Using User Reviews to Enrich Social Recommender Systems

Ahmad Jawaad Shah
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Durban 4000 RSA
218029400@stu.ukzn.ac.za

ABSTRACT

Recommender systems have become increasingly popular in the online domain and play a critical role in suggesting information of interest to users. Various techniques have been explored while implementing these systems, such as collaborative filtering, content-based filtering, and preference-based solutions. In recent times, personal data has become ubiquitous and has prompted researchers to explore avenues that use user reviews on items of interest and social relationships to improve output recommendations. Furthermore, to the foregoing, the immense interest in Neural Networks has provided a platform for applying deep learning techniques to improve existing recommender system solutions. In this paper, we present a novel deep neural network framework (**RevNet**) for social recommendations through the utilization of user reviews. The model captures the user-user and user-item spaces typically found in social recommender systems, as well as adds another dimension in the form of a user-review space. In this regard, in the user-review space, we aim to capture the reviews provided by users on items. Thus, we propose to employ the use of doc2vec representations of user reviews on items which will be incorporated into a neural network for ratings prediction. Evaluation of the proposed solution on a real-world dataset, shows that the inclusion of user reviews in a social recommender system is effective.

CCS CONCEPTS

•Information Systems → Social recommendation
•Computing Methodologies → Neural networks

Keywords

Social Recommender System; Neural Networks; User Reviews

1. INTRODUCTION

Recommender Systems have been around for quite some time. These systems provide personalised service support to users by learning their previous behaviours and predicting their current preferences [1]. Moreover, there has been a rapid increase in web technologies that employ recommendation systems to enhance the users' experience. The driving factor behind this is that it solves a major problem surrounding today's times, information overload. Recommender systems can be used to overcome this phenomenon by automating the decision-making process and providing a set of informed recommendations, thereby enhancing a user's experience [1].

Various approaches have been adopted over the years and as such, has lead to the creation of complex architectures wherein recommender models utilise more information than previously explored. Work in the collaborative filtering [9] domain has attracted much attention, however with recent advancements in neural networks and deep learning, these have been extended to deep architectures, as can be seen in [2]. Furthermore, the use of user-item interactions through a medium such as ratings was widely adopted for recommender models. However, with the advent of social media and the ubiquitous presence of user data, newer architectures have been proposed such as, [2], [3] and [4], which incorporate social information via user-user interactions, also referred to as social relations.

The incorporation of social relations into recommender systems has been backed by social theories, where it is argued that people are influenced by their social connections, which ultimately leads to them having similar preferences [2] [5] [6]. Thus, in social recommender models, it is imperative to have an architecture that is able to effectively represent the users as well as capture and use the social relation effectively. As depicted in Figure I, a user is involved in both the item and social spaces of a social recommender system. The user-item space denotes the interactions between a user and items, whereas the user-user

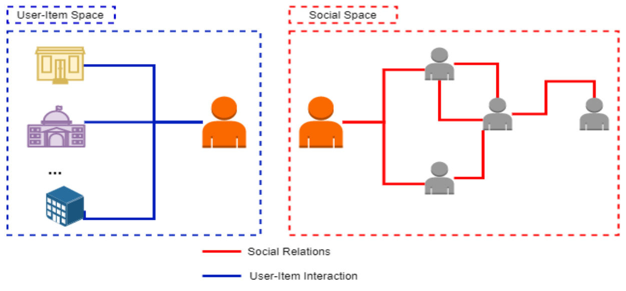


Figure I: An illustration of a user being involved in the user-item and user-user spaces

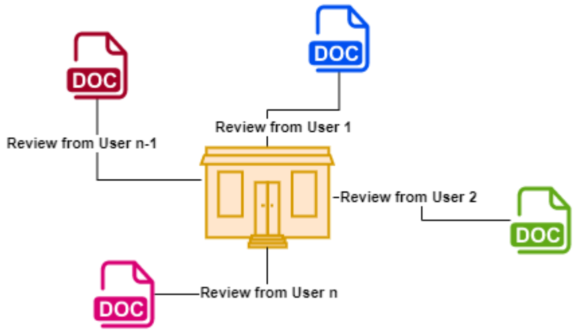


Figure II: Multiple reviews for an item

space captures the friendships amongst users. Furthermore, in social recommendations, it is important to incorporate the social network information into user and item latent factor learning [3].

This study aimed to add another dimension to the user-item and user-user spaces in social recommender systems. In particular, we propose a third space, i.e., a user-review space through which we aim to capture the sentiment expressed by the user towards a certain item via a textual review. Thus, the goal is to find an appropriate embedding technique and measure its effectiveness when incorporated into a social recommender system. Moreover, as stated in [3], learning representations of users and items are key in a social recommender system. Thus, following from the aforementioned, using a deep neural network for the proposed problem will provide a platform for effective latent factor learning as well as provide an opportunity through which we can advance social recommender systems. Figure II depicts the relationships between reviews and items, i.e., an item may have many reviews from different users. The three proposed spaces will be used to learn latent factors, which will thereafter be combined and used for rating prediction.

In this paper, we propose a deep neural architecture for social recommendations, **RevNet**, The contribution from this paper can be summarised as follows:

- We propose a deep neural architecture which utilises user reviews for social recommendations. A principled approach for manipulating user reviews into a format that can be utilised as neural network embeddings; and
- An in-depth evaluation of the effectiveness of the proposed architecture on a real-world dataset.

The remainder of this paper is organised as follows. We discuss and review previous work done in the recommender domain in Section 2. In Section 3, we introduce the proposed framework and elaborate on the design choices made. In Section 4, we conduct experiments on a real-world dataset to illustrate the effectiveness of the proposed architecture. Finally, we conclude the paper by discussing future directions for work to be done in the recommender domain, acknowledgement of notable contributors, and a compilation of sources used in this paper.

2. LITERATURE REVIEW

Recommender systems are considered one of the most useful tools in the digital world. The aim behind recommender systems is to manipulate available data such as user profiles, social relations with other users and item data to infer customer interests [7]. Various techniques have been proposed throughout time, ranging from content-based filtering [8], collaborative filtering [9] and combining both approaches known as a hybrid filtering recommender system.

Collaborative filtering techniques can be summarised into three categories, i.e., user-based, item-based and model-based methods [7]. A user-based collaborative approach predicts a relevant item by considering the similarity between users. This is achieved by comparing ratings on a common item. Furthermore, the average of the ratings is calculated by considering the user's social relations, and a quantitative value is obtained and used as a guide on how desirable a product is. Item-based collaborative filtering uses the similarity between products.

An alternative to collaborative filtering is content-based filtering. Content-based recommender systems provide recommendations based on the previous likes and dislikes of a user [7]. User profiles are generated by analysing the details of products that the user has provided ratings on. Furthermore, the recommender system will use the generated profiles to make informed predictions. As can be seen, both the previous approaches have benefits; however, any disadvantage found in an individual technique can be overcome by using multiple techniques. This prompted many individuals to apply a combination of techniques to enhance recommender systems. This is known as hybrid filtering. The practical advantages of using a hybrid technique are that it can overcome sparsity and cold start problems [7].

