ContentNCF: Content Based Neural Collaborative Filtering

Anup Anand Deshmukh
School of Computer Science
University of Waterloo
aa2deshmukh@uwaterloo.ca

Anupa Murali
School of Computer Science
University of Waterloo
a23mural@uwaterloo.ca

Udhav Sethi
School of Computer Science
University of Waterloo
udhav.sethi@uwaterloo.ca

Abstract

Some recent works have shown promising results in using deep learning, in particular convolutional neural networks (CNN), for collaborative filtering (CF) [1, 2]. These approaches employ deep neural networks to learn high order implicit correlations between users and items.

Up until now, deep learning based methods in collaborative filtering have focused on learning user-item interactions solely based on IDs as item representations. Modeling user-item interactions based on more extensive item features can provide more context with which we may recommend items to a user. By including content features, we can implicitly learn properties of items that similar users may prefer.

In this work, we extend Neural Collaborative Filtering (NCF) [1], to content-based recommendation scenarios and present a CNN based collaborative filtering approach tailored to image recommendation. We build upon the Pinterest ICCV dataset used in [1] so as to include image features, and use it to make content-based image recommendations. This content-based approach, ContentNCF, proves successful in predicting user item interactions on our new Pinterest Image 2019 dataset.

1 Introduction

There is a large amount of visual information available in today's online world. Users are constantly faced with the dilemma of sifting through a large volume of data to find relevant and novel content. Recommender Systems (RS) aim to solve this by predicting the user's rating on an item, or recommending items to users based on their preferences.

Based on a user's profile and their interactions with past items, recommender systems find new items which may be of relevance to them. In general, these recommendations are generated based on user preferences, item features and past user-item interactions. Many RSs exploit textual data to recommend items and content to users, even in the presence of image content. Due to the abundance of visual information online today, there has been recent success in making recommendations based on visual features in addition to user features and textual data for items [3].

The problem of data sparsity that often plagues recommender systems is due to insufficient information on a user's preferences. Content based image recommendations may solve this problem by

Submitted to Introduction to Machine Learning (CS480/680, 2019 Fall). Do not distribute.

extracting latent information about the user's preferences from images. Our work will build upon existing literature on CNN based collaborative filtering [2, 4] and extend them to make content based image recommendations.

It is known that a convolutional neural network (CNN) is good at learning local features, and the number of feature maps can reveal multiple aspects of local dimension correlations. ConvNCF [4] and SECNCF [2] employ CNNs to learn high-order correlations among embedding dimensions. Both these works have shown promising capability to handle content information and mention the possibility of future work in this direction.

2 Related Work

Among different techniques used by RSs, collaborative filtering (CF) [5] is one of the most popular. The goal of a CF algorithm is to suggest new items or to predict the utility of a certain item for a particular user based on the user's previous likes and the opinions of other like-minded users. The opinions of users can be obtained explicitly from the users, i.e., ratings and reviews, or through implicit feedback, which indirectly reflects users' preference through behaviours like watching videos or clicking items. In the next section we will discuss few CF based methods which motivate our project idea.

2.1 Matrix Factorization (MF)

Many areas in machine learning use the idea of learning latent (hidden) factors. The MF represents both items and users by vectors which are obtained from their interaction matrix [6]. The high correlation between these user and item factors lead to a recommendation. In these set of methods, users and items are mapped to a joint latent space of dimensionality k.

Each item i is associated with a vector $\mathbf{q}_i \in \mathbb{R}^k$, and each user u is associated with a vector $\mathbf{p}_u \in \mathbb{R}^k$. The elements of a $\mathbf{q}_i$ would capture the extent of latent factors of that particular item. The user's interest in items is captured by the elements of $\mathbf{p}_u$. Finally, the dot product between $\mathbf{q}_i^T$ and $\mathbf{p}_u$ captures the overall interest of user u in the item i. Following equation captures the above mentioned steps.

$$\hat{r}_{ui} = \mathbf{q}_i^\top \mathbf{p}_u \tag{1}$$

To learn the latent factors, $\mathbf{p}_u$ and $\mathbf{q}_i$, the method involves following minimization of the regularized squared error on the set of known ratings where L is the training set.

$$\min_{\mathbf{q}^*, \mathbf{p}^*} \sum_{(u,i) \in L} \left(r_{ui} - \mathbf{q}_i{}^\top \mathbf{p}_u\right)^2 + \lambda(||\mathbf{q}_i||_2^2 + ||\mathbf{p}_u||^2) \tag{2}$$

Recent works have shown limitation of MF caused by the use of simple inner product to capture the complex user-item interaction in the latent space. Recently, this has motivated researchers to use Deep Neural Networks (DNNs) to model this interaction function and we are going to exploit the same for the task of content based image recommendation.

