Expert Systems With Applications 83 (2017) 300–313

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid recommender system using artificial neural networks

Tulasi K. Paradarami*, Nathaniel D. Bastian, Jennifer L. Wightman

Department of Predictive Analytics, Northwestern University, 405 Church Street, Evanston, IL 60208, USA

ARTICLE INFO

Article history:
Received 6 February 2017
Revised 22 April 2017
Accepted 23 April 2017
Available online 29 April 2017

Keywords:
Information retrieval
Artificial neural networks
Recommender systems
Supervised learning

ABSTRACT

In the context of recommendation systems, metadata information from reviews written for businesses has rarely been considered in traditional systems developed using content-based and collaborative filtering approaches. Collaborative filtering and content-based filtering are popular memory-based methods for recommending new products to the users but suffer from some limitations and fail to provide effective recommendations in many situations. In this paper, we present a deep learning neural network framework that utilizes reviews in addition to content-based features to generate model based predictions for the business-user combinations. We show that a set of content and collaborative features allows for the development of a neural network model with the goal of minimizing *logloss* and *rating misclassification error* using stochastic gradient descent optimization algorithm. We empirically show that the hybrid approach is a very promising solution when compared to standalone memory-based collaborative filtering method.

1. Introduction

When the processing capability of a system is far exceeded by the amount of input, *information overload* occurs; technology has been the primary reason for causing this problem in the recent times. In addition, when a large amount of data is available in a variety of formats and the rate at which the data is produced is too fast for user to efficiently process, it leads to information overload (Edmunds & Morris, 2000). Consequently, when information overload occurs, the quality and effectiveness of decisions suffer (Speier, Valacich, & Vessey, 1999). In the current era of smart devices and Web 2.0, massive amount of data generated from a wide variety of sources, including but not limited to, social networking sites, multimedia sharing sites, hosted services, email, group communications, instant messages. According to Vickery and Vickery (2005), it is getting harder for users to find relevant content on the web, and on many occasions this information overload can cause users to respond to the problem by omitting something important or making an error in the process of making a decision. Whether it is for personal or professional needs, users cannot ignore the information available to them but need intelligent techniques that can efficiently filter the data and present the most relevant information.

A recommender system (RS) is an application that is built to cope with the problem of information overload and provide intelligent suggestions on items to users (Resnick & Varian, 1997; Ricci, Rokach, & Shapira, 2011). This ability of a RS to provide an efficient means to find relevant items is extremely useful, which has led RS to become an important research area that has attracted attention in both academia and industry (Adomavicius & Tuzhilin, 2005). In simple terms, RS generates a personalized list of ranked items for users to buy from, where rank is computed from heterogeneous sources of data acquired from the user like interests, likes, dislikes, and demographics (Ricci et al., 2011). Interest in recommender systems has remained high because of the wide variety of applications that can help deal with information overload by providing personalized recommendations. Examples of these recommendation systems are product recommendations by Amazon.com (Linden, Smith, & York, 2003), movies by Netflix (Amatriain, 2013), news articles (Nanas, Vavalis, & Houstis, 2010), financial services (Felfernig, Isak, Szabo, & Zachar, 2007), and twitter (Gupta et al., 2013). A RS can support a variety of functions that gives reason for service providers to deploy these techniques on their infrastructure. Some of the more important functions of RS are increasing sales by selling more items, improving user experience and satisfaction, and to understand better the users' needs (Ricci et al., 2011).

In this research, we develop a new hybrid RS technique that builds on the capabilities provided by traditional approaches like collaborative filtering and content-based filtering by utilizing the metadata associated with review text to train and build an Artifi-

* Corresponding author.
E-mail addresses: tulasi.krishna.p@gmail.com (T.K. Paradarami), ndbastian@northwestern.edu (N.D. Bastian), jennifer.wightman@northwestern.edu (J.L. Wightman).

http://dx.doi.org/10.1016/j.eswa.2017.04.046
0957-4174/

cial Neural Network (ANN). We develop a multi-categorical classification model that predicts the class of a rating. LogLoss, a convex function, is the cost function minimized by applying stochastic gradient descent and accuracy of predictions is used to measure efficiency of the model. We perform computational experiments using the Yelp Academic Dataset and demonstrate a 75% improvement in the accuracy of predictions when compared against user based collaborative filtering algorithm. From all the models reviewed during the training phase, we choose the model with lowest logloss and utilize these parameter weights for making predictions on the test dataset. We assess effectiveness of the hybrid model by analyzing the percentage of observations with correct predictions. We also assess the effectiveness of these rating predictions when translated to *yes/no* recommendations.