[10] explores the various techniques used in designing textual recommender systems. At the heart of these systems are good representations of textual data. A good representation is obtained by conducting the necessary feature extraction. As noted in [10], feature extraction can take many different forms. This may vary from simple keyword-based techniques to much more complex methods such as deep learning approaches.

Keyword-based approaches for feature extraction is the simplest form available. Text pre-processing is conducted, following which a relevant keyword-based technique is applied. Furthermore, the keywords are transformed from text into numerical data as a result of this transformation. One such approach of keyword-based feature extraction is TF-IDF scores [10][11]. In this approach, the text data is represented in an m-dimensional vector, where m is the number of terms extracted from the data. Term frequency is used to analyse the occurrence of each term. Furthermore, a weighting mechanism is used to give an overall representation of each term in the review.

Topic modelling is another approach for feature extraction. This approach can be used to identify the main topic for a few lines of text. One possible way of conducting topic modelling is through Latent Dirichlet Allocation. LDA is used to cluster words that co-occur in documents to form topics. A K-dimensional topic distribution can represent each document. Each topic is assigned a word distribution to indicate the probability of a particular word being related to it [11]. Using this approach, we obtain a vector of features which represent the identified topics [10].

Embedding techniques such as word embeddings deals with representing textual data in a vector of real numbers. Word2Vec is one such model. Doc2vec is an extension of word2vec. The advantage of using doc2vec is that it provides an algorithm for paragraph vector embedding. Moreover, in the context of recommender systems, it can be used to facilitate the conversion of textual reviews into numerical representations while also preserving the meaning carried by long paragraphs of text [12]. Using the doc2vec approach as an embedding technique for a deep social recommendation is an area which has not been explored.

The use of deep learning solutions to machine learning problems, such as the one being discussed, can be used to provide better results than contemporary methods. Automation of intelligent behaviour is the goal behind AI and its associated deep learning techniques. These techniques mainly cover six areas: knowledge engineering, reasoning, planning, communication, perception, and motion [13]. All of the areas above can be applied to enhance recommender systems except for the motion aspect [1], which has no specific role in a recommender system. These benefits further motivate the use of Deep Neural Networks.

Various deep learning solutions to recommender systems have been explored. In particular, solutions such as Multi-Layer Perceptron were implemented to replace similarity modelling between users and items [14]. Furthermore, Convolutional Neural Network architectures for interaction modelling were proposed; however, a trade-off is made between better performance and an increase in model complexity and time cost. Regarding content-enriched recommendation models, autoencoder and attention models have been proposed for modelling textual content. Autoencoders provide a general neural solution for unsupervised feature learning. The intuition behind autoencoders follows from treating each item's content as raw features, such as a bag of words. This approach does not consider the uniqueness of text but instead leverages autoencoders and their variants to learn the hidden content representations of items.

Social recommender systems are comprised of two main components, i.e., user and item modelling [3]. The goal of user modelling is to learn latent factors about a user. This is captured by considering the opinions of a user on an item as well as social relations. Hence, item and social aggregation are essential. Item aggregation seeks to capture the relationship between a user and an item while providing a quantitative measure of their satisfaction with the item. This is done using the rating value supplied by a user. Therefore, to learn item-space user latent factors, we will consider users and the ratings they provide for items. To better understand users, it is crucial to consider their social relations when modelling user latent factors. Furthermore, a user may have varying strength of ties with each of their "friends".

Considering each of the proposed architectures described above, and despite the compelling results achieved in many of these models, a gap has been identified wherein very little attention has been paid to user reviews in social recommenders.

Furthermore, there is an abundance of text modelling techniques, whose effectiveness in a social recommender domain has not been explored. Thus, we propose a novel social recommender that utilises user reviews, **RevNet**, to fill this gap.

3. METHODS AND TECHNIQUES

Table I: Notation

Symbol	Definition and Description	Symbol	Definition and Description
u_i	An individual user i in the model	d	The length of the embedding vector
vj	An individual item j in the model	F(i)	The mapping of friends for a user u_i
r_{ij}	The rating r_{ij} of a user u_i on and item v_j	h^S_i	The user latent factor of user u_i, from the space containing the social friends F(i) of user u_i
r'_{ij}	The predicted rating r of a user u_i on and item v_j	I(i)	The set of items that the user u_i has interacted with
t_{ij}	A textual review from user u_i on item v_j	h^I_i	The item space user latent factor of the items for I(i) for the user u_i
s_{ij}	Represents the sentiment from user u_i on item v_j	R(i)	The set of textual reviews that the user u_i has made for the items in I(i)
rec_{ij}	Represents if a user u_i would recommend the item v_j	h^R_i	The review space item latent factor
eu_i	Embedding of u_i	h_i	The combined latent factor after concatenation
ev_j	Embedding of v_j	$\oplus$	Concatenation operator applied on vectors
et_{ij}	Embedding of a text review made by u_i on v_j	η	The learning rate of the neural model
eo_i	The embedding of an opinion from a user.	α	The alpha parameter used in the LeakyReLU activation function
ef_i	The embedding of the friends for a u_i	W, b	The weights and bias of the neural network, respectively

3.1 Definitions and Notations

Let U = $\{u_1, u_2, u_3, \ldots, u_n\}$ and V = $\{v_1, v_2, v_3, \ldots, v_m\}$ be the set of users and items respectively, where n represents the total number of users and m the total number of items. For each user, we obtain a mapping of social relations, i.e., those users with whom u_i is friends. We choose to represent this as F(i), the friends of u_i, as F(i) = {ui: $\{u_1, u_2, \ldots, u_k\}$}, where k is the number of u_i's friends. Let I(i) represent the items that a user has interacted with. Furthermore, for each user-item interaction, we obtain a rating r_{ij}. For each r_{ij}, we extract a further two parameters, namely s_{ij} and rec_{ij}, which represent the sentiment and whether the item is recommendable according to u_i. In addition, for each user-item interaction, there is an accompanied textual review which we denote as t_{ij}.

Following the approach used in [3] and [9], we provide embedding vectors for users and items through a fixed dimension. Thus, we represent the embedding of users and items as $eu_i \in \mathbb{R}^{100}$ and $ev_i \in \mathbb{R}^{100}$, respectively. Lastly, the embedding of a user's friends and text reviews are represented variably through a d- dimensional embedding. Thus, we represent the embedding of a review as $et_{ij} \in \mathbb{R}^{d}$ and of friends as $ef_i \in \mathbb{R}^{d}$. All of the aforementioned information is summarised in Table I.

