(1) Recall@20
(2) Recall@50
(3) NDCG@100

Similar to [12], for each user in the test set, we compared the predicted ranks of the movies with their true ranks. The predicted ranks were decided by sorting the output of the last layer of the VAE network (softmax activation) which essentially gives a probability distribution on the movies, for each user. While Recall@R considers all items ranked within the first R items to be equally important, NDCG@R uses a monotonically increasing discount to emphasize the importance of higher ranks versus lower ones. Formally, $w(r)$ is defined as the item at rank r, $I[\ ]$ is the indicator function, and I_u is the set of held-out items that user u clicked on. Then Recall@R for user u is defined as:

$$Recall@R(u, w) = \frac{\sum_{r=1}^{R} I[w(r) \in I_u]}{min(M, |I_u|)} \tag{1}$$

We choose the minimum of R and number of items clicked by user u as our denominator to normalize Recall@R, which corresponds to ranking all relevant items in the top R positions.

The truncated, discounted cumulative gain (DCG@R) is defined below. NDCG@R is the DCG@R linearly normalized to $[0, 1]$, after dividing it with the best possible DCG@R, where all the held-out items are ranked at the top.

$$DCG@R(u.w) = \sum_{r=1}^{R} \frac{2I[w(r) \in I_u] - 1}{log(r + 1)} \tag{2}$$

Two types of evaluation schemes are used,

- Eval 1: The fist scheme is where the training is performed on 118,400 users with the testing on 10,000 users and validation on 10,000 users. The evaluation metrics NDCG and Recall are then computed for each test user over all of the 26,621 movies.
- Eval 2: In the second scheme, the training is carried out on on 118,400 users with validation on 10,000 users. In the testing procedure, the click history of each test user is divided into an 80/20 split. The binarized rating of the movies in the 20% split is set to 0 and the remaining movies in the 80% split of the test user data is left unchanged. This feature vector is then passed through the trained model to obtain a probability distribution over all the movies. The evaluation metrics NDCG and Recall are then calculated for each test user considering only the 20% split.

The second evaluation scheme is stricter and similar to the real world because of the fact that it evaluates the prediction of the model on movies which are previously unseen by the user.

3 MOVIE FEATURE EXTRACTION

One of the important aspects of this paper is to combine information from an auxiliary source to the primary user-rating information, for the purpose of collaborative filtering. This information is included by feeding it to the original VAE collaborative filtering model as an embedding input. The initial feature extraction process is done on three different levels: Movie Genres, Genome Tags and Features from the IMDb movie summaries.

3.1 Movie genres

The movie genre listings provided in the MovieLens-20M dataset are used for this category. The dataset categorizes the movies into a set of 19 genres, and each movie can belong to multiple genres. A binary encoding of all genres for each movie is created as a feature vector for a movie.

3.2 Genome tags

The MovieLens-20M contains predetermined genome tags ranging from characteristics of the plot to actors in the movies. A few examples of these genome tags are "based on a book", "computer animation", "1920" and etc. A total of 1128 tags have been used in this dataset, where, each movie is tagged with multiple tags and a relevance score is associated with each movie-tag pair. For this paper, the top 20 tags for each movie are considered based on their relevance scores and a binary vector of these top tags is created as the feature vector.

3.3 IMDb summary

3.3.1 Data Collection. The data for this category is collected using the Online Movie Database API [2] available freely, and features are extracted for 26,621 movies. Each movie is associated with a language, a movie certification, a viewer review score/IMDb rating and, the movie plot.

3.3.2 Feature extraction.

- Language: A one-hot encoding for the language of the movie is created using all languages mentioned in the dataset.
- Certification: This is a one-hot encoding for the certification given to the movie. Eg: PG-13, R etc,.