1.1. Literature review

Recommender systems are broadly classified into the following categories based on the underlying technique used for making recommendations (Adomavicius & Tuzhilin, 2005; Burke, 2002; Jannach, Zanker, Felfernig, & Friedrich, 2010):

1. **Collaborative filtering:** It evaluates the relevance of an item for a user based on the opinion of the members of a community (or cluster) (Nanas et al., 2010). It runs on the premise that users with similar interests tend to prefer similar items.
2. **Content-based filtering:** Such systems are developed on the assumption that items with similar characteristics will be rated in a similar way by the users (Wu, Chang, & Liu, 2014). That is, it recommends items that are similar to the ones liked by the user in the past.
3. **Knowledge-based recommender systems:** Such a system recommends products based on specific domain knowledge on how certain item features satisfies users' needs and specifications (Ricci et al., 2011).
4. **Hybrid recommendation systems:** Any combination of two or more techniques described above can be categorized as a hybrid recommendation system (Jannach et al., 2010).

Artificial neural networks (ANN) have a long history, and the research of McCulloch and Pitts (1943) and Cochocki and Unbehauen (1993) are generally considered the beginning of Neurocomputing. ANNs started to gain popularity in 1980s (Cochocki & Unbehauen, 1993) as the computational capabilities of computers started to meet the demands of an ANN. By choosing an appropriate nonlinear activation function(s), ANNs can be modeled to detect complex nonlinear relationships between features and independent variables, to identify higher polynomial features, their interactions, and to leverage the availability of multiple optimization algorithms (Tu, 1996). Therefore, a model built using an ANN is well positioned to learn the complex relationships between users and items, as well as predict better recommendations. Deep-learning is a new and evolving field in machine learning where powerful models are developed by utilizing a deep structured neural network (Theano, 2015).

Extensive work has been done to build recommendation systems using various machine learning techniques such as Bayesian methods (Guo, 2011), clustering (Pham, Cao, Klamma, & Jarke, 2011), ANNs (Gunawardana & Meek, 2009), linear regression (Ge, Liu, Qi, & Chen, 2011), and probabilistic models (Li, Dias, El-Deredy, & Lisboa, 2007). Each method trains a model that learns from the data with the goal of minimizing error while predicting the rating of an item for each user. Hybrid recommendation systems combine two or more recommendation techniques to improve performance and have fewer limitations compared to stand-alone techniques. Below is a list of methods that are commonly used in building hybrid recommendation systems (Burke, 2002):

- **Weighted:** A linear combination of predictions from different recommendation techniques is computed to get the final recommendation.
- **Switching:** Using a switching criteria, the system switches between different recommendation techniques.
- **Mixed:** A list of results from all recommendations derived from applying various techniques are presented as a unified list without applying any computations to combine the results.
- **Feature Combination:** Results from the collaborative technique are used as another feature to build a content-based system over the augmented feature set.
- **Cascade:** Multistage technique that combines the results from different recommendation techniques in a prioritized manner.
- **Feature Augmentation:** Another technique that runs in multiple stages such that the rating or classification from an initial stage is used as an additional feature in the subsequent stages.
- **Meta-level:** Model generated from a recommendation technique acts as an input to the next recommendation technique in the following stage.

Content-based and collaborative recommendation techniques will always suffer from "new item" and/or "new user" problems since both techniques require prior history to produce effective predictions and the hybridization techniques discussed above alleviate some of these limitations.

Balabanović and Shoham (1997) propose a hybrid recommendation engine that performs content-based analysis using user profiles to identify clusters of similar users. This knowledge is used towards building a collaborative recommendation. This approach has the advantage of making good recommendations to users that do not share similarity with other users by building a profile based on content of the items. Melville, Mooney, and Nagarajan (2002) develop a content-boosted collaborative filtering recommendation technique that proposes a method to solve the problem of sparse rating data. For each user, a vector of pseudo user-ratings is created such that if the user has rated an item, that rating is used and if an item is not rated, a pure content-based recommendation system is used to predict the rating. Using the pseudo user-rating vectors for all users, a dense matrix of user-ratings is generated that is then used as input to collaborative filtering to compute final recommendations. User pair similarity is computed using the Pearson correlation coefficient. Harmonic mean weighting is computed to incorporate confidence in correlations, and a final content-boosted collaborative filtering prediction for an active user is generated.

In situations when there is a predictable pattern in user actions like in an e-learning environment, the items a user is interested in depends on how far he/she is in the learning process of any subject/course. Chen, Niu, Zhao, and Li (2014) propose a two-stage hybrid recommendation system for an e-learning environment to recommend items in users' learning process. The two stages of this system are: (1) item-based collaborative filtering to discover related item sets, and (2) a sequential pattern mining algorithm to filter items according to common learning patterns.

