Survey on Deep Learning for Collaborative Filtering

Neil He, Yifei Liu
qhe497, yliu523
COMPSCI 753 Group Deep Learning 2
University of Auckland

***Abstract*—The immediate impact of recommender systems can be felt all around us. As the inner workings of recommender systems begin to mature and evolve to be more sophisticated, neural recommenders have become the natural progression up from the already widely-adopted Collaborative Filtering used by the likes of YouTube and Facebook. Among the various collaborative filtering techniques, matrix factorization is the most popular one that projects users and items into a shared latent space, using a dot product of the latent user/item vectors to approximate the user's preference. Neural collaborative filtering utilizes deep learning to theoretically fit arbitrary functions on user-item interactions instead of relying on a simple dot product, allowing different features to be flexibly combined with increasing or decreasing the complexity of the model.**

In this paper, we will review the recent advances in neural recommender systems, comparing each approach's strengths and weaknesses and a summary of the current state of research in this topic alongside some of our own recommendations.

1. Introduction

The accelerated pace at which technology has been developing in recent years has led to an explosion in big data, with major tech companies embracing and further encouraging this direction of growth, we have more ways than ever to collect and utilize user data. Needless to say, recommender systems have become a mainstay in the world of Machine Learning and Big Data.

As Steve Jobs once famously said - *'People don't know what they want until you show it to them'*[1]. This is the exact problem recommender systems have set out to resolve. As an umbrella term, 'recommender system' simply describes any algorithm that has the main objective of suggesting likely **user-item** pairs. An **'user'** could be a person or groups of people, while an **'item'** could be anything ranging from a commercial/retail product like a movie or book, or academic items such as research papers to even professional advice belonging to highly technical fields such as law or healthcare.

With such a broad range of applications, significant efforts have gone towards enhancing the accuracy and reliability of recommender systems in recent years. This paper will have an overarching focus on the *Collaborative Filtering* approach - which operates off the core assumption that *two users who have agreed on an item in the past will agree again when presented with a similar item in the future*. More specifically, we will be having an acute focus over **Neural Collaborative Filtering (NCF)** methods, which has proven to be the next step in the natural progression towards better Collaborative Filtering performance.

The main advantage brought by NCF methods is its ability to incorporate deep neural network (DNN) modules, often as an additional layer atop existing architectures in order to enable the recommender system to pick up more complex *non-linear* relationships between user-item pairs. Since these DNN modules are able to capture non-linear relationships, it must also be possible for the additional NCF modules to be *generalized* to capture *linear* relationships. This is the core assumption that our seed paper - *'Neural Collaborative Filtering'*[2] operates on. The authors of the paper have successfully designed an NCF framework that utilizes both a generalized version of Matrix Factorization (MF) alongside a Multi Layer Perceptron (MLP) module, combining the capabilities of the two algorithms via transfer learning to propose a novel framework named '**NeuMF**'. Empirical evidence from their paper shows that the addition of deeper layers of neural networks offers increased performance over traditional Matrix Factorization.

The following sections of this paper will contain a brief summary of all essential concepts required to appreciate the inner workings of NCF, before further diving into the how's and why's of NCF's superior performance. Additionally, we conduct a systematic review of related deep ... to assist the reader in developing a broader understanding over the state of current NCF research, as well as some of our original ideas that arose during the creation of this report.

2. Literature Survey

We now introduce some of the concepts and models essential to this report:

2.1. History of Collaborative Filtering

The first recorded use of the term 'Collaborative Filtering' was in 1992 by Dave Goldberg and his peers at Xerox PARC[3]. An experimental mail system named 'Tapestry' was developed as a technique to handle large amounts of emails and messages sent to users by recording their reactions to the documents users have read. In this very primitive form of Collaborative Filtering, only very simple implicit feedback was used - meaning the system's only knowledge of user preference is whether the user sent a reply on a message or not. The main use case of this primitive iteration of collaborative filtering is to pick up and filter on common trends - 'If users **Alice**, **Bob** and **Charlie** all replied to a message, prioritize and mark this message as highly interesting for user **Dan**'. While it's quite intriguing to learn about the history of Collaborative Filtering (CF), is it not very practical nor sophisticated for the standards of our information-overloaded world of today, which brings us to our next topic - Matrix Factorization.