2.2 Neural Collaborative Filtering (NCF)

Deep Learning (DL) has shown to be promising for problems in recommender systems. This includes, in a broader sense, an opportunity to reinvent the way we handle the user-item interactions. Mainly, DL methods are able to capture the non-linear relationship between users and items effectively. Xiangnan He et al. [1] proposed one such seminal method called Neural Collaborative Filtering (NCF) which effectively captures the non-linearities in user-item relationships by adopting a multilayer representation. NCF's predictive model can be formalized as equation 3,

$$\hat{y}_{ui} = f(\mathbf{P}^\top \mathbf{v}_u^U, \mathbf{Q}^\top \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f) \tag{3}$$

where $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$, denote the latent factor matrix for users and items. Also, $\mathbf{v}_u^U$ and $\mathbf{v}_i^I$ represent feature vectors for user u and item i respectively. f is a neural network and Θ_f

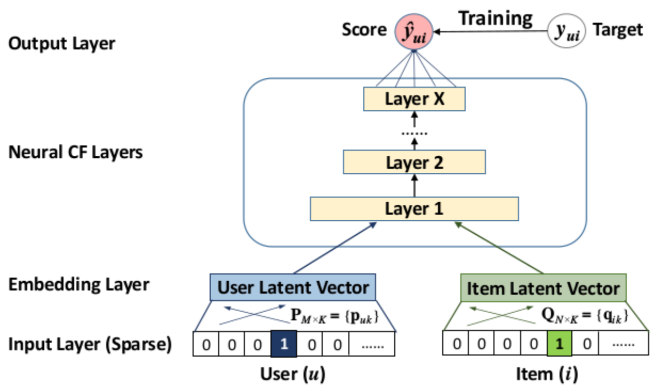

Figure 1: Framework for Neural Collaborative Filtering [1]

is the model parameter. Since ϕ_x is the mapping function for the x-th layer as given in Fig 1, f can be formulated as equation 4. To learn the model parameters authors of NCF use straightforward regression with squared loss.

$$f(\mathbf{P}^\top \mathbf{v}_u^U, \mathbf{Q}^\top \mathbf{v}_i^I) = \phi_{out}(\phi_X(..\phi_2(\phi_1(\mathbf{P}^\top \mathbf{v}_u^U, \mathbf{Q}^\top \mathbf{v}_i^I)) \tag{4}$$

It is interesting to note that, stacking more layers in NCF is proven to be beneficial to the recommendation accuracy which also encourages the effectiveness of using deep models for collaborative recommendation. There are two drawbacks in employing NCF to the task of image recommendations. Firstly, NCF simply stacks the user and item embeddings and feeds it to the DNN i.e it does not explicitly consider the correlation among embedding dimensions [4]. On the other hand, in the context of content based image recommendations, explicitly modelling the pairwise correlation between embedding dimensions becomes very important. Secondly, it does not make use of content information while performing CF and relies on user and item IDs instead. This motivates us to use the appropriate image representation along with the Convolutional Neural Network (CNN) within the NCF framework. Next section discusses our project idea in detail.

3 Our Contribution

While CF based recommendations exploit similar users' ratings, content-based recommenders recommend an item to a user based on the content and description of the item. For multimedia items like images and videos, extracted visual features have richer semantics which can capture more potential user preferences. With these richer semantics, it becomes more important to find the correlation between user and item embedding dimensions. CNNs have been shown to be beneficial for this, as the correlation among multiple dimensions could be represented by a convolutional filter [4]. To summarize, this project aims to build a content based image recommender system which exploits advantages of a CNN based NCF framework. Our contribution is two-fold as follows.

3.1 Pinterest Image Dataset 2019

Our initial plan was to use the Pinterest data used in the NCF paper to train and evaluate ContentNCF. However, it did not work out to be feasible since for our models, it is important to preserve the image content while training. The Pintereset data used in NCF simply maintains the ID numbers for images and completely ignores the URL or content information. To deal with this problem, we create our own dataset (Pinterest Image 2019) as described in the following sections.

