School of Informatics

Informatics Research Review
Deep learning: A panacea for shortcomings of traditional recommender systems?

[redacted]

January 2021

Abstract

Traditional recommender sytems have long been plagued with a multitude of issues which reduce their effectiveness. Methods from deep learning, the field of machine learning which concerns itself with neural networks, have recently achieved state-of-the-art results in many fields. In this review, I investigate how neural networks can be utilized to address prevalent issues in the two main paradigms of recommender systems, namely, content-based systems and collaborative filtering. I find that neural networks are effective at dealing with many challenges, but call for a unified evaluation benchmark, such that their efficacy can be more accurately and transparently measured.

[redacted], [redacted]

[redacted]

1 Introduction

We are living in the information age. The past century has been defined by an epochal shift in which information technology has replaced traditional industry as the main driver of the economy and innovation. As the name suggests, this has resulted in ever-increasing amounts of information being collected and stored digitally. In particular, the internet has ushered in a new paradigm of interaction. We retrieve information online through search engines, we order products via e-commerce, and watch movies on streaming websites. With this meteoric rise in information published on the web, consumers are inundated by the number of products and services available. The research of Iyengar and Lepper (2000) showed that when presented with an excess of choices, consumers tend to consume less than when presented with far fewer options. The aforementioned issue is a manifestation of information overload. The term has been coined to refer to excessive amounts of information that result in sub-optimal decision-making.

Recommender systems (RS) are a subset of information filtering which aim to ameliorate the aforementioned issue. They do this through utilizing data related to users and their preferences for predictions which are expressed in the form of personally relevant recommendations. In general, RS are therefore useful to consumers who are able to make better decisions. More specifically however, these systems are of particular interest to businesses and organizations which rely on accurate recommendations to operate. A famous example of the importance of RS is highlighted by the Netflix Prize. The eponymous competition awarded one million dollars to the team that could create a system which outperformed their current system by at least 10%. Bennett et al. (2007) stated that recommendations are of such importance to Netflix since without them subscribers would have trouble finding movies they enjoy and as a result they would lose interest and unsubscribe.

Traditional RS are generally classified as content-based (CB), collaborative filtering (CF), or hybrid. Although, there are some other other types such as knowledge-based recommenders, the aforementioned are the most prevalent (Kumar and Sharma, 2016). Consequently, in the remainder of the paper we mainly consider those. This is in line with literature such as the paper of Lü et al. (2012). CB systems model attributes of items and use the preferences of users towards those attributes to construct recommendations. CF captures similarities between users or items determined by past user behaviour to predict preferences for unrated items. It relies on either the notion that users who have agreed in the past will agree in the future or that a user would be interested in items similar to those the user previously purchased. Hybrid recommender systems combine the two other methods.

Traditional recommenders, however, have a number of drawbacks. Lü et al. (2012) posited sparsity, scalability, the cold-start problem, and diversity as major challenges. Sparsity occurs since the item space is immense and an individual user will not have rated the vast majority of the elements within that space. Despite the sparse user-item rating matrix, a large user base combined with an item selection of high magnitude can result in computational problems and therefore a lack of scalability. The cold-start problem refers to when an item has a lack of ratings or to a new user who rated very few items yet. This lack of defined preferences inhibits recommender systems. Finally, diversity or more specifically the lack therefore is related to sparsity. As mentioned before, many systems rely on previous consumption to determine recommendations. As a result, recommendations are often not very diverse nor do they recommend novel items frequently. This becomes an issue when users become exhausted from similar items. Sparsity and scalability are relevant to both CB and CF, while diversity and the cold-start problem are specific to CF. Although from this it may seem like CB is preferred over CF, CF is generally more successful (Van den Oord et al., 2013).