2.2. Matrix Factorization

One flavour of CF that is both highly sophisticated and has received widespread adoption in industry is*Matrix Factorization*. Originally a well-established technique called *Matrix Decomposition* within the mathematical domain, data scientists have translated the core ideas into a trainable machine learning model and it has been proven to be a model that is simple but effective time and again by the winners of the Netflix Prize[4].

The intuition behind this technique is to 'factorize' a large **preference matrix** into smaller **user** and **item** matrices P_u and Q_i containing compressed, latent representations of their respective data. Those user and item matrices can then be re-combined with a simple dot product to 'approximate' the original preference matrix. This process is repeatedly trained until a function that can replicate a near-perfect approximation of the preference matrix is reached.

$$\hat{y}_{ui} = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{1}$$

As suggested by the equation above, MF captures two way interactions between latent user-item pairs, with the assumption that their latent space is independent of one another, linearly combining them with the same weighting. As such, MF can be considered as a simple linear model of latent factors. Though it works remarkably well for how simple it is, MF does have its own limiting factors.

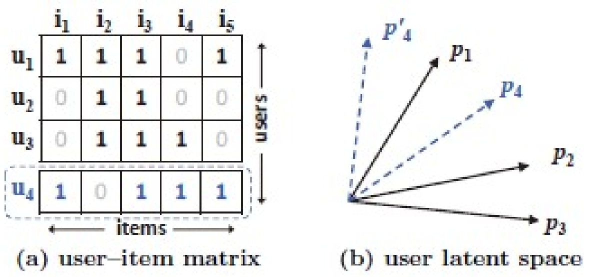


Figure 1. Illustration of MF's drawback[2]

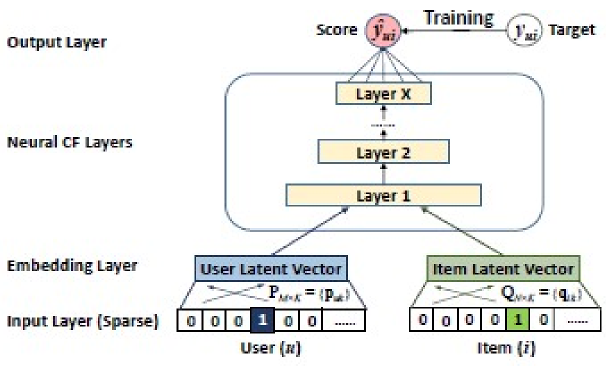


Figure 2. The General Framework of NCF

Figure 1 below demonstrates a common scenario where user 4 is evidently more similar to user 1 than all other users, however since the dot product of two vectors can be interpreted as the cosine of the angle between two vectors, the trained latent transform could place $p4$ closest to $p1$, which would actually make $p4$ closer to $p2$ in the latent space, incurring a major overall ranking loss during validation and thus demanding additional training resources until the result of the latent transform starts to resemble the position occupied by $p'4$.

This is one of the core unmitigable problems central to MF that NCF is attempting to solve. Replacing the dot product with an arbitrarily trained deep learning layer, essentially removing the problems that come packaged with the linear nature of MF, while also allowing for more intricate user-item interactions to be captured.

2.3. Neural Collaborative Filtering

The seed paper we are studying, Neural Collaborative Filtering [2], was introduced in 2017. At that moment, there were not many studies regarding deep neural learning on the recommending system. The primary purpose of this paper is to use a multi-layer deep neural network to construct the user-item interaction model. The test data-set mainly focused on implicit feedback data, for example, browsing history, purchasing history, etc. The paper used a deep neural network structure to model the latent features and designed a new collaborative filtering architecture based on deep neural network called NCF. But we are going to talk about the Matrix Factorization first.

Figure 2 shows the **general framework** for NCF. The framework uses *Multi-Layer Perceptron(MLP)* to simulate the interaction y_{ui}. The bottom layer is the **input layer** with two feature vector V_u^U and V_i^I, denotes as user u and item i. Various ways can design the input vector; it supports a wide range user and item models; for example, it can be designed as content-based, neighbor-based, etc. In the paper, we only use user and item as the input vector and use one-hot encoding to transform them into a binarized sparse vector. The cold-start problem can be resolved easily