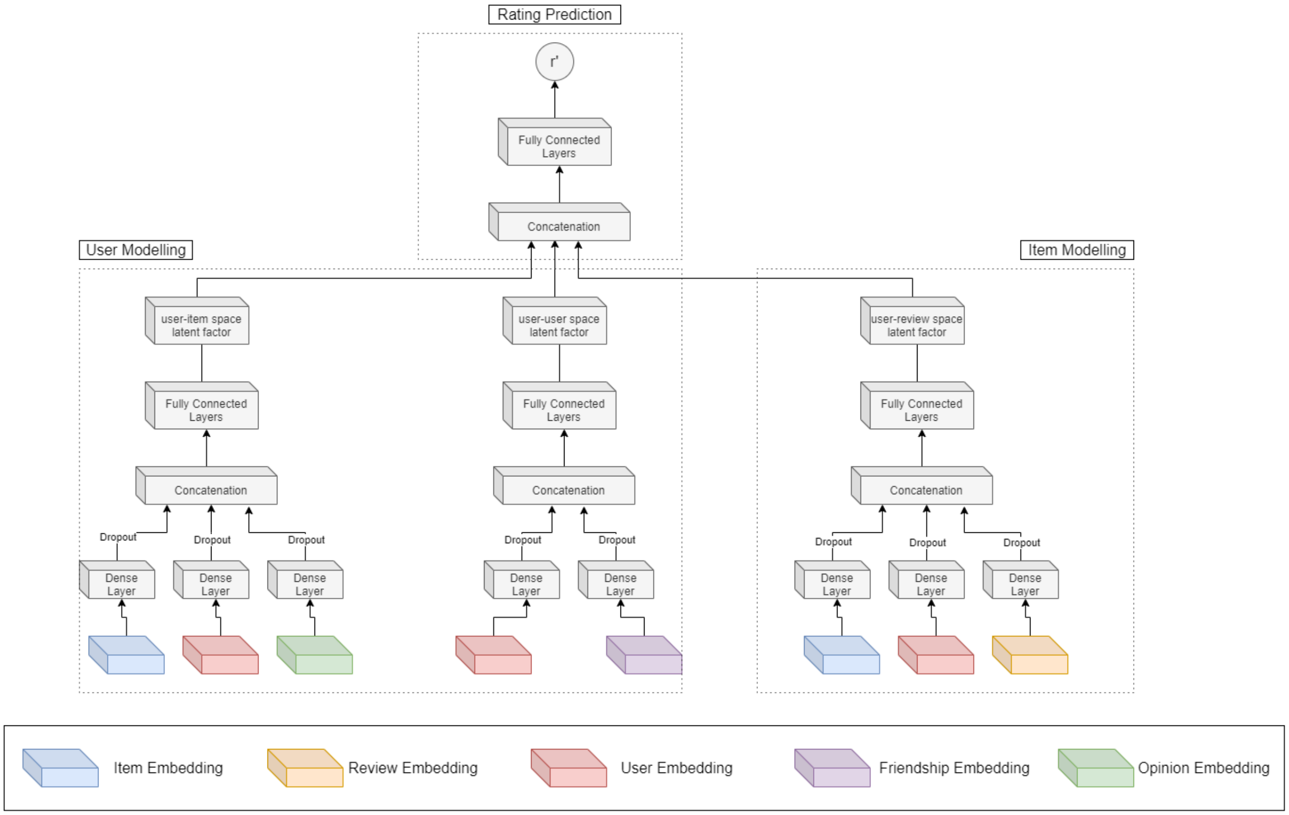


Figure III: The architecture of the proposed model. The model contains user-item and user-user spaces for user modelling and a user-review space for item modelling, with the ultimate goal being rating prediction.

3.2 An Overview of the Proposed Framework

In this subsection, we outline some of the high-level concepts used in the design of the proposed architecture. As depicted in Figure III, the model contains three main components i.e., user modelling, item modelling and rating prediction. Furthermore, we derive latent representations from the user modelling and item modelling components, via their respective spaces.

User modelling seeks to understand users better, thus we facilitate this via the user-item and user-user spaces. Regarding user-item interactions, we propose appropriate embeddings to capture a user's opinion on an item. Furthermore, as in a typical social recommender system, the social relationships are considered. This is done to model users from the social perspective. The outputs obtained from each of these spaces represent item user latent factors (h^I_i) and social user latent factors (h^S_i), respectively. We choose not to combine the obtained user latent factors into a single user latent factor representation.

The second component of the proposed architecture is item modelling. This is done through the consideration of text reviews and user opinion on items. Thus, we receive a latent factor representation (h^R_i) for the items.

Lastly, to facilitate rating prediction, we combine user and item modelling latent factor representations through concatenation, which is thereafter fed into subsequent neural network layers. The remaining subsections will discuss each of the model's components in detail.

3.3 User Modelling

The left portion of Figure III depicts the user modelling portion of the neural network. The main goal behind user modelling is to learn user latent factors defined as h^I_i and h^S_i. The item user latent factor (h^I_i) is learnt from the user-item space through user opinion on items. In particular, embedded representations of users, items and opinions were utilised. User opinions are scalar values that measure the user's sentiment towards an item. These were each passed through a single dense layer before being concatenated for item space user latent factor learning. In general, for each neuron in the first dense layer, we can represent the process mathematically as

$$y = W.e_i + b \quad (1)$$

$$z = \max(\alpha * y, y) \quad (2)$$

Where e_i represents any one of .... (eo_i), user embedding (eu_i), and item embedding (ev_j.

Learning user latent factors from the social space was facilitated in a similar manner. The two main components of the social space contained a user embedding as well as a friendship representation. Due to sparsity that arises with a nxn user matrix, friendship representations underwent pre-processing. In particular, the friends of user u_i, F(i), were converted into vectors through the doc2vec model.

3.4 Item Modelling

In the item modelling component of the neural network, we seek to learn item latent factors defined as h^R_i. In particular, this is done through user, item and review embeddings. The incorporation of reviews seeks to utilise text in a social recommender system. Furthermore, the reviews contain user reviews on an item v_j. Effective representation of the reviews was learnt beforehand through the doc2vec model. Furthermore, the reviews had to be pre-processed in order to capture the most important parts of the reviews.

Similar to the approach used to learn user latent features, a dense layer is utilised before concatenation. Thereafter, the concatenated representation is fed into a set of fully connected layers.

3.4.1 Review Feature Selection

This paper's main contribution towards social recommender systems is incorporation of user reviews to improve the efficacy of social recommender systems. Natural Language Processing (NLP) techniques were used to extract meaningful features from the reviews. The text was pre-processed as follows: (i) First, appropriate NLP techniques are utilised to transform the raw reviews into a processed and uniform state, (ii) the processed reviews are transformed into d-dimensional vectors.

3.4.2 Review Pre-Processing

User reviews in their raw state contain many words that are not helpful to our NLP task. These include the likes of stop words, hashtags and webpage links. Thus, we follow a pre-processing pipeline through which we seek to uniformly represent the data and improve its appropriateness for the NLP task at hand.