The specific classes of recommenders face more issues unique to them. First, matrix factorization (MF) is a popular family of methods that belongs to the CF class. Koren et al. (2009) provided a survey of a plethora of variants and explain how they are better than existing methods. In the canonical version, matrix factorization decomposes the user-item matrix into two latent feature matrices. These can be interpreted as a user-preference matrix and an item-feature matrix. In the user-preference matrix, the rows represent a user's preference to a certain feature, while in the item-feature matrix the rows correspond to which degree the features are present in an item. Traditional MF constrains the interactions between the latent vectors of the matrices to linear only. This restriction on the model of the interactions is a severe limitation. On the other hand, pertaining to CB systems, they require extensive feature engineering to function. This is because they depend on being able to define attributes or features for the items and users' preferences towards those features. Although hybrid systems attempt to remedy some of the aforementioned problems, issues still persists.

Deep learning is a subclass of machine learning based on the application of deep artificial neural networks (ANNs). Deep learning methods have achieved state-of-the-art results in a multitude of fields, including but not limited to, computer vision, natural language processing, and speech recognition (LeCun et al., 2015). Unsurprisingly, the research community has made efforts to incorporate deep neural networks into RS.

Due to the immense success it has enjoyed in recent years, deep learning has received intensive attention from academics and industry alike. As a result, the number of neural architectures is enormous. Additionally, neural networks are very flexible and multiple architectures can be combined in many ways. As a result, a veritably endless number of combinations exist. Therefore, to limit the scope of this review, we exclude papers that hybridize multiple neural architectures. Furthermore, I am mainly interested in the ability of neural networks in addressing shortcomings of traditional RS, more specifically CB and CF RS. Consequently, I constrain the selection of papers to those who tackle such challenges. For example, regarding CF this means I focus on papers that address the limitations of linear interactions in matrix factorization. While for CB recommenders, I collect papers that reduce the need for manually constructed features. I do not discuss issues in hybrid RS specifically and separately since they are designed to tackle challenges in the other two types of RS. In that sense, they can be considered an alternative to deep learning. They are therefore only considered insofar they contribute to improving CF and CB RS.

Formally, this paper does not provide a survey of RS which utilize neural networks in some way. Instead its primary objective is analyze deep learning based remedies for traditional RS. This means that deep RS models which do not attempt to solve any issues in traditional RS are excluded. I aim to achieve this by first determining how deep learning methods are used to enhance and supplement traditional RS. Furthermore, I question which shortcomings of traditional RS these deep learning methods can at least in part solve, and finally whether this results in an improved performance.

The remainder of the paper is structured as follows. In section 2, I provide a brief background on traditional RS and deep learning. In section 3, I present the main body of the literature review. Finally, I draw conclusions and suggest avenues for future research in section 4.

2 Background

Recommender systems are broadly divided into content-based or collaborative filtering. In this section I provide a brief overview of basic concepts on a high-level and pertinent terminology to facilitate understanding of the literature review.

2.1 Content-Based Systems

Content-based systems are a broad group of models which utilize the same principle. Namely, items are defined by their attributes, and users by their preferences towards those attributes. For a single user and item these can be represented as a preference and feature vector respectively. The two vectors have the same dimension where each element in the preference vector corresponds to an element in the feature vector. The user's rating of a certain item then equals the dot product between the two vectors which results in the following:

$$y_{ij} = \sum_{k=1}^{n} \theta_{jk} x_{ik} \tag{1}$$

Here y_{ij}, x_{ik}, and θ_{jk} represent user i's rating for item j, their preference towards attribute k, and the value of attribute k for item j respectively. Attributes for movies can be genres or color for clothes. The values of the attributes would represent how strongly that attribute is present in the item. So for example, how much romance there is in a book. The attributes and preferences vectors for all users can be stacked into two matrices, namely, a user preference and item feature matrix. From equation (1) it is easy to see the major drawback of CB systems. To obtain the rating for a specific item we must have features representing the items and the user's preferences towards those items. For the model to work well the features must be qualitatively sensible and numerically accurate. Choosing the right features for items therefore requires considerable amounts of feature engineering. To estimate the values for the features θ Lops et al. (2011) describe various methods. A commonly used one is TF-IDF. The preferences x can then be obtained through a regression as in equation (1) for the known ratings.