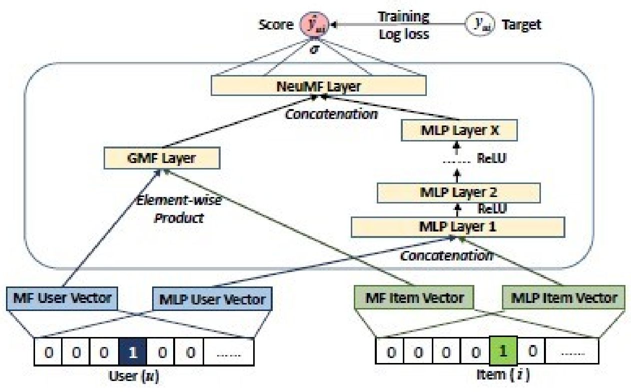


Figure 3. The General Framework of NeuMF

by using content features to represent users and items with such a generic feature.

Embedding Layer is a full connect layer to turn the sparse vector from the input layer to dense vector. User(item) embedding can be identified as the latent vector for the user(item) in the latent factor model. Then we send those embedding vectors into the multi-networking structure we term it as **Neural Collaborative Filtering Layer**, then we get the prediction score. Every layer of NCF can be customized to find out specific latent structures in user-item interaction. The size of X in the last layer decides the capability of the model. The final output layer is the prediction score $\hat{y}_{ui}$, and the goal in the paper is minimizing the point-wise loss between $\hat{y}_{ui}$ and y_{ui}. The NCF predictive model can be:

$$\hat{y}_{ui} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) \tag{2}$$

where P and Q denotes the latent factor matrix for user and item respectively; Θ_f denotes the model parameters of the interaction function f.

In paper[5] the result shows only MLP is hard to learn the inner product, and in the experiment part of seed paper[2] can show that as well, when the embedding size gets larger, the result of **MLP** will lower than **Generalized matrix Factorization(GMF)**, thus we are going to introduce **Neural Matrix Factorization(NeuMF)** which combine GMF and MLP.

2.4. Neural Matrix Factorization(NeuMF)

Figure 3 shows the structure of **Neural Matrix Factorization**. It is the fusion of *MLP* and *GMF* under the NCF framework so that they can mutually reinforce each other to better model the complex user-item interactions. They share the same embedding layer and then combine the outputs of their interaction functions. There is a special case, MF can be explained as the NCF. Since the embedding vector can be seen as latent vector, we denote $P^T V_u^U$ as the user latent vector p_u, and $Q^T V_i^I$ as the item latent vector q_i, then we can identify the mapping function for the first neural layer:

$$\phi_1(p_u, q_i) = p_u \odot q_i, \tag{3}$$

the output layer equation as:

$$\hat{y}_{ui} = a_{out}(h^T (p_u \odot q_i)), \tag{4}$$

where h denote as the edge weight, and a_{out} denote as activation function. If we assume h all one, and a_{out} is an identity function, then it is an MF model. Thus, MF is a special case in the NCF framework. If we replace the activation function with a non-linear function, then non-linear MF would be more expressive than the liner MF model. Thus, this model was named Generalized Matrix Factorization.

In the figure 3, on the right side, we can find out a standard **Multi-Layer Perceptron(MLP)**, since the simple vector concatenation in NCF can't explain any interactions between user-item latent features, this is not enough for CF. Thus, a hidden layer with standard MLP could be helpful to learn the interaction between user-item latent features.

The simple way to fuse MLP and GMF is to let them use the *same embedding layer*, then incorporate the output. However, using the same embedding layer may limit the performance of the model, for example, both MLP and GMF would face the same size of data-set, which is not the best collection. For the best result, we allow MLP and GMF to learn separate embeddings, then connect them by the last hidden layer. The NeuMF model combines the *linear* by MF and *non-linear* by DNNs for modeling the latent structure of user-item interaction.

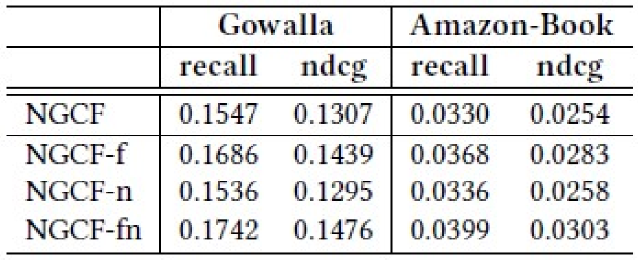
2.5. LightGCN

In this paper[6], we will look for a few questions: what is **LightGCN**, and why is it light? How was **LightGCN** implemented? And How is the performance?