3.1.1 Preprocessing

To create the training and testing data to suit our needs, we use the Pinterest ICCV dataset [7]. The dataset contains a mapping of boards, which represent users, to their pins, which represent images.

Users "pin" images to their own boards, showing their preferences of these images. In our dataset, each image is represented by a unique ID, as well as a URL. During training, this maintained URL for every user item interaction is used to extract the features. Each user may have consumed multiple images, and each image may have been consumed by multiple users. Also, each such user-image entry is marked as 1 or 0 indicating whether the user has consumed the item or not. This way we have four columns in our training file where each row takes the form: user ID, image ID, binary interaction, image URL. Processing all of the Pinterest ICCV data spread across multiple bson files led us to arrive at a very large dataset, surpassing our computing power. As such, we prune our dataset to arrive at a filtered dataset containing 500 users and 24498 user-image interactions.

3.1.2 Feature Extraction

With focus on the pure collaborative filtering setting, NCF uses only the identity (ID) of an item as the input feature, transforming it to a sparse binary vector with one-hot encoding. The sparse representation, $\mathbf{v}_i^I = [0, 0, 1, 0, 0, 0, \ldots]$, is mapped to a dense vector, $\mathbf{Q}^\top \mathbf{v}_i^I$, that represents an item embedding. This is then fed into a multi-layer neural architecture in NCF. We extract feature representations from the images themselves as described below.

CNN With ContentNCF, our goal is to leverage image features to get a richer feature representation than the item IDs. To obtain the content information for each image, we download the image using its unique URL and save it in a file named after its ID. Next, we face the non-trivial problem of representing the image as a vector that describes it effectively and accurately. As we know, deep learning methods have achieved the most success in computer vision, and many powerful deep models based on CNN have shown promising results in learning features from 2D image data. We use VGGNet with pre-trained weights to extract features from the image and represent it as a feature vector. VGGNet uses only 3×3 convolutional layers stacked on top of each other in increasing depth. Alog with the convolutional layer there are max pooling layers which handle the reducing volume size. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. We pop the VGG layers and remove the last softmax layer. This allows us to take the output of last fully connected layer as an image representation.

PCA The above approach leaves us with a 4096 dimensional representation of each image. Next, we use Principal Component Analysis (PCA) to reduce the dimensionality of image representations to the number of latent factors configured in our algorithm. Thus, an image is finally represented by a k-vector, which is then fed into the ContentNCF architecture described in the following section.

3.2 Content Neural Collaborative Filtering - ContentNCF

3.2.1 Architecture

See Figure 2 for an architecture diagram of ContentNCF. For the purpose of our discussion of the algorithm, let the latent space used in the Generalized Matrix Factorization (GMF) layer have dimension k, and let the first layer of the Multi-Layer Perceptron (MLP) take input with dimension d.

User Features We obtain two dense representations of the input from the sparse input vector $\mathbf{v}_u^U = [0, 1, 0, 0, 0, \ldots]$ for user u. We call these two representations $\mathbf{v}_u^{GMF} \in \mathbb{R}^k$, which we will use as input into the GMF layer, and $\mathbf{P}^\top \mathbf{v}_i^I = \mathbf{v}_u^{MLP} \in \mathbb{R}^{d/2}$, which we will input into the MLP. $\mathbf{v}_u^{GMF}$ is "MF User Vector" and $\mathbf{v}_u^{MLP}$ is "MLP User vector" from 2.

Dense Layers We use two densely connected layers, as shown in 2, to learn rich vector representations of the input image. The output of each dense layer is then passed into the GMF Layer and the MLP.

As noted in previous sections, we are using the image itself as input for item i in ContentNCF, rather than an image ID as in NCF [1]. We obtain a rich vector representation, $\mathbf{v}_{ik} \in \mathbb{R}^k$, of the image using a CNN and PCA, as described in the previous section on feature extraction.