2.2 Collaborative Filtering

In CF, item attributes and user preferences are not modeled explicitly. This removes the need for manual feature engineering. Instead, generally only the user-item matrix is used. The rows in this matrix correspond to users while the columns belong to items, where the elements in the matrix represent the rating of item j given by user i. An example of such a matrix is illustrated in Table 1. The different types of CF depend on on how they utilize the user-item matrix. In

	Item 1	Item 2	...	Item n
User 1	?	3	...	4
User 2	7	?	...	8
...	...	...	...	...
User m	?	6	...	?

Table 1: User-item matrix. The elements are ratings given to items by users

the first, a user's predicted rating is computed using the known ratings of similar users. As a result, in this subset of CF it assumed that users who have agreed in the past will agree in the future. This approach is now termed as user-user CF and was introduced by Goldberg et al. (1992). Alternatively, CF can use similarities between items to make recommendations. This method was proposed by Sarwar et al. (2001) and is known as item-item CF.

More recently, matrix factorization has seen more interest. The method entails decomposing the user-item matrix into multiple matrices of lower dimension. Sarwar et al. (2000) achieved this through the singular value decomposition. An improvement using stochastic gradient descent to factorize the user-item matrix into two matrices was published in an online blog post by Simon Funk. This approach popularized MF and attracted the attention of the research community. A description of this approach was given by Koren et al. (2009). The two resulting matrices are often called latent and can be interpreted as the user preference and the item feature matrices described in the previous subsection. Meaning that the rows of first matrix correspond to the preferences towards the features of a user and that the rows of the second matrix can be read as the values for the attributes. The primary limitation of MF is that it decomposes the user-item matrix as the product of two matrices. Consequently, this means that each element in the user-item matrix is the dot product of two vectors from the decomposed matrices. For a single element this results in the following:

$$y_{ij} = \theta_j^T x_j = \sum_{k=1}^{n} \theta_{jk} x_{ik} \tag{2}$$

This results in the rating being a linear combination of the preferences and attributes. A linear model for the ratings however may not be sufficient. Furthermore, we can see that this is essentially equivalent to what we see in equation (1). What distinguishes MF from CB is that the latent vectors θ_j and x_i and the equation are obtained indirectly as a result of decomposing the user-item matrix. In CB, we must directly model and compute θ and x.

2.3 Why Deep Learning?

It is important to understand why we require deep learning in RS. Deep learning can be described as an area of machine learning that deals with the application of artificial neural networks. ANNs are computational models that consist of an input layer, multiple hidden layers, and an output layer. The depth of an ANN refers to the number of hidden layers it contains. Hence the more hidden layers, the deeper the network. Each layer consists of a number of neurons, which take the weighted and biased output of the previous neurons as an input and transform them non-linearly. The output of the neuron is then forwarded to all neurons in the next layer.

Through the hidden layers, neural networks can learn representations from raw data (LeCun et al., 2015). Bengio et al. (2013) explained the importance of representation learning for the success of machine learning and we will later see how this is especially useful for CB RS. Furthermore, neural networks are highly flexibly. There are a plethora of different architectures which excel at different tasks. For example, convolutional neural networks (CNNs) perform particularly well in image processing as shown by Krizhevsky et al. (2017) while recurrent neural networks (RNNs) are well suited for sequential tasks such as natural language processing (LeCun et al., 2015). Furthermore, Hornik et al. (1989) showed that neural networks are universal function approximators. This implies that they can compute any arbitrary function between the input and the output. This means that in RS they can compute complex non-linear functions between the latent user and item vectors which MF could not. Due the aforementioned, the application of deep learning in RS is of particular interest.

3 Deep Recommender Systems

In this section we go through the main body of literature. The subsections are divided into research that combines deep learning with either CB or CF.