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

[2] http://www.omdbapi.com/

- IMDb rating: The score given is a continuous value that ranges from 0 to 10. This is incorporated directly without any transformation.
- Plot: The plot is analyzed and various features, describing different aspects of the text are extracted.
  - Linguistic Inquiry and Word Count (LIWC) [8]: LIWC is a lexicon dictionary which associates words in the English language to various linguistic, psychological and sociological processes. Words are mapped to a total of 64 categories in the LIWC dictionary. The plot text is tokenized and each token is mapped to a binary vector of the LIWC category it belongs to. The average vector for the whole plot is then arrived at, by averaging the 64 dimensional binary vectors for all the tokens in the plot.
  - Valence Arousal and Dominance(VAD) [15]: VAD is also a lexicon dictionary that associates words with a 3-dimensional score for the factors of valence, arousal, and dominance they depict. Here valence pertains to the pleasantness or the unpleasantness of an experience, arousal refers to the intensity of the situation and dominance relates to concepts on control. The plot text is tokenized and scores are averaged for all words in the movie plot.
  - Word2Vec [14]: The Word2Vec vectors for each word capture the semantic relation between words in an embedding space. Looking to capture the semantic differences and similarities between different movies and their plots, the averaged Word2Vec vector for the plot text is also considered as a feature. The 300-dimensional version of Word2Vec is considered for this task.

  All the above mentioned features are then concatenated to give a 671-dimensional movie feature vector.

4 MOVIE EMBEDDINGS

Incorporating a high-dimensional feature vector for each movie into an already high dimensional user-rating input is computationally not plausible. To tackle this problem, a Movie-VAE (M-VAE) is used to encode the movie feature vectors into a low-dimensional latent space. The M-VAE is trained over the movie features extracted from all 26,621 movies. The size of the latent space considered is equal to the dimension of the movie embeddings. This paper considers a movie embedding of size 3.

The movie features are encoded into these embeddings and used in the Hybrid-VAE network for the collaborative filtering task. The hyper-parameters for Movie-VAE are as follows:

- batch_size = 20
- input_dim = 671(IMDb), 1128(Genome), 18(genre)
- intermediate_dim = 50(IMDB, Genome), 10(genre)
- latent_dim = 3
- Optimizer = Adam optimizer
- Encoder-Decoder activation = ReLU
- Output activation = Sigmoid

5 IMPLEMENTATION DETAILS

5.1 Standard-VAE

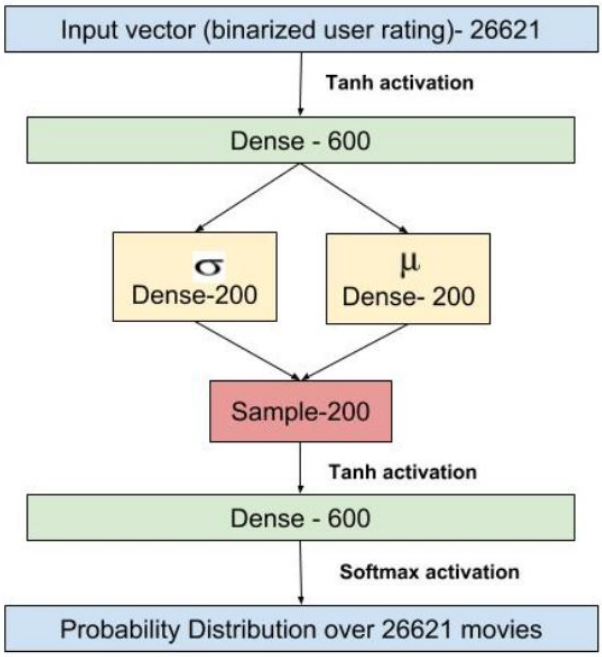

Figure 2: VAE architecture

The Standard-VAE considered in this paper takes the N-dimensional user ratings x_u as input. The user input is encoded to learn the mean, m_u and the standard deviations σ_u of the K-dimensional latent representation through the encoder function $g_\phi()$ (3) . The latent vector for each user, z_u is sampled using m_u, σ_u. The decoder function $f_\theta()$ (4) is then used to decode the latent vector from K-dimensions to a probability distribution π_u in the original N-dimension. This distribution gives us the probability of the N-movies being viewed by user u.

$$g_\phi(x_u) = m_u, \sigma_u \quad z_u \sim N(m_u, \sigma_u) \tag{3}$$

$$f_\theta(z_u) = \pi_u \tag{4}$$

The standard-VAE in this paper differs from the normal VAE which has the final output as the reconstructed input. Here, the output is a probability distribution over the K items. The objective function/loss used in the model is the ELBO[12] given in (5).

$$loss = \log p_\theta(x_m|z_m) + KL(q(z_m)||p(z_m|x_m)) \tag{5}$$

Where, x_m is the movie feature vector while z_m is its latent representation. Here, the first part of the equation considers the log-likelihood for a movie given its latent representation and the second part is the Kullback-Leibler (KL) divergence measure.