Schein, Popescul, Ungar, and Pennock (2002) propose a hybrid recommendation technique that builds a single probabilistic framework utilizing features from both content and collaborative content. Using a new performance metric called *CROC curve*, it is empirically demonstrated that the various components of the framework combine in an effective manner to enhance the performance characteristics of the recommendation systems. Gunawardana and Meek (2009) also propose a new hybrid system using a model-based approach with unified Boltzmann machines, which are probabilistic models that combine content and collaborative information in a coherent manner. Using the content and collaborative information as feature vectors, parameter weights that reflects how

well each feature predicts user actions are learned by model training. This unified approach has an advantage over other approaches because there is no need for careful feature engineering or post-hoc hybridization of distinct recommender systems. Gunawardana and Meek (2009) is constrained to predicting future binary actions by the user, such as buying a book or watching a movie.

Schein et al. (2002), Balabanović and Shoham (1997), Melville et al. (2002), Chen, Chen, and Wang (2015), and Gunawardana and Meek (2009) all propose different kinds of hybrid models that utilize both content and collaborative information to develop a combination of memory and model-based recommendation systems. In this era of Web 2.0, it is common for most websites to give users an option to write reviews of the products in addition to providing a rating. Text analytics and/or sentiment analysis techniques enable the extraction of reviewers' sentiment, latent factors, opinions, and contextual information. In situations where ratings are not available, reviews are used to infer the customers' preference for a product from the opinion expressed in the reviews – referred to as a virtual rating as described by Zhang, Narayanan, and Choudhary (2010); this virtual rating is used in a traditional approach like collaborative filtering.

In situations when both reviews and ratings are available, extensive research has been done by using reviews to augment and enhance ratings with the collaborative filtering technique. Review information is augmented in different ways with ratings such as review helpfulness (Raghavan, Gunasekar, & Ghosh, 2012), context using Latent Dirichlet Allocation (Hariri, Mobasher, Burke, & Zheng, 2011; Moshfeghi, Piwowarski, & Jose, 2011; Zheng, 2014), overall opinion (Blattner & Medo, 2012; Pero & Horváth, 2013), and emotion (Moshfeghi et al., 2011).

Raghavan et al. (2012) propose a two-stage model that first estimates a review quality score using the formula: ratio of "helpful votes" to "total votes" (originally proposed by Kim, Pantel, Chklovski, & Pennacchiotti (2006)) which is weighted with the user rating. For recent reviews that do not have enough votes to compute a quality score, a regression model is trained to predict rating quality score directly from the review text. Various feature extraction methods were evaluated using text mining techniques like bag-of-words, topic modeling, content (metadata) based features, and a hybrid method using both text and metadata-based feature extraction. In the second stage, a probabilistic collaborative filtering model is developed based on a quality score weighted rating.

There has been extensive research performed in the field of recommendation systems developed using contextual information (Adomavicius & Tuzhilin, 2011; Champiri, Shahamiri, & Salim, 2015; Hariri, Mobasher, & Burke, 2012; Hariri et al., 2011; Zheng, 2014). Champiri et al. (2015) conduct a review to identify contextual information and methods for recommendations in digital libraries. Hariri et al. (2011) propose a new model that is context aware and the inferred context is used to define a utility function for the items reflecting how much each item is preferred by a user given his/her current context. Context inference is achieved using labeled LDA, which performed well for the "TripAdvisor" dataset used for this research. Standard item-based k-nearest neighbors is used to compute the utility score for an item i and user u and is defined as a linear combination of $predictedRating(u, i)$ and $contextScore(u, i)$.

Opinion-based recommendation systems is another domain of active research. Pero and Horváth (2013) utilize both ratings and inferred opinions from product reviews to propose a new model using matrix factorization for predicting ratings. Blattner and Medo (2012) formulate within an opinion formation framework where social types play a major role and users' opinions are assembled in two stages (external sources and social interactions).

Moshfeghi et al. (2011) tackle the issues of data sparsity and cold-start by proposing a framework that is an extension of LDA and gradient boosted trees that considers item-related semantics and emotion. They identify semantic and emotion space to construct latent groups of users, and in each space the probability that a user likes an item is computed. Finally, the information from different spaces is aggregated using supervised machine learning techniques like gradient boosted trees.