3.1 Deep Content-Based Systems

As mentioned before, CB systems require extensive manual feature engineering and metadata related to the attributes of the items. Deep learning has mainly contributed to the improvement of CB through extracting features from raw data.

McFee et al. (2012) introduced an excellent example of how this issue can be circumvented in the field of music recommendations. They did this through learning the attribute feature vector by extracting information from music directly. They used a collaborative filtering algorithm to train their feature extraction method. Subsequently, they used the extracted features in a CB system to make predictions. Although this appears to be a hybrid RS since it combines both CF and CB, the authors still dubbed their method as CB. Likely, because the predictions were ultimately made through the CB. We will see in the remainder of this subsection that this is often the case. Furthermore, this method did not employ any deep learning methods. However, it is the basis for the seminal paper of Van den Oord et al. (2013).

In their paper, the authors replaced the feature extraction method of McFee et al. (2012) with a convolutional neural network. They used the Million Song Dataset (MSD). In addition to the MSD they had information on the playback counts for over a million users on a subset of MSD. They used the playback counts to construct the user-item matrix. Where the rating a user gave to an item depended on the number of times they played a song. They then applied weighted matrix factorization (WMF) to obtain latent factor representations for the users and the items. The authors stated that, similarly to McFee et al. (2012), they utilized the latent representations as the ground truth for training a model to map raw audio data to latent factors. This mapping is essentially a regression task for which they tested two distinct approaches. The first of which were used as baselines and entailed a linear regression, multi-layer perceptron (MLP), and metric learning to rank (MLR) proposed by McFee and Lanckriet (2010) with bag-of-words input features. The MLR method is essentially equivalent to what was done by McFee et al. (2012). The bag-of-words contained the processed versions of mel-frequency cepstral coefficients for the songs (MFCC). The second method was to use CNNs with intermediate time-step frequency as inputs. The authors reported a significantly better performance as measured by the area under the curve (AUC) for the CNN as compared to the baselines. By training the CNN on songs of which we know the latent vectors the authors were able to extrapolate this to new unseen songs. Effectively, the authors constructed a model that can now compute latent features directly from raw music. The latent vectors can now be used in a regression model to predict ratings for songs for users. This approach also remedies the cold-start problem since ratings for new unrated songs can be accurately predicted.

Since then, a number of variants have been proposed. Wang and Wang (2014) replaced WMF with probabilistic MF. Furthermore, instead of CNNs to extract features from audio, they employed a deep belief network (DBN). DBNs are another form of deep learning devised by Hinton et al. (2006). They compared their model against a number of benchmarks including the model of Van den Oord et al. (2013) and used root mean squared error (RMSE) as their metric. Their results reported a significant improvement over PMF and a regular CB system. Although they also show improvements over the model of that of Van den Oord et al. (2013). The difference is marginal. Schedl (2019) provided a review of deep learning for recommendations in music. Most of the methods they cover are CB. The paper of Van den Oord et al. (2013), however, remains among the most important within CB systems and in particular music recommendation.

The preceding papers were focused on music recommendation and mainly employed neural networks through extracting features from audio. In contrast, Musto et al. (2016) use neural networks to derive information from textual data. More specifically, they implemented Word2Vec, a method for creating embeddings from text proposed by Mikolov et al. (2013b). Embeddings are low-dimensionial dense vector representations. They applied their methodology to book and movie recommendations since rich textual data is available for these types of items in the form of item descriptions. They collected their item descriptions from Wikipedia and the ratings data from MovieLens and DBbook. They used the obtained embeddings as the representation vector in a CB model to make recommendations. They related their results to other non-deep methods for creating embeddings for CB and MF. They showed improved results based on F1-score.