The log-likelihood function considered is given as,

$$\begin{aligned} \log p_\theta(x_u|z_u) = \sum_i x_{ui} \log \sigma(f_{ui}) \\ + (1 - x_{ui}) \log(1 - \sigma(f_{ui})) \end{aligned} \tag{6}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ taken over all the items i.

The Kullback-Leibler Divergence is calculated for the latent state of the model, z_u.

The plot training (loss) and validation loss (val_loss) vs number of epochs is given in Fig. 3. As expected, the training loss keeps on decreasing while the validation loss starts to increase after a certain number of epochs.

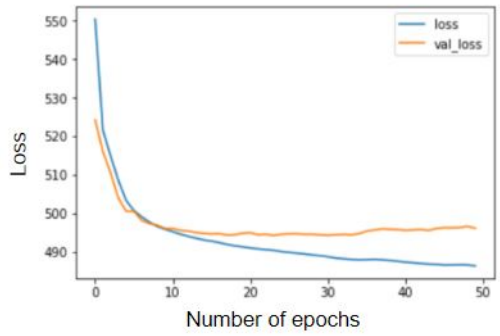

Figure 3: Training and Validation Loss (ELBO) vs Number of Epochs (on MovieLens Dataset)

The other network hyper-parameters used are:

- batch_size = 500
- input_dim = number of movies (for movielens dataset) or number of songs (million playlist dataset). Input has been binarized for both datasets.
- intermediate_dim = 600
- latent_dim = 200
- Optimizer = Adam optimizer
- Encoder-Decoder activation : tanh activation function

5.2 Hybrid-VAE

The Hybrid-VAE (H-VAE) is similar to the Standard VAE, but it contains an extra layer which combines movie embeddings with the user ratings for each movie. The embeddings are obtained from the Movie-VAE (M-VAE). Indices of movies are maintained across the M-VAE and H-VAE so that embeddings and ratings can be matched. Movies with user-click history as 0 are assigned a zero-embedding (a 3-dim 0 vector). The architecture for the Hybrid-VAE is given in Fig. 4.

Given a user click history x_u, the embedding input x'_u, for each movie i, is given by,

$$x'_u =< e_1, e_2, e_3, ..., e_n >,$$

$$e_i = \begin{cases} Movie\_Embedding(i),\ if\ x_{ui} = 1, \\ Movie\_Embedding(0),\ if\ x_{ui} = 0 \end{cases} \quad (7)$$

The successive steps follow the same procedure as the Standard-VAE, with x'_u replacing x_u in the encoding procedure. However, the objective function still considers the input user-click history x_u instead of the embedding input x'_u. All variations for the H-VAE are run for 20 epochs. The other hyper-parameters for H-VAE are as follows:

- batch_size = 500
- input_dim = number of movies
- intermediate_dim = 600
- latent_dim = 200
- Optimizer = Adam optimizer
- Encoder-Decoder activation = Tanh
- Output activation = Softmax

It is worth noting that the embedding layer output in the H-VAE is a 3D matrix of dimensions,
($batch\ size \times num\ of\ movies \times movie\ embedding\ dimension$). The intermediate dense layer, however, requires a 2D vector as input. There are two ways to introduce the embeddings into the intermediate layer:

- Flatten the 3D embedding layer output into a 2D layer: In this case the input to the intermediate dense layer is a vector of length equal to
  $number\ of\ movies \times movie\ embedding\ dimension$
- Convert the 3D embedding into a 2D embedding using a Dense layer: In this case the input to the intermediate dense layer is a vector of length equal to the $num\ of\ movies$

To determine the best approach of the two, the model is run on the IMDb feature embeddings. The results can be seen in Table 1. It is seen that the first approach of flattening the 3D vector into a 2D vector gives better results for Recall@k but the second approach gives slightly better results for NDCG@k. The better results with Approach 1 are because of the information loss that occurs in Approach 2 while converting embeddings of size 3 into embeddings of size 1. Approach 1 is used for all the tasks further reported in the work because it had a better Recall@k and only a slightly worse NDCG@k.