Amini, Nasiri, and Afzali (2014) and Lee, Choi, and Woo (2002) propose new recommender frameworks that also utilize content/metadata in addition to ratings to improve the accuracy of the item predictions. They both utilize ANN to build a hybrid recommendation engine but differ in what role the ANN plays in their frameworks. Amini et al. (2014) build a hybrid classifier using collaborative and content-based data available in the MovieLens dataset; following classification models were trained and evaluated: spiking neural network, multi-layer perceptron neural network, decision tree, naive bayes. Duch and Jankowski (1999) performed extensive research of activation functions (also referred as transfer functions in the domain of ANN) and Ozkan and Erbek (2003) compared the performance of the following most common activation functions (Civco & Waug, 1994; Kaminsky, Barad, & Brown, 1997): linear, logistic/sigmoid, tangent hyperbolic.

Scalability and performance are key metrics for deploying a solution in real production systems and Lee et al. (2002) propose a two stage self-organizing map (SOM) neural network based recommendation system to improve these key metrics. Lee et al. (2002) initially segment users based on demographics followed by clustering them according to preference of items using a SOM neural network. To recommend items for a user, the user's cluster is first determined and the CF algorithm is applied to all the users whom belong to same cluster in order to predict the user's preference for the items. Experimental results show that the proposed system has better predictability than the traditional CF-based approach and also greatly improves the computational time to calculate correlation coefficients. Lee et al. (2002) differentiate their approach from other research by preprocessing the data to decrease the dimensionality of the user and item space before computing correlation coefficients.

1.2. Research objectives

Collaborative filtering algorithms based on user ratings are hugely popular. However, in this era of Web 2.0 where user reviews are available for most of the products/services, review text can be mined efficiently and used in combination with rating data to deliver high quality recommendations. Reviews are typically written in text form describing assessment of the product or experience of the service provided. However, rating is a homogeneous value and does not capture the sentiment and/or context behind users' experience in a way that a review would. For example, a young family really liked their experience at a restaurant that has child-friendly services readily available and, hence, give a very high rating for the restaurant with following review: "This is an excellent restaurant with child-friendly services that makes it an ideal place to visit for families with young children. However, if you are on date this is not ideal because it can be bit noisy". High quality reviews tend to receive more number of votes from other users who find the review useful in their decision making process and/or share the sentiment expressed in the reviews. Hence, votes related meta-data associated with reviews can be extremely useful features in the development of a supervised learning model.

As far as we know, there is very little research done utilizing the meta-data associated with reviews and ratings in conjunction with user and business attributes to develop a supervised deep-learning rating prediction model. In general, an efficient recommender system should be able to model and capture the complex, nonlinear relationships between users and businesses. ANNs are particularly well-suited to learn about these relationships. In this

paper, we extend the work from previous literature to incorporate quality of ratings, assessed by the number of votes received, in addition to content and collaborative features for businesses and users into the feature space and train a deep learning model using ANNs to predict the rating and evaluate the improvement in recommendation predictions. Further, we demonstrate how well the proposed technique works by performing computational experiments using the Yelp Academic Dataset and comparing its efficiency against a user based collaborative filtering model.

2. Materials and methods

For this research, we consider the problem of predicting the rating as a multi-label classification problem where each rating is treated a label. Logarithmic Loss (or Log Loss) is a classification loss function that quantifies the accuracy of a classifier by penalizing false classifications (Murphy, 2012). Log Loss (*LL*), which is a convex function and can be minimized by stochastic gradient descent, is defined by Eq. [1]

$$LL = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m=5} y_{ij}\log p_{ij} \tag{1}$$

where n is the number of training examples, m is the number of classes (or ratings), p_{ij} is the model probability of rating the sample i as j and y_{ij} is the actual rating j of the sample i. Performance of the classifier is maximized by minimizing *LL*.

2.1. Artificial neural network - supervised learning

ANNs are excellent modeling tools for approximating any nonlinear relationships and finding patterns within the data (Hastings, Mosteller, Tukey, & Winsor, 1947). Let X be the dataset with m reviews and n content and collaborative features, including the *bias* term. A vectorized representation of neurons in a hidden layer is represented by the Eq. [2].

$$h^{(l)} = \sigma(\Theta^{(l)T}h^{(l-1)} + b^{(l)}) \tag{2}$$

where $h^{(l)}$ is a vector such that each element represents a neuron in layer l, $\Theta^{(l)}$ is the weight matrix in layer l such that $\Theta \in R^{i\times j}$ where i is the number of nodes in the hidden layer l and j is the number of nodes in the previous layer (including the bias term). $b^{(l)}$ is the bias term. σ is the activation function. Each neuron in the network is a nonlinear combination of inputs $h^{(l-1)}$ weighted by the parameters θ_i. *tanh, sigmoid, relu* are the activation functions for hidden layers, *softmax* is used for the output layer with the constraint that there are 5 nodes in the output layer - one for each class of rating. An activation function with nonlinear properties is important because of its ability to discriminate relationships in the feature space and strongly influences the complexity and performance of ANN.