The first step in our pre-processing pipeline was to convert the entire review into lower case. Thus, case uniformity is established, and as a result, case sensitivity will not be an issue when conducting future text operations. The next operation carried out is stop word removal. Stop words are words, such as "and", "a" and "is", that occur frequently throughout a text, which carry very little meaningful information. Thus, the removal of stop words reduce the overall word count and allow us to focus on the words that carry meaningful semantic information. Following from the previous, the removal of hashtags and web links was done. Thereafter, the removal of punctuation marks was facilitated. All of the removal operations contribute significantly to the overall representation of the text, as it removes unneeded data and reduces the overall noise.

The last operation in our pre-processing pipeline was normalization through Lemmatization. Lemmatization converts a word to its base form; however, it considers the context in which the word appears and provides a meaningful base form [20]. On the other hand, stemming converts a word into its root form, thus not capturing the contextual information carried by the text. For this reason, lemmatization was chosen over stemming. [20] provides an intuitive example where the word "**caring**" is lemmatised to "**care**" and stemmed to "**car**".

3.4.3 Review Embedding

Review embeddings formed the basis of the text representation portion of the proposed social recommender solution. As such, the technique used was doc2vec vectors, introduced as paragraph vectors in [18]. The motivation behind doc2vec lies in its construction, as it able to construct representations for sequences of any length, does not require tuning of word weights and does not rely on parse trees [18]. Furthermore, paragraph vectors are able to capture the semantics of the words as well as is able to consider word order.

In particular, in this paper, we are interested in the effectiveness of the two provided doc2vec algorithms and their impact in a social recommender solution. Typically, doc2vec representations can be utilised for next word predictions; however, we adopt it for review representations as it is effective in capturing the context of text. The first algorithm used for paragraph vectors is the distributed memory model algorithm. The process carried out by the algorithm essentially entails unsupervised training on the text to get word vectors, softmax weights U, b and paragraph vectors [18]. The second algorithm is the disturbed bag of words algorithm which does not use word ordering. This algorithm is conceptually simpler than the previous algorithm as well as consumes less storage space as word vectors are not stored.

We employ both of the provided algorithms interchangeably to measure the effectiveness of user reviews.

3.5 Rating Prediction

In this subsection, we discuss the final layers of the RevNet neural model for the selected recommendation task. In particular, we focus on ratings predictions as opposed to the other recommendation task of item ranking.

Regarding ratings predictions, each of the spaces latent factors are used to contribute towards the final prediction. In particular, the method chosen to combine the latent factors of the user-item (h^I_i), user-user (h^S_i) and user-review (h^R_i) spaces is through vector concatenation. Thus, we obtain h_i. Mathematically, we can define the process as follows,

$$h_i = [\, h^I_i \oplus h^S_i \oplus h^R_i] \tag{3}$$

Following the concatenation, h_i, is fed into a further two hidden layers, the first containing four hidden neurons and the latter containing one neuron to facilitate a single value output. The output value represents the predicted rating r'_{ij}. Both hidden layers utilise a LeakyReLU activation function with α being set to 0.1. The steps below complete the rating prediction process carried out in the neural model

$$y = w_1 \cdot h_i + b_1 \tag{4}$$

$$y = \max(\alpha * y, y) \tag{5}$$

$$z = w_2 \cdot y + b_2 \tag{6}$$

$$z = \max(\alpha * z, z) \tag{7}$$

$$r'_{ij} = z \tag{8}$$

Lastly, due to rating prediction being a regression problem where the neural model provides only an approximate rating value, through which receiving whole numbers as output is rare, we proposed to use a rounding mechanism on r'_{ij}. This is only done because the r_{ij} provided by users on items are whole numbers. In this regard, r'_{ij} is updated as defined below.

$$\begin{cases} r'ij = ceil(r'ij) & if\ r'ij - 0.5 \geq floor(r'ij) \\ r'ij = floor(r'ij) & otherwise \end{cases} \tag{9}$$

3.6 Model Training

In order to estimate the parameters of RevNet effectively, an appropriate loss function needed to be selected and optimised. Taking into consideration that the model's main goal is ratings prediction, the objective function selected is the mean squared error (MSE) loss function. The MSE formula is defined as follows,

$$\text{Loss} = \frac{1}{n} \sum_{i=1}^{n} (r_{ij} - r'_{ij})^2 \tag{10}$$

Where n is the number of data points, r_{ij} is the ground truth rating and r¢$_{ij}$ is the predicted rating from the model.

To optimize the objective function, the Adam [15] optimizer was adopted. The Adam optimiser is a hybrid algorithm between the AdaGrad [16] and RMSProp [17] optimizers, which utilises the advantages of both of the aforementioned optimizers. Furthermore, the motivation for the use of the Adam optimizer stems from the fact that it requires minimal memory usage [15] as well as provides a smooth path to the loss functions global minimum. The η of the model was set to a relatively small value, 0.001, however; this value was varied during experimentation and the results can be found in Section 4.

For the representations of users and items, one hot encoded vectors were not used as they have a high dimensionality which leads to sparsity issues when training. Rather, these features were embedded into low-dimensional spaces for ease of execution.

Since a dedicated ratings matrix is not being used, due to the previously mentioned issues, the ratings provided by each user was used to create three additional parameters, namely, an average rating, a user sentiment, and a user recommendable value, of which the latter indicates if the user would recommend the item to others. These are used in appropriate embeddings, in specific, the opinion embedding eo_i. Furthermore, a representation of a user's textual review and social network was learnt beforehand via a doc2vec [18] model. The doc2vec models were trained on either one of the distributed memory or distributed bag of words techniques. Further evaluation and consequence of each technique will be explored in Section 4.

To tackle the issue of overfitting that may arise in a neural network, a dropout approach was used. A dropout strategy drops some neurons during the training process. In this regard a standard dropout value of 0.2 was defined; however, this can be varied as desired. Lastly, an early stoppage call-back was added to the model. The inclusion of early stoppage halts the training process after 20 epochs if no significant changes has been made in minimising the validation loss.

4. RESULTS AND DISCUSSIONS

In this section, we evaluate the proposed model with respects to MAE, RMSE and Accuracy. We evaluate the model to measure its performance on a real-world dataset. First, we begin experimentation by investigating the impact of hyperparameters on the proposed model's performance. To this end, the model is evaluated under different hyperparameter configurations.

The second part of experimentation explores a comparative analysis between RevNet and two variants of itself. We compare the model against Mod-RevNet, a model containing no social and review information, and Soc-RevNet, a model containing social information, but no review information.

4.1 Experimental Settings

Initial design and implementation of the proposed solution was done on an Intel i3 core, with 4 GB of RAM and a CPU clocking at 2.2 GHz. Thus, the default parameters, provided in the code, in terms of users, reviews and doc2vec epochs are relatively small. For model evaluation, the code was optimised through parameter passing and executed on the Hippo[1] cluster made available by the astrophysics department of the University of KwaZulu-Natal.

Lastly, the solution was implemented using Python 3.9, using Keras[2] 2.5.0 with a Tensorflow backend. Doc2Vec training was conducted using Gensim[3] 4.0.1.