Suglia et al. (2017) argued that the aforementioned approach is not considered deep, since Word2Vec uses what can be considered a shallow neural network. Nonetheless, inspired by their work, they used recurrent neural networks to derive information from textual data. Similarly, they also implemented their approach for movie and book recommendations. More specifically, for movies they collected synopes from Wikipedia and IMDB, while for books they retrieved plots from DBbook. Their model takes the description of the item and the user identifier as an input and outputs the probability that user would like that item. The item description is passed into a long short-term memory (LSTM) network and the user identifier to an embedding layer. The LSTM constructed by Hochreiter and Schmidhuber (1997) is here adopted to learn a latent representation of the textual item description. The embedding layer is used to create a dense vector representation for each user. The learned representations are then forwarded to a final logistic regression layer which outputs the probability. While perhaps not immediately obvious, this approach is still classified as CB. Here the user preferences and item attributes are learned through the LSTM and the embedding layer respectively. The performance is compared to a number of baselines including both CF and CB methods. They documented results superior to all baselines, which includes the Word2Vec-based method of Musto et al. (2016), for both book and movie recommendations. The metrics they used to evaluate the performance was F1-score.

In their successive work, the authors further attempted to improve the performance of their deep text model (Musto et al., 2018). They conducted a very similar experimental design, however, now they compared several neural architectures for deriving vector representations from movie descriptions. Namely, they compared RNNs, CNNs, auto-encoders, and bi-directional RNNs. They found that bi-directionals RNNs perform better than the LSTM in their previous work.

From the papers listed in this subsection we have seen two applications of neural networks in CB RS and one general approach. Namely, the approach is to extract latent feature vectors from data and the application was audio and text. Through utilizing deep CB RS we can effectively overcome the cold-start problem and diversity problem since CB systems do not suffer from these without having one of the drawbacks specific to RS.

3.2 Deep Collaborative Filtering

For CB, there appeared to be one main way with which deep learning could contribute. CF, in contrast, is more diverse, and as a result we will see more routes that can be taken to incorporate neural networks.

3.2.1 Neural Extensions to Matrix Factorization

We have previously discussed the limitations of collaborative filtering and in more detail matrix factorization, a popular method for CF. In short, an element in the user-item matrix is computed as the dot product between a latent user and item vector, which means the ratings are a linear combination of the user preferences and item attributes. The linear interaction is represented equivalently to that in equation (1).

The first and most influential work that directly addressed this limitation was published by He et al. (2017). They named their approach neural collaborative filtering (NCF). In essence, their method involves generalizing the dot product as shown in equation (3) with any arbitrary function which is equivalent to the following:

$$y_{ij} = f(\theta^T v_j^I, x^T v_i^U | \theta, x, W) \tag{3}$$

Here v_j^I and v_i^U are feature vectors for items and users respectively. For matrix factorization these would be one-hot encoding vectors where they identify a specific user-item pair. Furthermore, W is the parameter matrix for the function. If we impose the one-hot encoding restriction the equation reduces to:

$$y_{ij} = f(\theta_j, x_i | \theta, x, W) \tag{4}$$

which looks more similar to the familiar equation (3). We have already shown that neural networks are universal function approximators. Therefore, they are well suited to serve as the function between the latent vectors and the ratings. The general architecture of an NCF can be illustrated as follows. The input to the neural networks are the one-hot vectors v_j^I

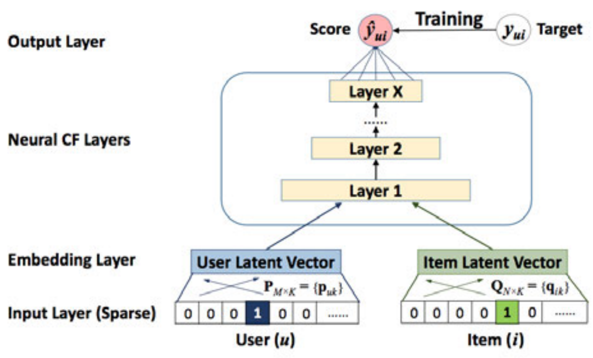

Figure 1: Illustration of Neural Collaborative Filtering. Picture taken from He et al. (2017)

and v_i^U that identify a single user and item pair. The embedding layer transforms the high-dimensional sparse vectors into a low-dimensional dense representation. These representations are then passed on to the first hidden layer. From here on, the model functions as a regular neural network. Applying their model to the Pinterest and MovieLens data sets, they found improvements in all tasks based on hit rate and NDCG. Their paper laid the foundation for more neural networks based CF methods.