For the purpose of the RS, we estimate the probabilities of a customer rating a business into one of the discrete classes in $c \in [1,\ldots,5]$. We use an ANN with 5 neurons in the output layer such that each output neuron corresponds to a rating and the neuron with the maximum probability is the predicted rating for the observation. To estimate these discrete probabilities, *sigmoid* function is generalized in the form of *softmax function* shown in Eq. [3].

$$Pr(Y_i = c|h^{(o-1)}, \Theta^{(o)}) = \frac{e^{\theta_c^{(o)T}h^{(o-1)}}}{\sum_{k=1}^{K=5} e^{\theta_c^{(o)T}h^{(o-1)}}} \tag{3}$$

where $\Theta^{(o)}$ is the weight matrix for the *output* layer.

Gradient descent is a popular first-order optimization algorithm that finds the minimum of an objective function by taking steps proportional to the negative of the gradient at the current point (Curry, 1944). A learning model estimates the weights by computing partial derivatives of the weight vector at each point and stopping when it reaches the minimum of the error function.

$$\theta_{k+1} = \theta_k - \alpha\frac{\partial E}{\partial \theta} \tag{4}$$

where θ is the parameter vector, $E(\theta)$ is the cost function and α is the learning rate. A problem with using gradient descent for optimization is the issue of *local minima*. Gradient descent is an iterative algorithm designed to traverse feature space one step at a time along the gradient such that overall error decreases and the algorithm stops when the steps result in a higher model error. The problem with local minima occurs when the model updates its weights towards the point with minimal error but that point is minimum with the local neighborhood but not globally. This problem is overcome by adding a *momentum* term to the weight updates such that changes to the weight depends on the current error and changes in the previous iteration (Qian, 1999).

In order to compute the derivatives for a neural network, we apply the technique of back-propagation to minimize the cost function defined in Eq. [3] using gradient descent. To compute gradient descent, partial derivatives of the cost function for every weight in each layer of the network, $\frac{\partial C}{\partial \Theta}$, should be calculated. From Eq. [2], let $z^{(l)}$ represent the linear combination of weights and inputs in layer l, such that $z^{(l)} \in R^i$ where i is the number of nodes in layer l.

$$z^{(l)} = \Theta^{(l)T}h^{(l-1)} + b^{(l)} \tag{5}$$

The intuition behind this approach is, error vector at each layer l, as shown in Eq. [6], is propagated backwards from the output layer to the first hidden layer of the network (i.e., delta terms are first computed for the output term and propagate those error terms at each layer). Error at each node also depends on the incoming weights to the neuron and updating these weights will enable the network to learn. Error terms calculated at each node are propagated *backwards* within the neural network using the derivative of the activation function at that node. The back-propagation algorithm uses gradient descent to find weights of the network that minimizes the error in the output layer, hence, the need for activation functions to be differentiable. For the purpose of the recommendation system, we use *softmax* activation function in the output layer and for hidden layers, we performed experiments with different activation functions, such as *tanh, sigmoid, relu*.

$$\delta^{(l)} = \begin{cases} \frac{\partial E}{\partial \Theta^{(l)}} & \text{if } l \text{ is the output layer} \\ \Theta^{(l)T}\delta^{(l+1)} \odot \frac{\partial E}{\partial z^{(l)}} & \text{if } l \text{ is the hidden layer} \end{cases} \tag{6}$$

We developed the hybrid ANN model using Keras (Chollet, 2015) for the API (Application Programming Interface) layer and Theano (Theano Development Team, 2016) as the back-end computation engine. Keras is a Python based neural networks library designed to execute on top of Theano. We ran our experiments on CentOS based linux system with 12 GB RAM and 4 CPUs.

2.2. Similarity measures

Recommendation system implementations can be broadly categorized as in-memory and model-based. In memory-based systems like collaborative or content-based filtering, the initial step is to find users with similar preferences (neighbors). A weighted average of the ratings from users in the neighborhood is computed for the purpose (Carrillo, López, & Moreno, 2013). On the other hand, model based techniques use data mining algorithms to train a model for predicting user recommendations.