4.1.1 Dataset

To conduct the necessary experiments, we adopted the Yelp[4] dataset. The Yelp dataset contains a substantially large number of users that provide ratings and reviews on businesses. Each rating is one of the values {1, 2, 3, 4, 5}. Furthermore, the dataset contains social information in the form of friendships. Table II describes the simple statistics of the dataset. During experimentation, a sample portion of the dataset is used as it becomes computationally expensive to use the entire dataset.

Table II: Dataset Statistics

Yelp Dataset	
# Of Users	2 189 457
# Of Businesses	160 585
# Of Reviews	8 635 403
Subset Of Dataset Used	
# Of Users	1 000
# Of Businesses	160 585
# Of Reviews	10 000

4.1.2 Evaluation Metrics

In this paper, we adopt offline analytics [21] to measure the performance of the model. This entails splitting the data into training and testing portions and measuring the model's performance on the testing data.

Since the model is involved in rating prediction, we choose to measure the models Accuracy, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These can be defined as follows,

$$\text{Accuracy}=\frac{C}{|Q|} \tag{11}$$

$$\text{MAE}=\frac{1}{|Q|}\sum_{(i,\,j)\in Q} |\, r_{ij} - r'_{ij} \,| \tag{12}$$

$$\text{RMSE}=\sqrt{\frac{1}{|Q|}\sum_{(i,\,j)\in Q} (r_{ij} - r'_{ij})^2} \tag{13}$$

Where |Q| represents the number of samples in the test set, C the number of correctly predicted values, r_{ij} the rating provided by u_i on v_j and r'_{ij} the predicted rating for u_i on v_j from the model. The adoption of accuracy was solely for the purpose of identifying if our predictions r'_{ij} were close to the provided rating r_{ij}. Thus, we don't consider precision and recall, which are typically coupled with accuracy, for evaluation in this paper.

4.1.3 Benchmarking

To evaluate the performance of the proposed model, we compared its performance to two variants of itself. In the first variation of the model, Mod-RevNet, the user-user and user-

[1] https://astro.ukzn.ac.za/~hippo/
[2] https://keras.io/
[3] https://radimrehurek.com/gensim/
[4] https://www.yelp.com/dataset

review spaces were removed, leaving the model with only the user-item space to work with. Thus, the modified solution can be seen as a deep neural network solution that predicts item rating, without user reviews and social relations. The second variant, Soc-RevNet, keeps the social space of the original model whilst discarding the user-review space. Soc-RevNet can be defined as a deep social recommender model for ratings prediction.

4.1.4 Parameter Settings

The model was developed using Keras with a Tensorflow backend. For training, 80% of the read in data was used to learn parameters, 5% was used for validation and hyper-parameter tuning and 20% for testing the final performance.

Doc2Vec embeddings were set to an output dimension of d, where $d \in \{32, 64, 128, 256\}$. Furthermore, the number of epochs for doc2vec training was set to 20 throughout. We also choose to vary the doc2vec learning algorithm between Distributed Memory and Distributed Bag of Words.

Regarding the neural network, batch sizes of 64 and 256 were experimented with. Moreover, the learning rates were varied between {0.001, 0.005, 0.05, 0.01}. For the fully connected layers, the initial number of neurons was set to 256 and gradually decreased to 4, which gives us the latent representations of the respective spaces. Furthermore, a Leaky ReLU activation function was adopted in which the alpha parameter was fixed to 0.1.

Lastly, as mentioned earlier, an early stoppage approach has been adopted through which the model training is stopped when RMSE on the validation set has not significantly changed for 20 epochs.

4.2 Model Analysis

In this subsection, we discuss the results obtained through experimentation. In particular, we begin by analysing the overall model results received when training the doc2vec review embedding models with the two provided algorithms, i.e., distributed memory and distributed bag of words training algorithms. This is particularly vital to this study as its effectiveness motivates for the inclusion of user reviews in social recommendation. Moreover, we seek to measure the appropriateness of different review embedding sizes for user reviews. These review embedding parameters are varied along with the neural network learning rate and batch size. These results are summarized in Tables III, IV, V and VI.

4.2.1 Distributed Memory VS Distributed Bag of Words Doc2Vec Review Embeddings

As stated earlier, the doc2vec distributed memory and distributed bag of words vectors are trained differently as the distributed memory approach utilises context words in its vector representations. Thus, as expected, during evaluation, much more time was taken by the distributed memory algorithm for training.

Regarding the results presented above, the variation in embedding size does not adversely impact the model's predictions. The outlier in the data is the 128-dimensional review embedding where we see a plumet in output accuracy as well as a substantial increase in both MAE and RMSE. This is observed for both training algorithms; however, the distributed bag of words embeddings seem to perform better in that case. Through inspection, this could be related to the relatively higher learning rate (0.05) used in that particular experiment. We note that the next highest learning rate is 0.01, which was applied on the 256-dimensional embedding experiments; however, the results achieve are comparatively good.

Throughout the evaluation, we see a consistently small MAE and RMSE. Small MAE and RMSE values are desired, as these are a good indication that the model is performing well with predicted values (r'_{ij}) close to the expected values (r_{ij}). For an ideal model, an MAE and RMSE of zero would be best case scenario; however, to achieve such results is difficult.

Given the experimentation results in Table III and IV, where we vary the learning rates and embedding size, we would choose the distributed bag of words algorithm for representing review embeddings. The reasons for this are:

- Computationally, the algorithm is faster.
- The MAE and RMSE obtained from the neural model whilst using distributed bag of words is comparatively similar with a shorter training time.
- With higher learning rates in the neural model, using the distributed bag of words algorithm for review embeddings is able to aid the neural network to achieve substantially better results than its counterpart.

Table III: Results using Distributed Memory with a varied embedding size (d) and learning rate

Distributed Memory					
Parameters			Metrics		
Embedding Size	Learning Rate	Batch Size	MAE	RMSE	Accuracy
32	0.001	64	0.18	0.25	0.975
64	0.005	64	0.09	0.15	0.98
128	0.05	64	1.38	1.7	0.225
256	0.01	64	0.1	0.19	0.97

Table IV: Results using Distributed Bag of Words with a varied embedding size (d) and learning rate

Distributed Bag of Words					
Parameters			Metrics		
Embedding Size	Learning Rate	Batch Size	MAE	RMSE	Accuracy
32	0.001	64	0.15	0.22	0.97
64	0.005	64	0.15	0.21	0.97
128	0.05	64	0.47	0.58	0.635
256	0.01	64	0.076	0.15	0.975

Table V: Results using Distributed Memory with a varied embedding size (d) with a larger batch size

Distributed Memory				
Parameters		Metrics		
Embedding Size	Batch Size	MAE	RMSE	Accuracy
32	256	0.13	0.21	0.963
64	256	0.112	0.21	0.9635
128	256	0.15	0.2313	0.9545
256	256	0.118	0.207	0.9625

Table VI: Results using Distributed Bag of Words with a varied embedding size (d) with a larger batch size

Distributed Bag of Words				
Parameters		Metrics		
Embedding Size	Batch Size	MAE	RMSE	Accuracy
32	256	0.14	0.2228	0.9625
64	256	0.13	0.212	0.9625
128	256	0.1	0.1978	0.9645
256	256	0.122	0.2034	0.965

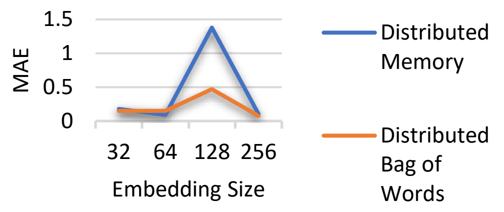


Figure IV: MSE Comparison.