What is LightGCN? LightGCN is a new graph convolution network based on *Neural Graph Collaborative Filtering(NGCF)*[7]. Compared to the NGCF, LightGCN omits the inner product part between adjacent nodes, thus speeding up the operation and ensuring the accuracy of prediction.Here is the equation of NGCF:

$$e_u^{(k+1)} = \sigma(W_1 e_u^k + \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}} (W_1 e_i^k + W_2(e_i^k \odot e_u^k))),$$
$$e_u^{(k+1)} = \sigma(W_1 e_i^k + \sum_{u \in N_i} \frac{1}{\sqrt{|N_u||N_i|}} (W_1 e_u^k + W_2(e_u^k \odot e_i^k))), \tag{5}$$

where e_u^0 and e_u^1 denote as user u embedding and item i embedding, k denote as layers propagation, σ denote as non-linear activation function, N_i denote as the neighbor item i interaction with user u, W_1 and W_2 denote as trainable weight matrix. If the current node is user e_u^0, then we can use **one-hop aggregation** to get all neighbour item embedding and do inner product with current node, then based on NGCF equation to get e_u^1.That is, the embedding of the sub-graph containing the current one-hop user node. Based on the experiment from the paper and figure 4, NGCF-fn is a NGCF but removed two feature transformation matrices and



	Gowalla		Amazon-Book	
	recall	ndcg	recall	ndcg
NGCF	0.1547	0.1307	0.0330	0.0254
NGCF-f	0.1686	0.1439	0.0368	0.0283
NGCF-n	0.1536	0.1295	0.0336	0.0258
NGCF-fn	0.1742	0.1476	0.0399	0.0303

Figure 4. NGCF performance comparison

non-linear activation function, and the recall rate and loss are getting better than NGCF, by study, we think feature transformation matrices and non-linear activation function wouldn't affect the result, thus we have the idea of Light-GCN.

In LightGCN, we use simple weighted sum aggregator and abandon the use of feature transformation and non-linear activation. Here is the graphic convolution equation:

$$\begin{aligned} e_u^{(k+1)} &= \sum_{i\in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^k, \\ e_u^{(k+1)} &= \sum_{u\in N_i} \frac{1}{\sqrt{|N_u||N_i|}} e_u^k, \end{aligned} \tag{6}$$

and follow by the layer combination and model prediction. Here is the equation of layer combination:

$$e_u = \sum_{k=0}^{K} a_k e_u^k; e_i = \sum_{k=0}^{K} a_k e_i^k \tag{7}$$

where a_k denotes as the importance of the k-th layer embedding in constituting the final embedding. In the experiment, they noticed if we set a_k to $1/(K+1)$, usually they can get a good result too. Thus, too keep the LightGCN as light as it can, there is not necessary to design anything else to optimize a_k. The prediction model for user and item is:

$$\hat{y}_{ui} = e_u^T e_i \tag{8}$$

In the paper, author also compare the performance on recall rate and ndcg between LightGCN and other similar methods, for example,NGCF, Mult-VAE[8], and etc. Also, author mentioned the ablation study is very important in recommending system recently. In this study, layer combination is important; In the network, we need combine sub-graph embedding from different hop by weighted summation method, compare to the signal GCN layer, this method could be better. And also, Layer normalisation is important as well, when calculate the one-hop aggregation, the normalisation process $\frac{1}{\sqrt{|N_u||N_i|}}$ is essential, it would affect the final result.

2.6. Session-based Recommendation with Graph Neural Networks(SRGNN)

In this section, we want to talk about the *session-based recommendations system*. Session-based recommendation means *the user information is not in the system, only based on the first action on the current session to make the prediction*. This kind of algorithm is widely used in online retailers or streaming platforms. One example could be the anonymized customer already watched **coffee machines**, **espresso machines**, and **coffee grinders**. At this moment, the system should recommend **coffee beans** to the customer. The previous three should be used for training, and the target is to predict the last one.

The mean method we want to talk about in session-based recommendation system is **Session-based Recommendation with Graph Neural Networks(SRGNN)**[9]. This method was introduced in 2018. In previous methods, each session can be represented as a continuous sequence by time, even though already used attention mechanism to capture the global information to generalize personalized representation. Still, the result from some of the perspectives is unreliable; thus, in the paper, the author introduces using GNN to trade the sequence as a graph to construct the model.