Measure	Approach 1	Approach 2
Eval 1 : NDCG@100	0.270	**0.271**
Eval 1 : Recall@20	**0.541**	0.539
Eval 1 : Recall@50	**0.573**	0.568
Eval 2 : NDCG@100	0.181	**0.183**
Eval 2 : Recall@20	**0.214**	0.211
Eval 2 : Recall@50	**0.377**	0.369

Table 1: Comparison between the approaches for handling the embedding layer

6 VISUALIZING EMBEDDINGS

To better appreciate the working of a Variational Autoencoder, user and movie embeddings learned from the VAE networks are visualized. For visualizing the user embeddings, the 200-dimensional latent representation of the users obtained from Standard VAE is clustered, using k-means clustering into 10 clusters. After obtaining the cluster assignments, t-Distributed Stochastic Neighbour Embedding (t-SNE) is applied to reduce the dimensionality from 200 to 2 for the purpose of visualization. As shown in Fig. 5, users do exhibit certain patterns in their movie choices which the VAE network aims to capture.

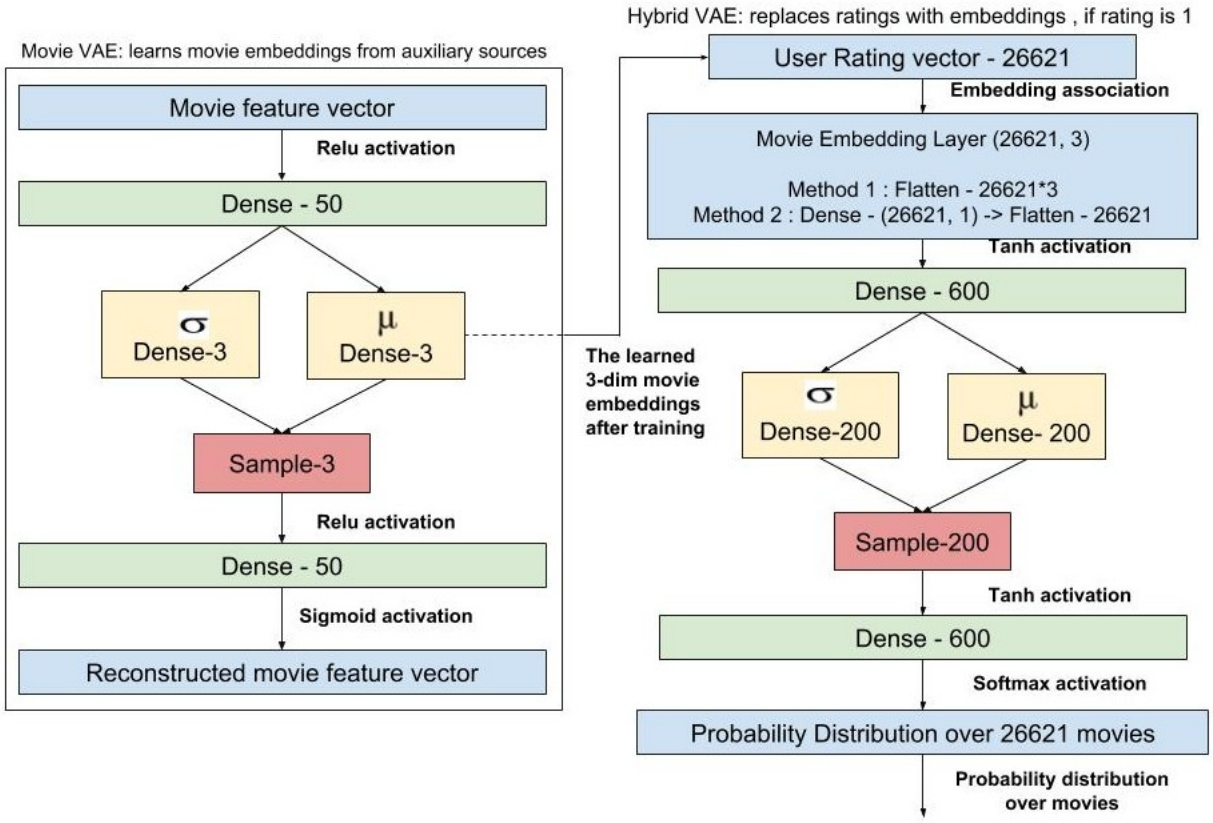

Figure 4: Hybrid VAE architecture

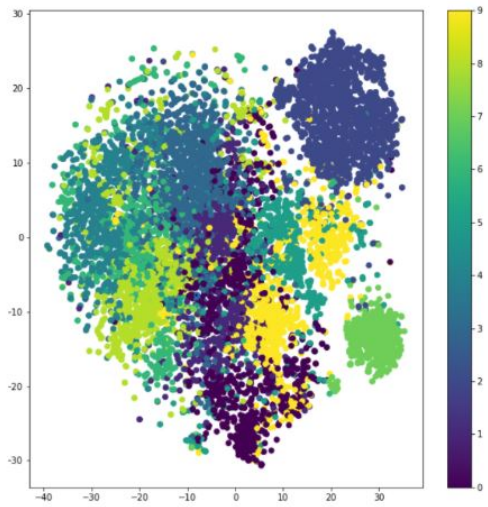

Figure 5: User Embedddings into 10 clusters

Movie embeddings are also visualized in the same way as user embeddings, Fig.6. These are obtained using the M-VAE using only genres as features, and are clustered into 18 clusters corresponding to the 18 genres. The visualization, encouragingly, shows a visible clustering.

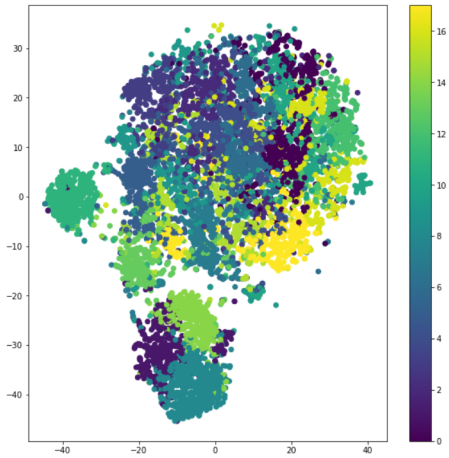

Figure 6: Movie Embedddings into 18 clusters learned using genres

7 RESULTS AND ANALYSIS

As discussed in Section 3, the model is run on the three feature sets and the most effective feature set is determined. The results in Table 2 show that the H-VAE outperforms the Standard-VAE by a good margin, thereby verifying the significance of our feature sets.

Measure	Standard-VAE	H-VAE (Random)	H-VAE (Genre)	H-VAE (Genome)	H-VAE (IMDb)
Eval 1 : NDCG@100	0.267	0.171	0.249	0.270	**0.271**
Eval 1 : Recall@20	0.534	0.297	0.501	0.537	**0.541**
Eval 1 : Recall@50	0.562	0.297	0.531	0.566	**0.572**
Eval 2 : NDCG@100	0.155	0.116	0.159	0.180	**0.181**
Eval 2 : Recall@20	0.208	0.127	0.213	0.208	**0.215**
Eval 2 : Recall@50	0.368	0.212	0.368	0.369	**0.377**