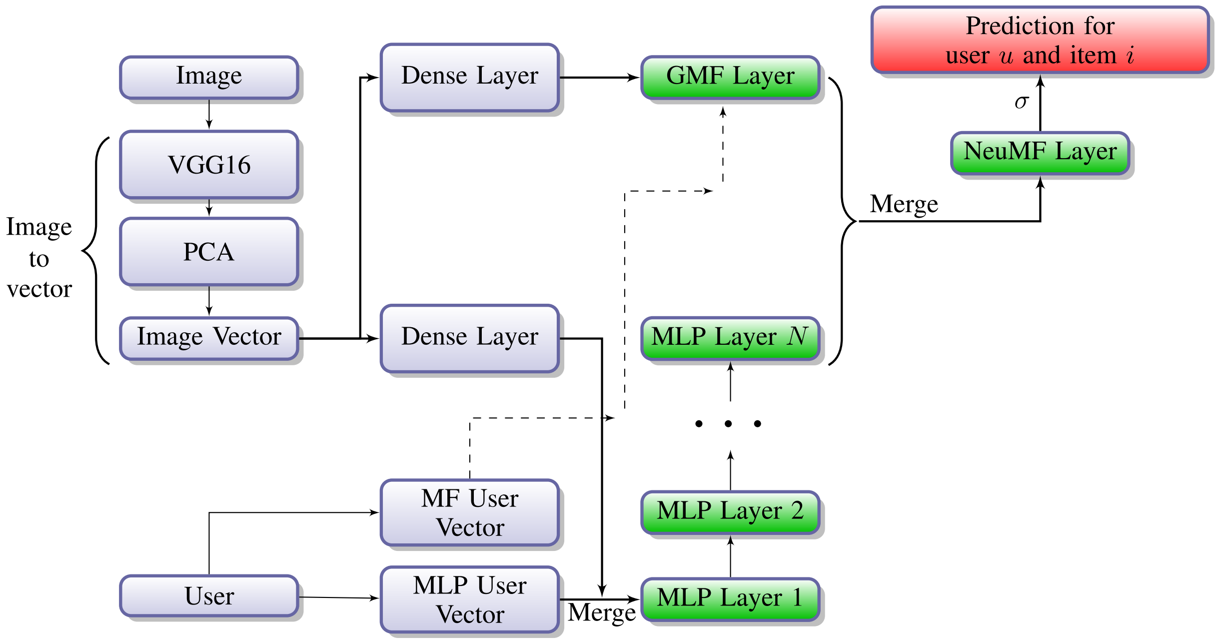

Figure 2: Framework for Content Based Neural Collaborative Filtering (ContentNCF)

To learn rich representations from our already reduced image vector, $\mathbf{v}_{ik}$, we can use densely connected layers. Note that a densely connected layer will feed each of the k features in $\mathbf{v}_{ik}$ into each of its neurons. This could ensure that we capture all of the information and patterns encoded in $\mathbf{v}_{ik}$.

For the dense layer feeding into the GMF layer, for input $\mathbf{v}_{ik}$ we let the output of this dense layer be $\mathbf{v}_i^{GMF} \in \mathbb{R}^k$.

For the dense layer feeding into the MLP, for input $\mathbf{v}_{ik}$ we let the output of this dense layer be $\mathbf{v}_i^{MLP} \in \mathbb{R}^{d/2}$. The rationale for this is that we wish to concatenate the latent vector representation of the image with user vector $\mathbf{v}_u^{MLP} \in \mathbb{R}^{d/2}$ (as per 2) to create a valid input into the first layer of the MLP with dimensions d.

GMF Layer The GMF layer computes user item interactions by computing the element wise product of its inputs, as in NCF [1]. It takes in both $\mathbf{v}_u^{GMF}$ ("MF User Vector" in 2) and $\mathbf{v}_i^{GMF}$ (the output of the dense layer which feeds into the GMF layer, as shown in 2). Note that $\mathbf{v}_u^{GMF}, \mathbf{v}_u^{MLP} \in \mathbb{R}^k$. The output of this layer will be the element wise product of vectors $\mathbf{v}_u^{GMF}$ and $\mathbf{v}_i^{GMF}$. Let us call the output of the GMF layer $\mathbf{v}_u^{GMF} \odot \mathbf{v}_i^{MLP} = \mathbf{v}_{GMF} \in \mathbb{R}^k$.