Collaborative filtering technique, an in-memory model, was developed by computing pairwise similarities among users and items (Breese, Heckerman, & Kadie, 1998). Weighted arithmetic mean is

commonly used in a RS in conjunction with the similarity metric to compute the predicted rating for a user-item pair.

Let $R(u, i)$ be the rating to be computed for user u and item i. U is the set of users that have rated the item i and $sim(u, u_k)$ is the similarity between the users u and u_k (Ricci et al., 2011; Sorensen, 2012).

$$R(u,i) = \sum_{i \in U} sim(u,u_k) * r(u_k,i) \tag{7}$$

Similarity functions are real valued functions that measure the degree of similarity between a pair of objects and are commonly used in collaborative filtering techniques to measure similarity between users and items. Some of the common similarity measures (Ricci et al., 2011) are *euclidean distance, Minkowski distance, Mahalanobis distance, cosine similarity* and *Pearson correlation*. To analyze the efficiency of an ANN based RS, we develop a reference model using collaborative filtering by computing user similarities using *euclidean distance, cosine similarity* and *Pearson correlation*. In a user based collaborative filtering recommendation system, predictions are generated by applying the formula provided below.

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u_k \in N} sim(u,u_k) \times (r_{u,i} - \bar{r}_{u_k})}{\sum_{u_k \in N} |sim(u,u_k)|} \tag{8}$$

where N is the set of all users that rated an item i, $\bar{r_u}$ is average rating by the user u, $r_{u,i}$ is the rating of item i by user u and $sim(u, u_k)$ is the similarity between u and u_k

Collaborative filtering techniques are hugely popular and perform well where there is sufficient rating information but their effectiveness suffers from following well-known problems:

1. **Data Sparsity:** It is related to the unavailability of ratings for a large number of items (Linden et al., 2003).
2. **Cold-Start:** This problem occurs when a new item is added to the catalog or a new user enters the system (Schein et al., 2002).
3. **Scalability:** Requires very expensive computations and grows polynomially with the number of users and items in a database (Papagelis, Rousidis, Plexousakis, & Theoharopoulos, 2005).

Model-based recommendation systems, in contrast, train a learning model to find patterns in the data and determine a parameter vector on feature dimension such that the prediction error (or loss) is minimized. Predicting the rating can be modeled as either a regression (Amini et al., 2014; Ge et al., 2011; Wu et al., 2014) or classification problem (Resnick & Varian, 1997; Zhang & Iyengar, 2002). Accuracy of predictions is the most important property of a RS and there are a number of options to measure it. Broadly, prediction accuracy measures are classified as listed below (Ricci et al., 2011); in this research, we utilize "accuracy" of a RS to assess it's effectiveness.

1. **Ratings prediction:** If the system predicts the rating of an item by a user (commonly used as 1-star to 5-star ratings), Root Mean Squared Error (RMSE), Sum of Square Error (SSE), Mean Square Error (MSE) and Mean Absolute Error (MAE) are common error metrics (Dooms, Pessemier, & Martens, 2015; Ge et al., 2011; Lee et al., 2002).
2. **Class prediction:** If the system models the predictions as a classification problem, where rating is considered a discrete class, following error metrics are commonly used: Accuracy, Precision, Recall, True Positive Rate (TPR), False Positive Rate (FPR). A comparison curve of TPR and FPR, Receive Operating Characteristic (ROC) curve, illustrates performance of the classifier (O'Mahony & Smyth, 2009).
3. **Ranking prediction:** If the system ranks items according to user's preference, Normalized Distance-based Performance Measure (NDPM), is commonly used as an error metric (Musat, Liang, & Faltings, 2013).

2.3. Exploratory data analysis

We demonstrate the efficiency of a supervised learning approach based ANN to predict the ratings of a business using the Yelp Academic Dataset (Yelp, 2015). This data source contains rich set of relational data objects for businesses, users, reviews, tips and check-ins. For the purpose of the research, business, users and reviews data objects are used. The relational model of these objects for attributes used to build a ANN is shown in Fig. 1. The dataset has a total of 61, 184 businesses from 27 different cities with a total of 366, 715 users that wrote 1, 569, 264 reviews. A business can be categorized as one of the 783 categories such as *Appliances & Repair, Bars, Car Rental, Doctors, Restaurants*, etc., and review content and style can vary for different categories of businesses. For the research, we choose the category *Restaurants* from *PA* to build the model and filter out businesses that are closed. Note that, businesses, users, reviews are relational objects that are stored in separate files and to enable model training, these objects are merged to form a flat structure such that each row represents an observation. This is achieved by performing an "inner-join" between these datasets; it is an operation between two database tables such that all rows that satisfy the *join* condition are returned. Open restaurants from *PA* are filtered from *businesses* and joined against *reviews* using the attribute *review_id*. This intermediate dataset (that is the result of joining *businesses* and *reviews*), is further joined against *users* using the attribute *user_id* to produce the final flat structure used for model training. Finally, the dataset used for model training has following attributes: *business average rating, total reviews for business, total fans for a user, total friends for a user, days user has been yelping since, total compliments received by the user, total votes received by users' reviews, total reviews written by the user, age of the review, total cool, funny, useful votes received by a review and rating associated with the review.* This final dataset has a total of 42, 337 reviews for 1, 116 open restaurants and 13, 869 users. An interesting observation is that *NV* has only 22% of the businesses but 48% of the users belong that state who wrote 42% of the total reviews.