Table 2: Performance of Hybrid-VAE using different feature sets compared with Standard-VAE

The features extracted from IMDb summaries give the highest scores, followed by genome tags. Both these feature sets outperform the movie genre feature set. This shows that the genres alone, are not a powerful contextual feature to characterize movies in recommendation systems. There are certain nuances about movies that fare beyond genres and this is shown by the fact that the H-VAE with genre features as embeddings performs worse compared to the baseline Standard-VAE.

It may be possible that the features extracted from movies have no effect at all, & the increase in scores is only because of an extra layer. To verify this, the model was trained with random embeddings. The results show that a random embeddings layer does not add information to the model and it performs poorly even in comparison to the Standard-VAE. This verifies the usefulness and the relevance of the movie embeddings considered in this work.

It is also possible that due to the updates during training, the embeddings change significantly from initialization, making the movie feature extraction irrelevant. The visualized IMDb feature embeddings shown in Fig. 7, depict that the training procedure does not alter the embedding space by much. And hence it makes sense to keep this information which is in the form of movie embeddings.

The IMDb features extracted are complex and contain information on the emotion and the sentiment a movie depicts. These embeddings capture the user preferences in a better manner when paired with the user rating data. The genome tags, though detailed, do not capture this sentiment/emotion and come a close second to the IMDb features. The genre feature is a high-level representation of a movie and fails to capture user preferences which appear to be more fine-grained. The genre feature set seems to add noise to the model than help in prediction. Thus, it can be stated that the IMDb feature set improves the model accuracy because it succeeds in capturing user preferences effectively, and aids in achieving a better representation of the user-item interaction.