MLP Layers The MLP in our model has multiple layers can be passed in as an argument to our implementation. By using multiple layers, it learns higher order patterns and correlations between the user and image. Recall that the dimensions of the input of the first layer of the MLP ("MLP Layer 1" in 2) is $d \times 1$. We create the input into the first layer of the MLP by concatenating $\mathbf{v}_u^{MLP} \in \mathbb{R}^{d/2}$ (i.e. "MLP User Vector" in 2) and $\mathbf{v}_i^{MLP} \in \mathbb{R}^{d/2}$ (the output of the dense layer which feeds into the MLP, as shown in 2).

We concatenate $\mathbf{v}_i^{MLP}$ and $\mathbf{v}_u^{MLP}$ to create a d-dimensional vector and feed it into "MLP Layer 1". Let us call the final output of the MLP $\mathbf{v}_{MLP} \in \mathbb{R}^{d_N}$, where d_N is the dimension of the output of the N^{th} layer.

Prediction Layer Finally we concatenate $\mathbf{v}_{GMF}$ and $\mathbf{v}_{MLP}$ and pass the resulting vector into the prediction layer (i.e. "NeuMF Layer" in 2). The prediction layer is densely connected and uses a sigmoid activation function in order to restrict the output to be in $(0, 1)$. This final layer will output a prediction for the interaction between user u and item i, $\hat{y}_{ui} \in (0, 1)$. $\hat{y_{ui}}$ denotes how likely i is salient to u, and therefore should be recommended to u.

3.2.2 Learning with Log Loss

Traditional squared loss methods to learn parameters may not work well with the implicit data. This is because $y_{ui} \in \{0, 1\}$ (i.e. is binary), indicating whether or not there exists a user-item interaction, for implicit data; if we were to use squared loss, however, we are to assume that y_{ui} comes from a Gaussian distribution [8]. We use the approach used in the NCF paper by formulating ContentNCF as a probabilistic model and optimize it with the log loss:

$$\mathcal{L} = - \sum_{(u,i) \in y} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log (1 - \hat{y}_{ui}) \tag{5}$$

We use the above objective function which we minimize using the Adam optimization algorithm. During training, we randomly sample negative instances from unobserved user-item interactions. The number of negative samples per positive sample can be passed in as an argument to our implementation. Our results from varying the number of negative samples per positive sample are described in the Results section.

4 Experiments

4.1 Evaluation Methodology

The model is trained on the user-image interaction matrix. For every positive interaction of a user we add certain number (num_negative) of negative examples. To evaluate the performance of our ContentNCF we use the leave-one-out strategy. For each user, on average we have 5 interactions in the test set and rest of the interactions are utilized for training. For each such test image, we randomly sample 100 images that the user has not interacted with. The goal of ConentNCF is now to rank the positive test image among these 101 images. This common strategy is also followed in [9]. The algorithm then generates the list of top-K recommendations which is evaluated by hit ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [10]. HR checks if the test image is present in the top-K list and NDCG measures the position of the hit by assigning higher scores to hits at top ranks. We calculated both metrics for each test user and reported the average score. We have made our implementation open source and it can be found here. [1]

4.2 Results

This section demonstrates the recommendation performance of ContentNCF with different parameter settings. In the first set of experiments we show the working of ContentNCF with top-K = 5, $k = 20$ and num_negative = 4 where k is the number of latent factors used while representing a user and an image.

Table 1: ContentNCF (Top-K = 5, $k = 20$, num_negative = 4)

Iterations	Evaluation Metrics		
	HR	NDCG	Loss
init	0.049	0.0293	
1	0.4759	0.2968	0.4659
2	0.4824	0.3092	0.2831
3	0.5127	0.3295	0.2758
4	0.5135	0.323	0.2688
5	**0.5118**	**0.3222**	**0.2649**

Table 1 and Table 2 show the performance of ContentNCF after every iteration. The top-K = 5 recommendations task is more difficult than top-K = 10 and hence we get lower HR of 0.51 opposed to HR of 0.74 for top-K = 10. In further experiments we set top-K to 10.