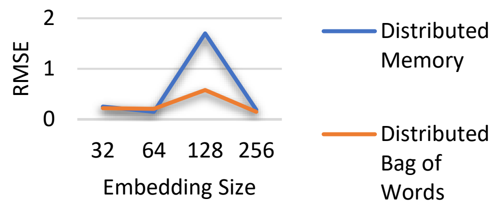


Figure V: RMSE Comparison.

4.2.2 Model Performance on increased Batch Size

The results in Table III and IV utilise small batch sizes when training the social recommender. Thus, keeping the learning rate constant and varying the review embedding size, an experiment was carried out wherein the effectiveness of user reviews was measured on an increased batch size. Table V and VI summarise the results obtained.

It is noticeable that the model trains and performs well on an increased batch size. Furthermore, the loss function, which we seek to minimise, is fitted well on both the 64 and 256 batch sizes. Moreover, the loss function is minimised well as it is relatively small and substantially close to 0. Figure VI depicts the loss function using a 256-batch size.

For both embedding algorithms, the MAE and RMSE obtained from the neural network are comparatively similar and differ by small margins. This suggests that the model is able to learn effective latent representations for rating predictions with a larger batch size irrespective of the review embedding size.

With regards to accuracy, there is a slight drop in the results shown in Table V and IV. However, the drop is not substantial and is still much higher than the 90% mark. Thus, compared to Table III and IV, the 256-batch size performs just as well. Moreover, this can be further attributed to the chosen learning rate of 0.001. As seen in the outlier case provided in the previous subsection, using a too high learning rate may have an adverse effect on the training of the neural network.

Thus, from the preliminary results discussed in this subsection, we can conclude the following:

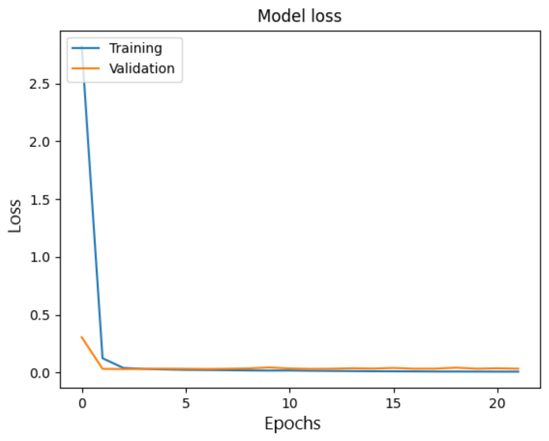


Figure VI: Training and validation loss function with a 256-batch size

- The model scales well on larger data portions as well as larger batch sizes
- There is an adverse effect when the neural networks learning rate is too high. A high learning rate, coupled with a large d-dimensional review embedding tends to decrease the performance of the model substantially.
- Both algorithms for review representation perform equally well, with the distributed bag of words learning algorithm performing better in outlier cases.

4.3 Performance Evaluation

In the previous subsection, we demonstrated the effectiveness of the proposed architecture on the Yelp dataset. The model contained components for latent factor representation learning in which we obtained item space user latent factors (h^{I}_{i}), social space user latent factors (h^{S}_{i}) and review space item latent factors (h^{R}_{i}).

In this subsection, we compare the model against Mod-RevNet and Soc-RevNet the benchmark models discussed earlier. In particular, the left-most fully connected portion of Figure III represents the architecture of the Mod-RevNet. Furthermore, the experiments performed on both models were under similar conditions as those performed on RevNet. Since the Mod-RevNet model contains no review and social embeddings, the experiments carried out measured the model's performance on varied learning rates and batch sizes only. Table VII and VIII summarise the results obtained from the benchmark models with respects to MAE, RMSE and accuracy.

Table VII: Results from the Mod-RevNet Benchmark Model

Parameter	Metric		
Learning Rate	MAE	RMSE	Accuracy
0.001	0.42	0.51	0.598
0.005	0.42	0.50	0.5935
0.01	0.44	0.54	0.5915
0.05	191.59	10825.08	0
Batch Size	MAE	RMSE	Accuracy
256	0.45	0.50	0.5725
512	0.49	0.54	0.5275
1024	0.52	0.60	0.4765

Table VIII: Results from the Soc-RevNet Benchmark Model

Parameter	Metric		
Learning Rate	MAE	RMSE	Accuracy
0.001	0.07	0.19	0.9625
0.005	0.05	0.18	0.9615
0.01	0.07	0.18	0.9645
0.05	81619	128329	0
Batch Size	MAE	RMSE	Accuracy
256	0.11	0.2	0.9605
512	0.13	0.22	0.96
1024	0.15	0.24	0.9585

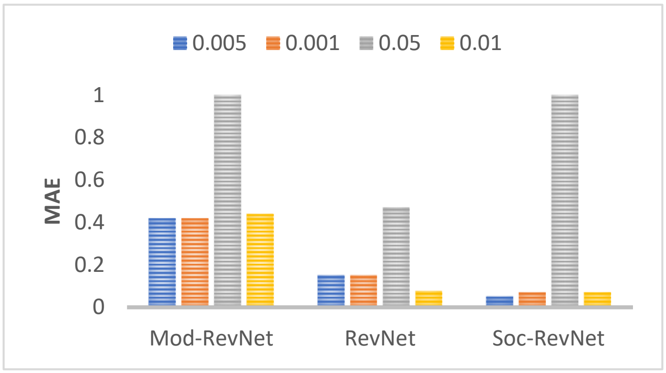
As noted in the previous subsection, the use of a high learning rate adversely affected the performance of RevNet. Regarding Mod-RevNet, we see a similar occurrence; however, the results are much worse. This is presented in Table VII where a learning rate of 0.05, causes Mod-RevNet to perform poorly. This is evident in the Soc-RevNet model as well, as a high learning rate disrupts the overall performance of the model. Furthermore, it is evident that the Mod-RevNet model does not reach the same heights as that hit by the proposed model. The following observations are made on the performance of Mod-RevNet and Soc-RevNet.