Similar to the Neural Attentive Recommendation Machine(NARM)[10], the author introduces the STAMP[11], it also captures the main purpose and the current purpose of the user. A simple example could be that when a customer wants to buy an espresso machine, he would like to check the coffee grinders, so coffee grinders are the current purpose, and espresso machine would be the main purpose. The recommendation system should consider what kind of recommendations to customers based on the current or main purpose, or we should combine them to make the suitable recommendation to customers.

Figure 5 is the model of the *workflow of the proposed SRGNN model*. From left to right, there are constricting session graphs, node representation learning, session representation generating and making recommendation, respectively.

Firstly, every session sequence can be constructed as a *directed* graph $G_s = (V_s, E_s)$, then normalize the edges in the graph based on the initial out-degree. For example, some session sequence could be $s_i = [v_1, v_2, v_3, v_2, v_4]$, the group structure could look like in the following figure 6.

The way to construct the connection matrix is to get the adjacency matrix, each row represents the start point, and the column represents the destination. If we use the figure 6 as the example for v_2 the out-degree and in-degree would be two, then do the normalize for in-degree and out-degree matrix. We can get the outgoing and incoming adjacency matrix, then combine them to the connection matrix.

Secondly, use the Gated Graph Neural Network(GGNN)[12] to learn node representation. The process is similar to the Gate Recurrent Unit(GRU)[13].

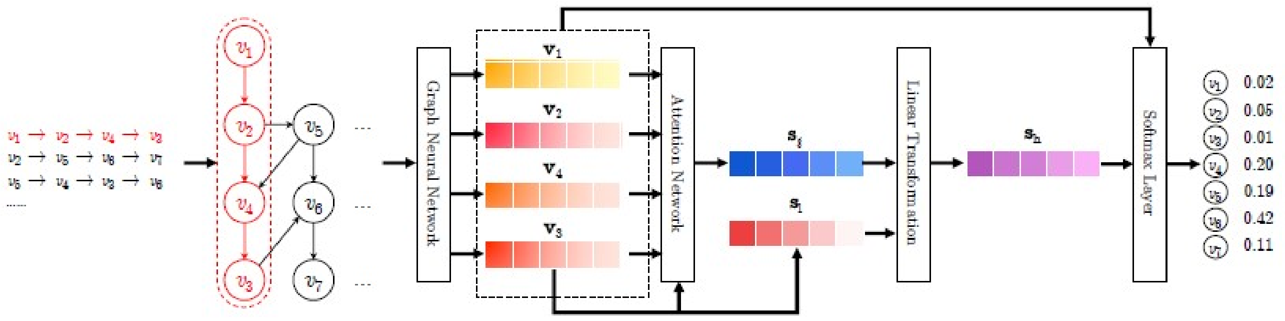
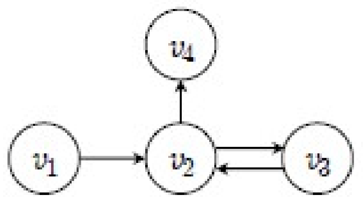
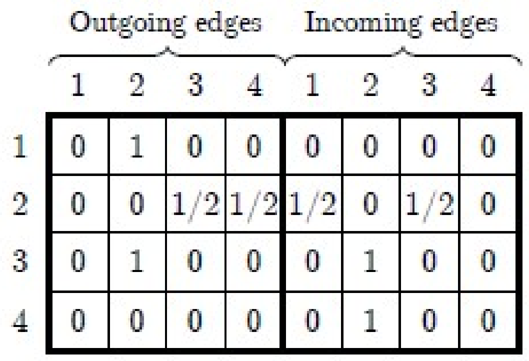


Figure 5. The workflow of SR-GNN method





	Outgoing edges				Incoming edges			
	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Figure 6. Example of session graph and connection matrix

Here are the equations:

$$\begin{aligned} a_{s,i}^t &= A_{s,i:}[v_1^{t-1},....,v_n^{t-1}]^T H + b, \\ z_{s,i}^t &= \sigma(W_z a_{s,i}^t + U_z v_i^{t-1}), \\ r_{s,i}^t &= \sigma(W_z a_{s,i}^t + U_r v_i^{t-1}), \\ \widetilde{v_i^t} &= \tanh(W_o a_{s,i}^t + U_o(r_{d,i}^t \odot x_i^{t-1})), \\ v_i^t &= (1 - z_{s,i}^t) \odot x_i^{t-1} + z_{s,i}^t \odot \widetilde{v_i^t}, \end{aligned} \tag{9}$$