For example, Song et al. (2018) explained how in the paper of He et al. (2017) non-interacted items were inherently assumed to be negative. This assumption was spurious according to them. They examined an alternative design in which non-interacted items are not treated as negative, but merely as less preferred. To do this, they altered the strategy with which the neural network learns by transforming the labels. They found that their approach resulted in better performance compared to NCF on MovieLens and Amazon Music, which are the two data sets they used. They used normalized discounted cumulative gain (NDCG) and hit rate to evaluate the models.

Two more subsequent papers attempted to mitigate the sparsity problem in CF. Both of them did so by using cross-domain recommendations since information from one domain can be used to supplement that of another. The first of which was by Lian et al. (2017). They supplemented the user-item matrix of the target domain with that of an auxiliary domain. They then factorized the matrix using a neural network as in He et al. (2017) where the items of both domains share the same latent attribute matrix. Although they showed improved results over a number of benchmarks, NCF was not included in the comparison. Wang et al. (2017) proposed neural social collaborative ranking (NSCR), which directly extends NCF to cross-domain recommendations. They combined user-item interactions from travel rating data with user-user interactions from social media. They were the first to bridge this gap and as a result had no existing papers to compare their results with. Nevertheless, they constructed benchmarks themselves using traditional state-of-the-art methods and found that NSCR performed significantly better based on AUC.

3.2.2 Item Embeddings

Diverging from neural extensions to MF another strand of deep CF focuses on improving the more traditional user-user and item-item CF as described in the background section. Items are recommended to users based on the similarity between either users or items. These methods are often based on similarity measures between the user and item vectors such as Pearson correlation. They generally suffer from cold-start problems and sparsity since new users and new items have few ratings and therefore similarity cannot be accurately measured. However, the item variant does scale very well. Three important contributions that use neural networks to model similarities between items were published around the same time. Namely, the works of Grbovic et al. (2015), Barkan and Koenigstein (2016) and Vasile et al. (2016).

In the former, the item embeddings are learned through skip-gram as described by Mikolov et al. (2013a) and the method was called Prod2Vevc. Skip-gram involves a two-layer neural network to create vector representations for words. The research was conducted by Yahoo through their e-mail service. They predicted the purchasing behaviour of 29 million users for 2.1 million items. The embeddings were created using the email receipts of user purchases, where they assumed that items which were bought together would be similar. Recommendations for items were made by recommending the most similar items based on the embeddings. The model was implemented into the Yahoo platform and according to the authors has improved performance in a number of key metrics. Vasile et al. (2016) improved on their work by incorporating metadata into training the skip-gram model. Accordingly, they named their model Meta Prod2Vec. They applied their design to music recommendation using the 30Music data set, consequently, the metadata came in the form of things such as artist IDs. Based on NDCG and hit rate, they reported that their model performed better than Prod2Vec.

The research of Barkan and Koenigstein (2016) is very similar to the previous two papers. The difference being they implemented skip-gram with negative sampling which is also known as Word2Vec. Although, Word2Vec is generally applied to text corpora, while Item2vec was applied to item baskets. Barkan and Koenigstein (2016) also made the assumption that items that were bought together would be similar. They tested their approach on the Xbox Music and Microsoft stores. They did not make direct recommendations, instead they investigated whether the resulting embeddings were good at measuring similarity. They did so by analyzing the genre metadata for items. They visualized their results using t-SNE, introduced by Maaten and Hinton (2008), on a 2D-space. The same was done for item vectors obtained from applying SVD to the user-item matrix. Visually, they observed that Item2Vec was able to cluster together items that belonged to the same genre better than SVD. They confirmed this using k-nearest neighbours.