In the next set of experiments we compare the performance by changing the number of latent factors (k). The latent factor influences the final image representation we get from PCA and higher values of

[1] https://github.com/udhavsethi/neural_collaborative_filtering

Table 2: ContentNCF (Top-K = 10, $k = 20$, num_negative = 4)

Iterations	Evaluation Metrics		
	HR	NDCG	Loss
init	0.1216	0.0534	
1	0.7282	0.3867	0.453
2	0.7273	0.3885	0.2814
3	0.7257	0.4041	0.272
4	0.7412	0.4023	0.2658
5	**0.742**	**0.4015**	**0.2629**

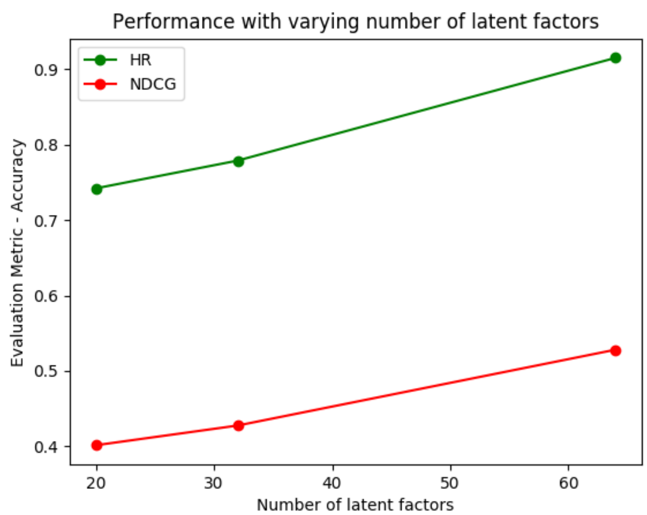

Figure 3: ContentNCF (Top-K = 10, varying k, num_negative = 4)

k would give us succinct representation of an image. Both, HR and NDCG increases as we increase k. HR jumps from 0.74 ($k = 20$) to 0.91 ($k = 64$) and NDCG reaches 0.53 for $k = 64$ (Figure 3).

Figure 4 shows the performance of ConentNCF where number of negative samples per positive sample (num_negative) ranges from 4 to 10. It can be seen that, best HR is achieved when num_negative is equal to 10. NDCG consistently increases till num_negative = 8 and then decreases further for num_negative = 10.

In our last set of experiments (Figure 5) we compare our ContentNCF with NCF [1] on Pinterest Image 2019 dataset. ContentNCF with the best parameter setting achieves HR and NDCG of 0.940 and 0.582 resp. Although ContentNCF fails to outperform NCF model in terms of accuracy measures, it has potential to give more diverse recommendations given it is a content based RS.

5 Discussion and Future Work

Instead of a simple embedding concatenation, ConvNCF [4] and SECNCF [2] employ CNNs to explicitly model the pairwise correlations between embedding dimensions. While ConvNCF applies outer product on the user and item embeddings, SECNCF combines different embeddings together in the same direction to form stacked embeddings. We plan to employ one of these techniques or their variants to model embeddings in our architecture as part of the future work.

There has been a lot of consistent effort in increasing the accuracy of recommender systems. A good recommender system should aim to improve user satisfaction apart from increasing traditional accuracy measures. Our goal in this possible extension would be to build on ContentNCF to produce diverse yet relevant recommendations by exploiting content information [11], [12]. There exist

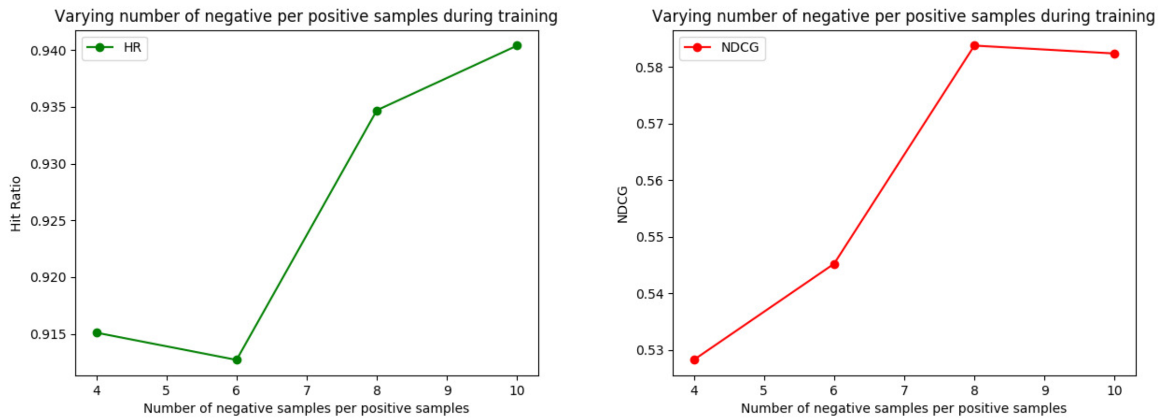

Figure 4: ContentNCF (Top-K = 10, $k = 64$, varying num_negative)