Skewness is a measure of symmetry such that *skew* is 0 when the distribution of the variable is perfectly symmetric, negative for long tail on the left side and positive for long tail to the right (DeCarlo, 1997). Skewness of a population is defined as

$$\gamma_1 = \frac{\mu_3}{\mu_2^{\frac{3}{2}}} \tag{9}$$

where kth *central moment* is defined as

$$\mu_k = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^k \tag{10}$$

Number of stars in the review dataset, which is the response variable, is the rating provided by the user for a specific business. An analysis of star rating shows a surprisingly negatively skewed distribution in favor of 4 or 5 stars, as shown in Fig. 2. It is observed that 66% of the ratings are 4 or 5 stars such that 4 is the most preferred star rating with close to 35% of all ratings having 4 stars and only 7% of the ratings have 1 star. Skewness is also computed for each of the predictor variables used for model development as shown in Table 1.

As shown in Fig. 3, compared to previous years, between 2011 and 2014, there was an increase of 163% in the number of reviews written. However, the average number of useful votes per review decreased from 1.14 to 0.67 representing a decrease of 41%, which leads us to believe that either the overall quality of reviews has

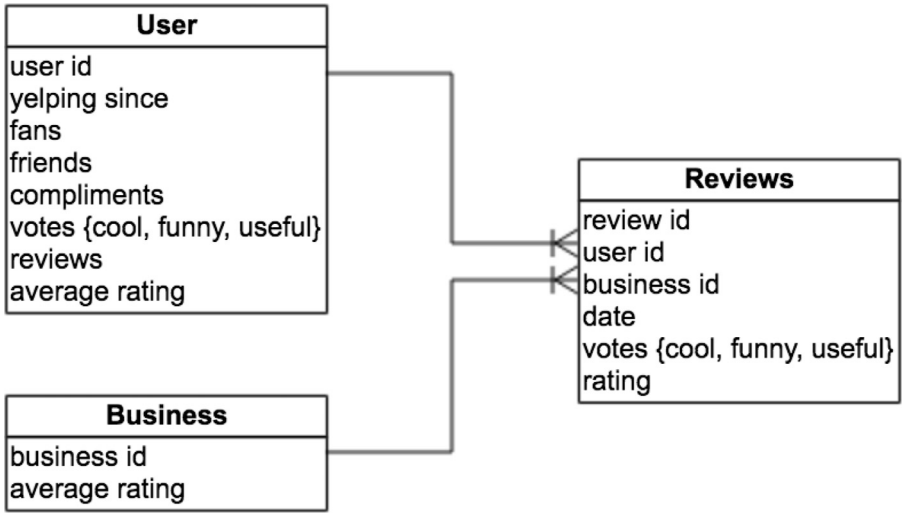

Fig. 1. Yelp dataset entity-relationship diagram.

Table 1
Descriptive statistics for the features used to build ANN model.

Dataset	Feature	Mean	StdDev	Median	Min	Max	Range	Skew
Business	Avg. rating	3.73	0.52	4	1	5	4	−0.68
Business	Total review	132.05	130.76	93	3	695	692	2.27
User	Total fans	6.53	31.37	1	0	1298	1298	20.16
User	Total friends	36.75	144.8	3	0	2103	2103	6.97
User	Yelping since	1729.07	720.46	1675	365	4109	3744	0.32
User	Total compliments	93.43	775.31	3	0	41,514	41,514	36.82
User	Total votes	516.3	2446	53	0	86,200	86,200	21.04
User	Total reviews	113.98	201.61	33	1	4573	4572	4.91
User	Avg. rating	3.74	0.62	3.77	0	5	5	−1.26
Review	Days since review	1134.73	626.17	991	358	3895	3537	0.99
Review	Total cool votes	0.39	1.02	0	0	29	29	6.4
Review	Total funny votes	0.28	0.88	0	0	25	25	7.07
Review	Total useful votes	0.87	1.57	0	0	48	48	5.23
Review	Rating	3.73	1.2	4	1	5	4	−0.78