8 GITHUB CODE REPOSITORY

https://goo.gl/cwYrmw

Above is the link to our open Github repository that holds the implementation of Standard-VAE, Hybrid-VAE and VAE adapted for Spotify 2018 RecSys challenge on playlist completion.

9 CONCLUSION

The MovieLens dataset is considered to study the applicability of VAEs in collaborative filtering. After obtaining additional context information on the movies, this context information is introduced into a collaborative filtering model in the form of movie embeddings. This proves to be effective in increasing the accuracy of the task at hand. The context information studies is of three types: genres, genome-tags and IMDb movie features.

The results signify that adding context information to the item set can help increase the accuracy of collaborative filtering. The IMDb features give the best performance with regard to accuracy. The proposed method gives an intuitive and a highly flexible approach of adding high dimensional context information into a VAE network. The results also go on to show the importance and relevance of such auxiliary context information when it comes to recommendations made by automated systems.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Tony Jebara for providing constructive feedback in helping us shaping this work and in reaching a meaningful conclusion.

REFERENCES

[1] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab, and Hayder Radha. 2017. Deep learning algorithm for autonomous driving using GoogLeNet. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 89–96.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[5] F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2016), 19.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[7] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

[8] Ryan L. Boyd James W. Pennebaker, Roger J. Booth and Martha E. Francis. 2015. Linguistic Inquiry and Word Count. *Pennebaker Conglomerates,Austin, Texas* (2015).

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[11] Wonsung Lee, Kyungwoo Song, and Il-Chul Moon. 2017. Augmented Variational Autoencoders for Collaborative Filtering with Auxiliary Information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1139–1148.

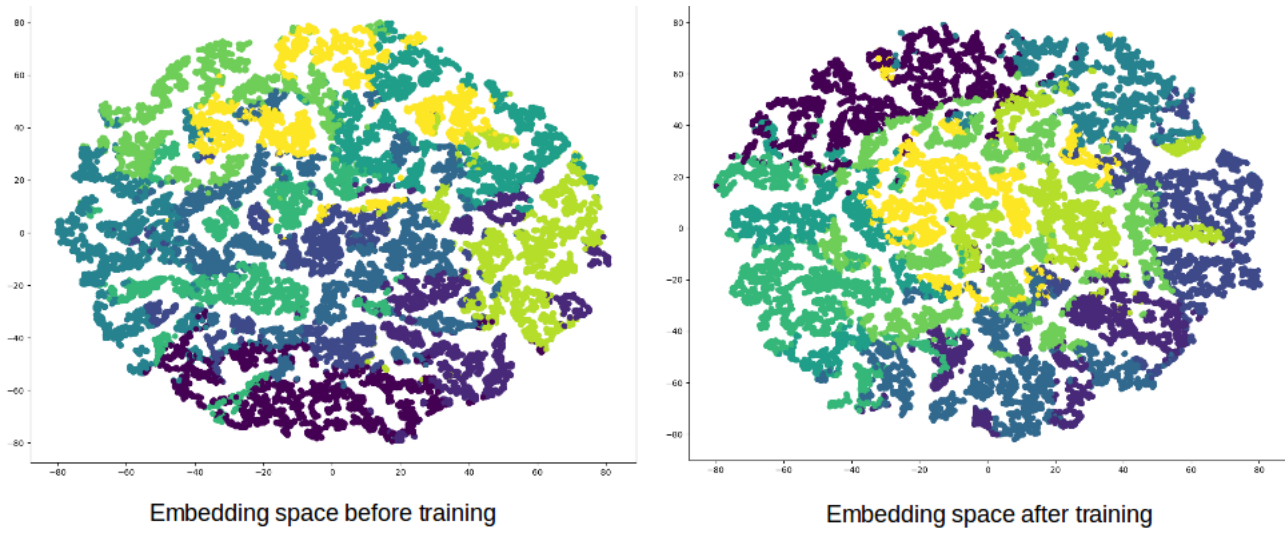

Figure 7: Comparison of embedding spaces before and after training

[12] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. *arXiv preprint arXiv:1802.05814* (2018).

[13] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[14] Kai Chen Greg S. Corrado Tomas Mikolov, Ilya Sutskever and Jeff Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[15] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior research methods* 45, 4 (2013), 1191–1207.

[16] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[17] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. *arXiv preprint arXiv:1605.09477* (2016).