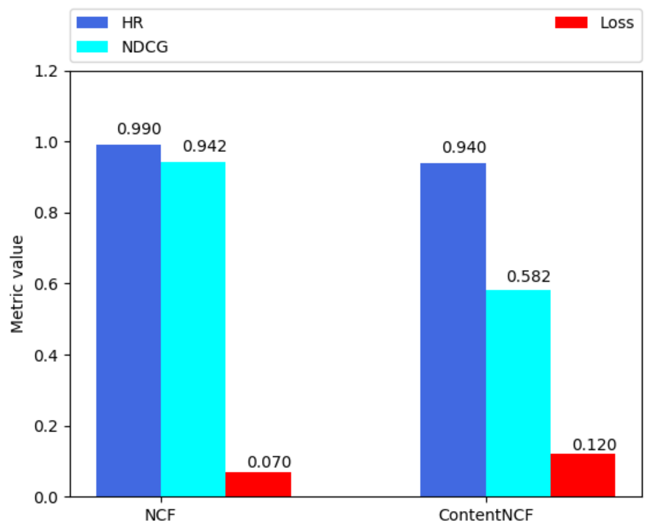

Figure 5: Comparison between NCF and ContentNCF

several ways in which we can formalize diversity. For example, in order to measure the diversity in a list of recommended items, Q_u, we can look at the pairwise distances between items in the list. We can then calculate the diversity value by averaging the normalized Hamming distance, $H(i, j)$, for all item pairs i, j in Q_u. Equation 6 can be used to calculate the diversity.

$$Diversity = \frac{2}{|Q_u|(|Q_u| - 1)} \sum_{i \in Q_u} \sum_{j \in Q_u, j \neq i} H(i, j) \tag{6}$$

6 Conclusion

We present ContentNCF, a content-based RS for learning user-item interactions, as opposed to the recent deep learning based methods which use only IDs as item representations. We propose the use of CNNs to leverage visual features from images to learn rich feature representations. We observe that ContentNCF achieves relevance prediction accuracy that is comparable with that of NCF, but also has the ability to model content information. With this ability it has potential to give more diverse yet relevant recommendations which are shown to increase the user satisfaction.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, International World Wide Web Conferences Steering Committee, 2017.

[2] L. Han, H. Wu, N. Hu, and B. Qu, "Convolutional neural collaborative filtering with stacked embeddings," in *Asian Conference on Machine Learning*, pp. 726–741, 2019.

[3] P. C. F. B. M. Y. Deldjoo, M. Elahi and A. L. E. Caielli, "How to combine visual features with tags to improve movie recommendation accuracy?," in *Proceedings of International Conference on Electronic Commerce and Web Technologies*, pp. 34–45, 2016.

[4] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," *arXiv preprint arXiv:1808.03912*, 2018.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*, pp. 230–237, Association for Computing Machinery, Inc, 1999.

[6] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

[7] X. Geng, H. Zhang, J. Bian, and T. Chua, "Learning image and user features for recommendation in social networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4274–4282, Dec 2015.

[8] L. Han, H. Wu, N. Hu, and B. Qu, "Convolutional neural collaborative filtering with stacked embeddings," in *Proceedings of The Eleventh Asian Conference on Machine Learning* (W. S. Lee and T. Suzuki, eds.), vol. 101 of *Proceedings of Machine Learning Research*, (Nagoya, Japan), pp. 726–741, PMLR, 17–19 Nov 2019.

[9] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 278–288, International World Wide Web Conferences Steering Committee, 2015.

[10] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1661–1670, ACM, 2015.

[11] A. A. Deshmukh, P. Nair, and S. Rao, "A scalable clustering algorithm for serendipity in recommender systems," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1279–1288, IEEE, 2018.

[12] D. Kotkov, J. A. Konstan, Q. Zhao, and J. Veijalainen, "Investigating serendipity in recommender systems based on real user feedback," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pp. 1341–1350, ACM, 2018.