- With a high neural network learning rate, the Mod-RevNet model fails to learn an effective latent factor representation for rating prediction. The aforementioned is evident in the data presented for Mod-RevNet as there as an increase in MAE and RMSE with an increase in the learning rate. On the other hand, the Soc-RevNet model generally behaves better than both RevNet and Mod-RevNet. However, with a high learning rate of 0.05, the Soc-RevNet model isn't able to perform as expected. Thus, from the observations above, the general performance of RevNet in higher learning rate environments (0.01 and 0.05) is better than its variants, albeit not always optimal.
- With larger batch sizes, both models tend to increase its MAE and RMSE. This is an indication that the model's predictions are further away than those provided with small batch sizes. However, this is not a major issue as the MAE and RMSE values are relatively small. Regarding Mod-RevNet, we also note that as a result of an increase in error, there is a substantial decrease in accuracy. In particular, there is a 5.1% decrease in accuracy from a 512-batch size to a 1024-batch size. Changes to batch sizes minimally affect Soc-RevNet. Apart from the minor increases in MAE and RMSE, the model maintains a comparatively similar metric values as compared to experiments done on smaller batch sizes.

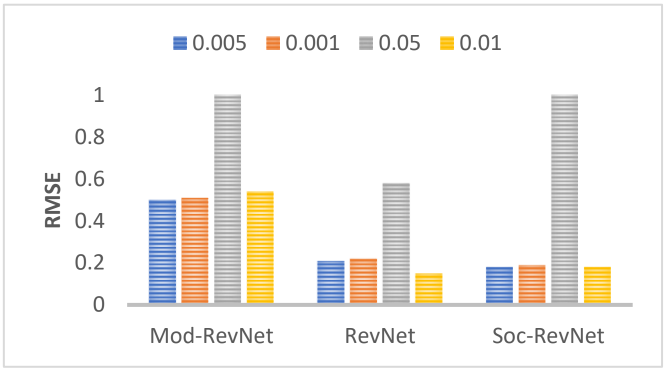
5. DISCUSSION OF RESULTS

In the proposed model, there are components to (i) utilise social relations and (ii) incorporate user reviews on items. As explained earlier, the embedding techniques for both of these components were conducted through the same method. Moreover, the benchmark model, Mod-RevNet, does not contain either of the aforementioned components, whereas Soc-RevNet maintains social relations when making predictions. Thus, we are able to measure the effectiveness of RevNet, with focus on the user-review space review embeddings and social space friendship embeddings. A comparison between RevNet and its variants can be defined as follows,

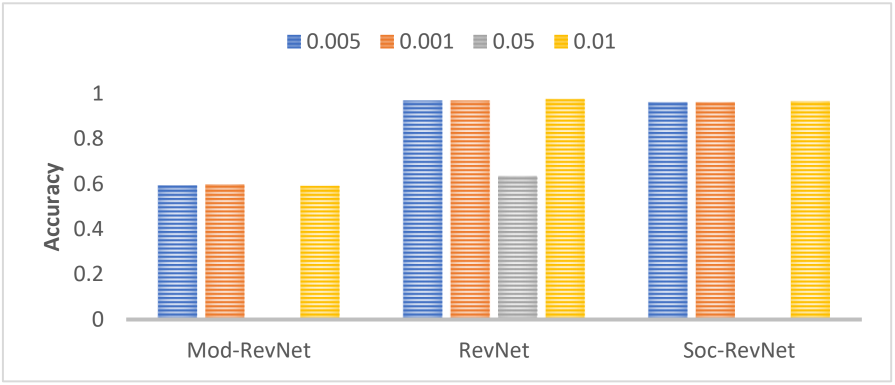
- RevNet: Contains user-item, user-user and user-review spaces through which latent factor representations are obtained and combined for rating prediction.
- Mod-RevNet: There is an absence of social relations (user-user) and user reviews (user-review) in the modified architecture. Thus, the final latent factor (h_i) used for



(a) MAE

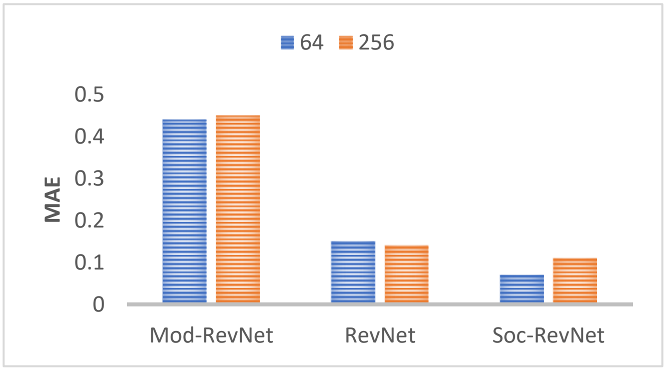


(b) RMSE

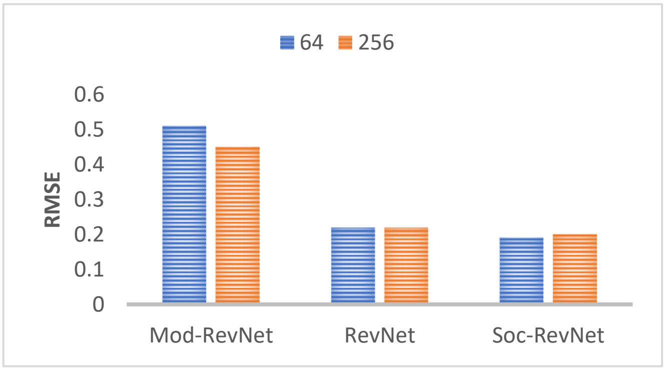


(c) Accuracy

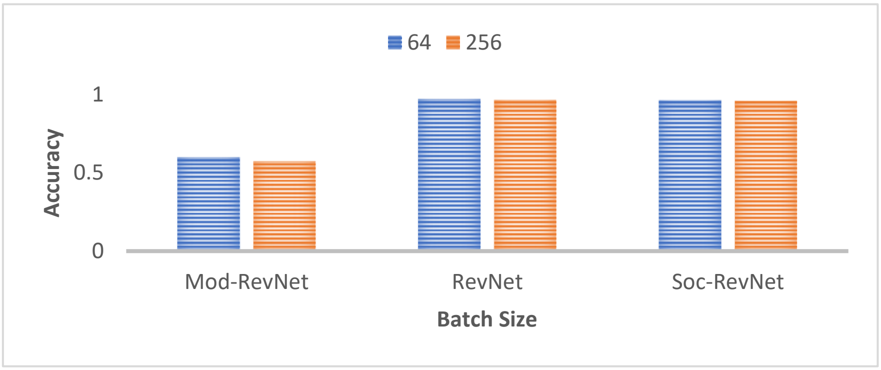
Figure VII: Performance Comparison Between RevNet, Mod-RevNet and Soc-RevNet with a varied Learning Rate



(a) MAE



(b) RMSE



(c) Accuracy

Figure VIII: Performance Comparison between RevNet, Mod-RevNet and Soc-RevNet with a varied Batch Size

Table IX: Model Performance with Equal Parameter Settings

Model	MAE	RMSE	Accuracy
Distributed Memory RevNet	0.09	0.15	0.98
Distributed BOW RevNet	0.15	0.21	0.97
Mod-RevNet	0.42	0.5	0.5935
Soc-RevNet	0.05	0.18	0.9615

rating prediction is dependent on the item space user latent factor (h^I_i) representation only

- Soc-RevNet: Contains user-item and user-user spaces. The final latent factor (h_i) is the concatenation of the item space user latent factor (h^I_i) and social space user latent factor (h^S_i).