The first line of the equation 9 is used for information propagation between nodes, under restrictions given by the matrix include income and outgoing. It can take out the latent vector of neighborhoods and feed them as input into the GNN. The second and third lines of equation 9 are the update and reset gate, respectively. They decide what information to be reserved and discarded. Next in equation line four, we constructs the candidate state by the previous state, the current state. The final line in the question is the combination of the previous hidden state and the candidate state, under the control of the upgrade gate. After that we can obtain the final vector.

We get the vectors of all nodes. We want to represent each session as an embedding vector, the first way is to consider the local embedding s_1 of session s. Then consider the global embedding s_g of the session grape G_s by aggregating all node vectors. Author introduced a new soft-attention mechanism to better represent the global session preference:

$$\begin{aligned} a_i &= q^T \sigma(W_1 v_n + W_2 V_i + c), \\ s_g &= \sum_{i=1}^{n} a_i v_i, \end{aligned} \tag{10}$$

Finally, from that we can get a hybrid embedding s_h by taking linear transformation over the concatenation of the local and global embedding vectors:

$$s_h = W_3[s_1; s_g], \tag{11}$$

After all that, we can use session representation s_h and item embedding v_i to get the score for each candidate item.Equation is:

$$\hat{z}_i = s_h^T v_i, \tag{12}$$

3. Discussion

In this section, we want to compare NCF with other methods we mention in the survey. We can't measure the performance under the same dataset since we don't have enough time to do it. If we get a chance next semester, we would like to do it. Thus, we can only compare the advantages, disadvantages, and other high-level points of view to analyze them.

Firstly, we take a look at the history of **Collaborative Filtering**, the most classic model in this history. The idea of it is based on the assumption, the behavior of a user can be predicted by the other user who has the same behavior as him/her. The concept is based on user and item interaction and uses group wisdom to make recommendations. Collaborative filtering can be categorized into three: *user-based CF, item-based CF, and model-based CF*. CF's problems are how to fill the unknown part, then we have the Singular

Value Decomposition method(SVD), which can lead us to the MF model.

The core idea of **Matrix Factorization** is to get the prediction score by using the inner product between user and item. The score can be used to represent the user's preference for the item. The higher the score, the higher probability that the item will be recommended to the user.

NCF and NeuFM are the match function learning model based on CF. Compared to the traditional CF, after obtaining user and item vector, it connects to the MLP network and gets an end-to-end model in the end. The advantage of this framework is flexible enough that the left and right embedding design can add any customized side info feature, and also the MLP networks can be customized as well. However, the disadvantage is the performance of MLP in NCF. In paper[5], we find out the result is the capability of MLP to learn and capture user and item vector is not as strong as we expected. Then lead to NeuMF.

NeuMF is using MF and MLP at the same time to fit the matching score. Since the goal is to make the model more flexible than before, NeuMF also GMF and MLP embedding separately, then get into the final NeuMF layer to make the prediction. From the left side and right side, we would get two vectors ϕ_{ui}^{GMF} and ϕ_{ui}^{MLP}, then put those two vectors into NeuMF, which is a non-bias full connect layer by $Z_{ui} = [\phi_{ui}^{GMF}, \phi_{ui}^{MLP}], \hat{y}_{ui} = \sigma(Z_{ui}^{T} h)$. If already pertaining by the data-set, the NeuMF layer can use *Hyperparameters* to connect those two vectors.

The idea of **lightGCN** is author argues that the feature transformation and nonlinear activation function is in GCN are not very useful for CF. It even lowers the effect of recommendation; thus, LightGCN only consists of neighbor aggregation. From a high-level point of view, lightGCN is a simple model design. And based on the test result, LightGCN is better than NGCF.

Session-based recommendation with Graph Neural Network(SRGNN) solve the problem of anonymized user recommendation issue, which means is when the user does not exist in the system, how is the system going to do the production for him/her. This method is useful because websites don't need login and somehow can't record users' watching behavior. The advantage we found out is that SRGNN uses a similar mechanism as NARM, which can capture global and current interests. Those are very helpful when making recommendations. The disadvantage is that the adjacency matrices are all 0-1 matrix; thus, we can't identify the weak or strong connection between them., but the experiment result proves that is just a manner issue.