To summarize, the main advantages of these approaches are that they are computationally efficient. Furthermore, they do not require explicit ratings, merely purchasing records. This alleviates the sparsity problem. It does not help particularly against the cold-start problem. Since in those situations, items have not been interacted with at all.

3.3 Collaborative Filtering with Auto-encoders

We have stated how traditional RS, including CF, suffers from the sparsity problem. We have already discussed some methodologies which alleviate this issue. Nevertheless, researchers have found that auto-encoders (AE) among others are especially adept at dealing with extremely sparse data for recommendations. AEs were popularized by Kramer (1991) and are unsupervised neural networks that reconstruct the input data.

The first method for using auto-encoders for collaborative filtering was proposed by Sedhain et al. (2015). They constructed two approaches, named U-AutoRec and I-AutoRec. The methods are based on the user-ratings and item-ratings vectors respectively. They found that I-AutoRec was more successful since the number of ratings items had received were usually much higher than the number of ratings users had given. Consequently, here I discuss I-AutoRec, though the methods are identical except for the input vector. Their idea was to use an AE to reconstruct the sparse item-ratings vector in which the many missing ratings were replaced with zeros. The AE would compress the input in the hidden layers and then reconstruct it in the output layer to predict the missing values. Their method was inspired by the research of Salakhutdinov et al. (2007), who used a restricted Boltzmann machine (RBM) instead. RBMs are also unsupervised neural networks which reconstruct input data. Their architectures differ somewhat to those of AEs however. Sedhain et al. (2015) explained how AutoRec has computational advantages over RBM CF while achieving a lower RMSE on the MovieLens data set. They found a reduction in RMSE compared to traditional MF variants as well.

The efforts of Sedhain et al. (2015) have spurred the development of more AE based CF methods. Generally, they employ different architectures for their AE. A fairly straightforward extension was proposed by Wu et al. (2016). They replaced the AE with a denoising AE, which adds Gaussian noise in the hidden layers, to train the network to be more robust. Li and She (2017) and Liang et al. (2018) both utilized a variational AE (VAE). The former specifically demonstrated the ability of their model to handle sparse inputs by using two data sets, Citeulike a and t, where 99.78% 99.93% of the entries in the user-item matrix were missing. Based on recall, they managed to outperform other deep and traditional RS substantially. Notably, the differences were larger when the data was more sparse. The methodology used by Liang et al. (2018) is similar. They distinguish themselves by assuming a multinomial distribution on the data rather than Gaussian and by altering the standard VAE objective, which they argued is over-regularized. They also measured their performance using recall in addition to DCG. The data sets were different however. Namely, Liang et al. (2018) used MovlieLens, MSD, and the Netflix prize collections. Although they did not use the work of Li and She (2017) as a baseline, they did compare against NCF and the denoising AE of Wu et al. (2016). They managed to outscore the denoising AE across the board, but NCF only with respect to some measures.

We see that although methods like NCF are still superior in some regards, auto-encoders have been shown to deal well with sparse inputs and are hence especially useful in addressing sparsity issues in RS.

4 Summary, Conclusion, and Future Research

In this paper, we have discussed how deep learning can contribute to recommender systems. More specifically, I first listed a number of challenges traditional RS face. I then presented a body of literature to illustrate how deep learning can be used to supplement traditional RS and address their shortcomings.

For CB systems, we have seen that existing research mainly attempts to catalyze the process of feature engineering, through directly extracting features from raw data. We have seen two main fields of application in which this has been done successfully. Namely, authors have used various types of neural architectures to extract latent vector representations from audio and textual data. They have all found improvements over traditional baselines, while circumventing the feature problem.