The visual performance comparison between RevNet and its variants are depicted in Figure VII and VIII. These figures compare the performance of the models with respect to MAE, RMSE and Accuracy. From the comparison of the three models, we have the following conclusions (i) Friendship embeddings used in a social recommender are effective, (ii) The incorporation of user reviews into a social recommender system does not deteriorate the performance of a social recommender, but rather adds another dimension wherein the model may succeed where a traditional social recommender fails, (iii) There is no convincing evidence that a social recommender system with user reviews is better than a standard social recommender.

Regarding conclusion (i), the use of the social space friendship embeddings for latent factor representation learning is effective and provides a boost to output recommendation performance. We saw a substantial decrease in MAE and RMSE in the Soc-RevNet model compared to the Mod-RevNet model with the introduction of the user-user space. This is further re-iterated in [3], where it has been proven that social relations contained in the user-user space, enhances the performance of the model by being able to learn more effective user latent factor representations as compared to other benchmarks.

Conclusion (ii) regards our contribution towards the social recommender domain. The use of doc2vec embedding techniques to represent user reviews is appropriate, as similar semantically related reviews will have a similar vector representation. Thus, providing a way for the model to understand when a review is good or not and providing an appropriate rating prediction. This is exemplified in the performance comparison between RevNet and Soc-RevNet. Although Soc-RevNet generally has better MAE and RMSE performance, the accuracy achieved by adding review embeddings is worth between one to two percent. Furthermore, the incorporation of user reviews provides better performance on higher learning rates where its variant counterparts fail to perform.

Lastly, in (iii), we state that there is no convincing evidence to suggest that one social recommender model is always better than the other. This conclusion is regarding RevNet and Soc-RevNet specifically. Moreover, we concur that both spaces, user-user and user-review, play an important role in improving output predictions in a recommender model. This statement stands as the absence of the relevant spaces in the Mod-RevNet model deteriorates the overall performance. Regarding Soc-RevNet and RevNet specifically, we saw results in favour of both models where experimental settings were equivalent. In Table IX, where experimental settings are the same, the performance of both models are similar in magnitude. In particular, the Soc-RevNet model has better MAE and RMSE values, whilst the RevNet model has a better output accuracy.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel deep neural network model (RevNet), that utilises user reviews for social recommendation. In particular, the model was designed for item rating prediction. We have provided components for the incorporation of user-item interactions, user-user interactions as well as employed and evaluated the use of doc2vec paragraph representations for review embeddings into the neural network. Experimental results on the Yelp dataset shows that the model performs rather well. Furthermore, RevNet significantly outperforms a modified version of the model, where the user-user and user-review spaces were removed.

To the foregoing, future work in this domain can include the incorporation of user profile data and other rich information that may be available. Furthermore, the use of other neural network architectures can be adopted, and their effectiveness evaluated in the recommender domain. Thus, an extension to this research would be to convert the fully connected architecture into a graph neural network architecture.

ACKNOWLEDGEMENTS

The work described in this paper has been supported through funding made available by the National Research Fund (NRF). A special mention must be made to Dr Edgar Jembere for supervising and guiding the work through the different phases. Lastly, the evaluation of the proposed model would not have been possible without the assistance of the HIPPO cluster made available by the astrophysics department from the University of KwaZulu-Natal.

REFERENCES

[1] Q. Zhang, J. Lu, and Y. Jin, “Artificial intelligence in recommender systems,” *Complex & Intelligent Systems*, vol. 7, no. 1, pp. 439–457, Nov. 2020, doi: 10.1007/s40747-020-00212-w.

[2] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li, “Deep Social Collaborative Filtering,” doi: 10.1145/3298689.3347011.

[3] W. Fan *et al.*, “Graph Neural Networks for Social Recommendation,” *The World Wide Web Conference on - WWW ’19*, 2019, doi: 10.1145/3308558.3313488.

[4] S. Ahmadian, N. Joorabloo, M. Jalili, Y. Ren, M. Meghdadi, and M. Afsharchi, “A social recommender system based on reliable implicit relationships,” *Knowledge-Based Systems*, p. 105371, Dec. 2019, doi: 10.1016/j.knosys.2019.105371.

[5] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. 2012. The role of social networks in information diffusion. In World Wide Web conference. 519–528.

[6] Wenqi Fan, Tyler Derr, Yao Ma, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep Adversarial Social Recommendation. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI-19.

[7] F. Mansur, V. Patel, and M. Patel, "A Review on Recommender Systems," 2017. Accessed: Jun. 12, 2021. [Online].

[8] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," Knowledge-Based Systems, vol. 157, pp. 1–9, Oct. 2018, doi: 10.1016/j.knosys.2018.05.001.

[9] J. Bobadilla, S. Alonso, and A. Hernando, "Deep Learning Architecture for Collaborative Filtering Recommender Systems," Applied Sciences, vol. 10, no. 7, p. 2441, Apr. 2020, doi: 10.3390/app10072441.

[10] S. Kanwal, S. Nawaz, M. K. Malik, and Z. Nawaz, "A Review of Text-Based Recommendation Systems," IEEE Access, vol. 9, pp. 31638–31661, 2021, doi: 10.1109/access.2021.3059312.

[11] L. Chen, G. Chen, and F. Wang, "Recommender systems based on user reviews: the state of the art," User Modeling and User-Adapted Interaction, vol. 25, no. 2, pp. 99–154, Jan. 2015, doi: 10.1007/s11257-015-9155-5.

[12] J. Chambua, Z. Niu, and Y. Zhu, "User preferences prediction approach based on embedded deep summaries," Expert Systems with Applications, vol. 132, pp. 87–98, Oct. 2019, doi: 10.1016/j.eswa.2019.04.047

[13] S. Norvig, ARTIFICIAL INTELLIGENCE: a modern approach. 2018.

[14] L. Wu, W. Member, K. Zhang, and M. Wang, "IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING 1 A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation," Apr. 2021. Accessed: Jun. 21, 2021. [Online]

[15] D. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,” Jan. 2017. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf.

[16] J. Duchi, J. Edu, E. Technion, Il, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization * Elad Hazan,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011, [Online]. Available: https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf.

[17] S. Ruder, “An overview of gradient descent optimization algorithms *.” [Online]. Available: https://arxiv.org/pdf/1609.04747.pdf.

[18] Q. Le and T. Mikolov, “Distributed Representations of Sentences and Documents,” 2014. Accessed: May 11, 2021. [Online].

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, WWW. 173–182.

[20] S. Prabhakaran, “Importance of Text Pre-processing | Pluralsight,” *www.pluralsight.com*, Oct. 02, 2018. https://www.pluralsight.com/guides/importance-of-text-pre-processing.

[21] M. Chen, “Performance Evaluation of Recommender Systems,” *International Journal of Performability Engineering*, 2017, doi: 10.23940/ijpe.17.08.p7.12461256.