4. Future Issue

Based on what we have covered already, we would like to explore some new avenues to expand the capabilities of deep-learning based recommender systems. We assert that given the relatively structured environments that recommender systems tend to exist in *(movie recommendation, book recommendation, product recommendation etc)*, it should be relatively effortless to set up a knowledge base that is self-maintaining. For example, we could set up a script that periodically scrapes new titles being added to Netflix and save their structured information - things like *image, title, description, actor information, and director information*. Due to the structured nature of the above information, the creation of a knowledge base could enable deep recommenders to take full advantage of its *non-linearity* properties to extract further insight from those complex user-item interactions.

Take NeuMF for example, the MLP component within simply concatenates user-item latent interaction vectors before passing it to the neural layers, however since we are no longer relying on a simple dot product, it would be possible for our model to concatenate *additional* structured information into the interaction vector to take full advantage of MLP's latent space learning capabilities, these could include things like a latent vector representation of textual reviews, or cover image, to even knowledge graphs of director/actor information. We noted that restricting the MLP side of the model to strictly basic user-item interaction without incorporating richer surrounding information seems like an enormous waste of potential, and the below section outlines the steps we propose to achieve our proposed addition.

To the best of our knowledge, incorporation of knowledge bases to **deep** RC's have yet to be proposed, however we did manage to find one somewhat relevant paper from 2016 - Collaborative Knowledge Base Embedding for Recommender Systems[14] that proposes different ways to incorporate surrounding information to collaborative filtering methods on a general basis, and we think the majority of concepts from this paper translates well into a deep RC context.

Firstly, the authors from the paper propose to project graph-structured information into a 'relationship space' via the use of **Bayesian TransR**, their proposed network embedding procedure. The final result of this procedure is that pairs that share the same relation (actor, director, extra, etc..) should be embedded into similar spaces, a different space for each relation. Figure 7 below shows a visual demonstration of the resulting 'relationship space'.

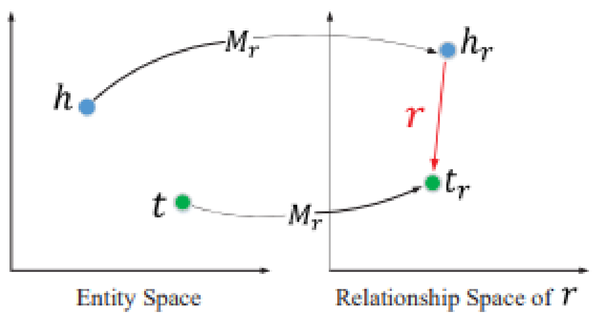


Figure 7. Visualization of relationship space between structured graph data[13]

Next, textual and visual information are proposed to be reduced into their most compact latent form via the use of appropriate **autoencoder networks**. Autoencoder neural networks serve the main purpose of transforming a full input to it's most compact representation, and also vice-versa to regenerate the full item from it's most compact representation, this means that the *middle layer of any autoencoder neural network should always be the most compact representation of data* that a neural network is able to compress, while at the same time being able to regenerate the original information. By extracting the most compressed representation vector directly from the middle layer of the autoencoder network, we should have a latent vector that is both high in compactness and high in explainability due to the autoencoder's regenerable nature. Figure 8 below illustrates the core concepts behind autoencoder networks.

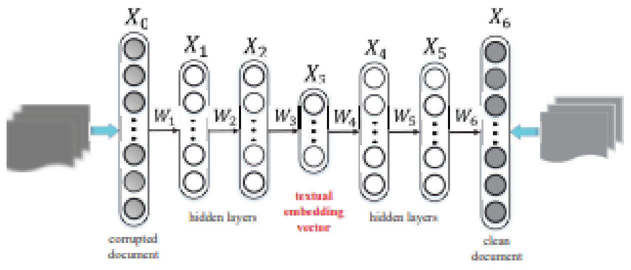


Figure 8. Visual representation of autoencoder compression/regeneration processes[13]

Once all the latent representations have been generated, we can simply slot them into the NeuMF component where user-item vectors are concatenated together - except this time also concatenating the latent vectors we have obtained above via the user of knowledge base embedding. We think the extra information concatenated to the interaction vectors should enable the deep learning module to pickup on otherwise difficult to spot nuances within the data, hypothetically, this change could allow our model to find insights such as *'user X has a high preference for movies directed by Y, but only when the main actor is Z and the colour of the poster is Blue'*. Figure 9 illustrates a high-level representation of what the new user-item-info interaction should look like under our proposed outline.