For CF, I showcased a more diverse set of approaches in which neural networks can be incorporated. First we have examined neural extensions to MF. This strand of literature has allowed for the ratings to be any function of the latent user and item vectors, which has greatly increased the flexibility of MF. Furthermore, we have seen a number of ways neural networks can be used to create low-dimension embeddings to represent items. These embeddings are computationally efficient, do not require user ratings to be created, and are accurate at representing item similarities. They however do cause a lack of diversity in recommendations since they are utilized to recommend items that users have previously interacted with. Finally, within CF, we have presented AE based filters which are particularly adept at handling sparse data.

We can conclude that, despite the apparent power of neural networks, that there is no free lunch yet. Currently, neural networks generally focus on solving a single issue in a given approach, however do little to nothing to combat other problems. For example, although NCF allows for non-linear interactions, it does nothing to improve diversity or scalability. Furthermore, where pertinent, all papers in this review, reported that their model showed improved performance over a number of baselines. Although this seems extremely promising, the chosen baselines, however, often seem somewhat arbitrary. Additionally, few experiments were carried out using the same or even similar data and evaluation metrics. Consequently, this could possible suggest that many of these models are not as powerful as their authors suggest and only achieve state-of-the-art results under specific circumstances.

For future research, I firstly recommend a unified framework for evaluating RS. I suggest the RS community take inspiration from computer vision and natural language processing. In these fields benchmark data sets such as MNIST or tasks such as GLUE exist such that novel methods can be fairly and accurately measured. Finally, I propose more research into the scalability of deep recommender systems. Deep neural networks are known to be extremely computationally expensive so much that research has been dedicated to gauging their impact on the environment (Strubell et al., 2019). This adds to the fact that traditional RS already suffer from scalability issues. Although we have seen computationally efficient methods such as the item embeddings, these lack the advantages of the other models, and consequently it would be beneficial to extend the efficiency to other deep models.

References

Barkan, O. and Koenigstein, N. (2016). Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., and Sharp, D. (2015). E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hornik, K., Stinchcombe, M., White, H., et al. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Iyengar, S. S. and Lepper, M. R. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, 79(6):995.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kumar, B. and Sharma, N. (2016). Approaches, issues and challenges in recommender systems: a systematic review. *Indian journal of science and technology*, 9(47):1–12.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Li, X. and She, J. (2017). Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 305–314.

Lian, J., Zhang, F., Xie, X., and Sun, G. (2017). Cccfnet: a content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the 26th international conference on World Wide Web companion*, pages 817–818.

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698.

Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer.

Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., and Zhou, T. (2012). Recommender systems. *Physics reports*, 519(1):1–49.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

McFee, B., Barrington, L., and Lanckriet, G. (2012). Learning content similarity for music recommendation. *IEEE transactions on audio, speech, and language processing*, 20(8):2207–2218.

McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *ICML*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Musto, C., Franza, T., Semeraro, G., de Gemmis, M., and Lops, P. (2018). Deep content-based recommender systems exploiting recurrent neural networks and linked open data. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 239–244.

Musto, C., Semeraro, G., de Gemmis, M., and Lops, P. (2016). Learning word embeddings from wikipedia for content-based recommender systems. In *European Conference on Information Retrieval*, pages 729–734. Springer.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Schedl, M. (2019). Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics*, 5:44.

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112.

Song, B., Yang, X., Cao, Y., and Xu, C. (2018). Neural collaborative ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1353–1362.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Suglia, A., Greco, C., Musto, C., De Gemmis, M., Lops, P., and Semeraro, G. (2017). A deep architecture for content-based recommendations exploiting recurrent neural networks. In *Proceedings of the 25th conference on user modeling, adaptation and personalization*, pages 202–211.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651.

Vasile, F., Smirnova, E., and Conneau, A. (2016). Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 225–232.

Wang, X., He, X., Nie, L., and Chua, T.-S. (2017). Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194.

Wang, X. and Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636.

Wu, Y., DuBois, C., Zheng, A. X., and Ester, M. (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162.