5. Conclusion

In this paper we have conducted a systematic review of popular deep collaborative filtering methods and compared each proposed method's advantages and disadvantages. We found that collaborative filtering methods are still classic. From the earliest method ItemKNN, UserKNN to MF, SVD and so on, more and more methods have been transferred to deep neural networks, for example, NCF which is our seed paper and CDAE, etc. Recently, more and more studies have been on graph neural networks. We talked about the lightGCN in our survey as part of it. It can easily transfer the relation between item and user to graph structure and implement a more effective method of prediction. Collaborative filtering is keeping improving year by year and close

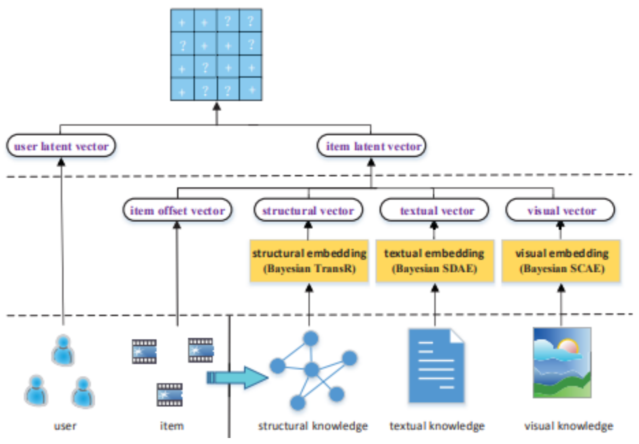


Figure 9. Visual representation of autoencoder compression/regeneration processes[13]

to the better and more accurate result. We also propose some potential methods for the inclusion of surrounding information to enrich the user-item interaction vector to be more descriptive. Note that our suggestion should be considered as a guideline to outline the type of thinking required to embed additional information to deep recommenders, and not a strict list of specifications to follow for implementation.

The key takeaways from our paper can be summarized as follows:

- NCF generalizes MF and solves the problems of MF's reliance on user-item linearity.
- NCF can embed other features to improve the accuracy of prediction.
- NCF allows for easy incorporation of surrounding information regarding user/item interactions via the incorporation of knowledge base embedding.
- Compare NCF with other CF based methods and session-based methods for recommendation systems.

We hope this paper has been an insightful read for beginners and experienced researchers of recommender systems alike, and special thanks to Dr. Kaiqi Zhao and Meng-Fen Chiang from the University of Auckland's department of Computer Science for their facilitation and guidance during the creation of this report.

References

[1] Steve Jobs

[2] 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, 2017.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Commun. ACM, vol. 35, no. 12, pp. 61–70, 1992.

[4] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," in Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08, 2008.

[5] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. Matrix factorization revisited," arXiv [cs.IR], 2020.

[6] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph Convolution Network for recommendation," arXiv [cs.IR], 2020.

[7] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural Graph Collaborative Filtering," arXiv [cs.IR], 2019.

[8] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational Autoencoders for Collaborative Filtering," arXiv [stat.ML], 2018.

[9] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with Graph Neural Networks," arXiv [cs.IR], 2018.

[10] J. Li, P. Ren, Z. Chen, Z. Ren, and J. Ma, "Neural Attentive Session-based Recommendation," arXiv [cs.IR], 2017.

[11] "STAMP: Short-term attention/memory priority model for session-based recommendation," Kdd.org. [Online]. Available: https://www.kdd.org/kdd2018/accepted-papers/view/stamp-short-term-attentionmemory-priority-model-for-session-based-recommend. [Accessed: 24-Oct-2021].

[12] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," arXiv [cs.LG], 2015.

[13] R. Dey and F. M. Salem, "Gate-variants of Gated Recurrent Unit (GRU) neural networks," in 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017, pp. 1597–1600.

[14] K. 2015 Organisers, "Collaborative knowledge base embedding for recommender systems," Kdd.org. [Online]. Available: https://www.kdd.org/kdd2016/subtopic/view/collaborative-knowledge-base-embedding-for-recommender-systems. [Accessed: 24-Oct-2021].