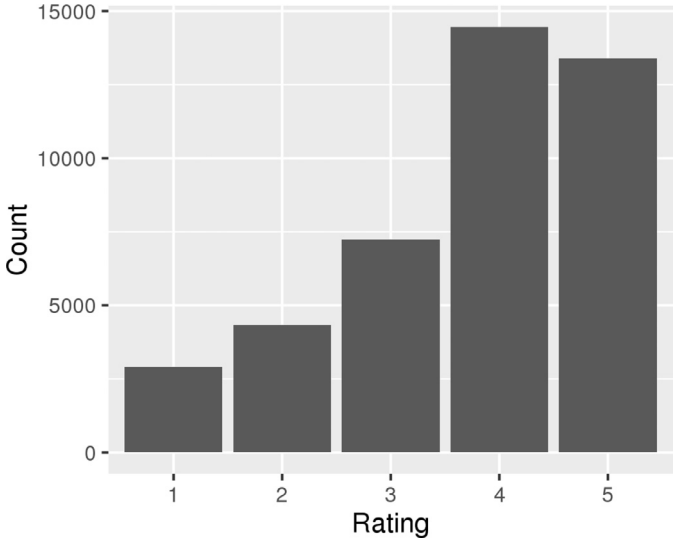

Fig. 2. Skewness in the distribution of business review ratings.

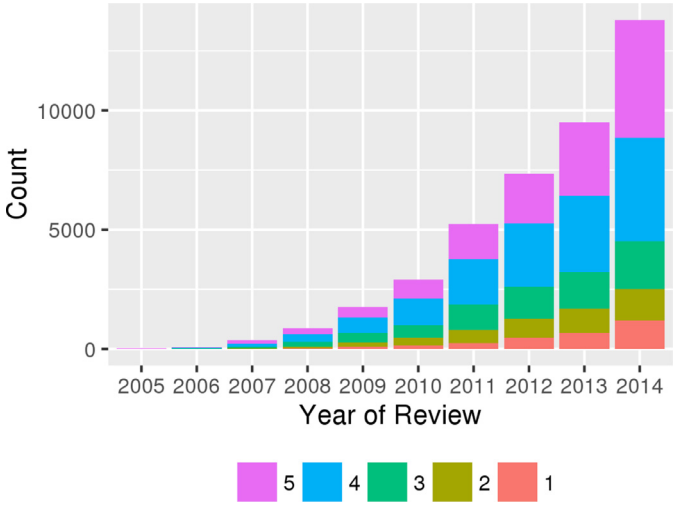

Fig. 3. Yearly ratings categorized by the star rating.

reduced during these years or more likely, users simply did not find time and/or need to assess and collaborate on the increased number of reviews. It is also interesting to observe that, during the same time, the average rating barely changed from 3.7184 to 3.7634. Descriptive statistics for features used to train the ANN model are shown in Table 1.

Close to 60% of the restaurants receive 25 or fewer reviews, and there are only 10 (1%) restaurants that have 250 or more reviews. On average, a business receives 38 reviews with an average rating of 3.6. Distribution of the number of reviews received per business

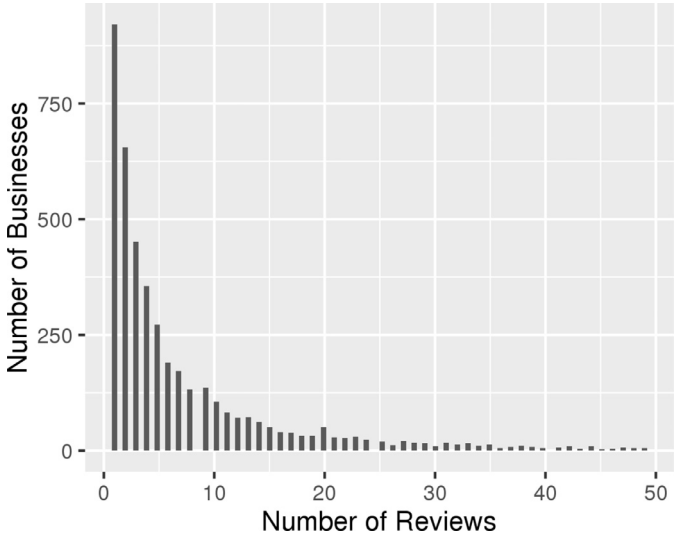

Fig. 4. Histogram of number of reviews per business.

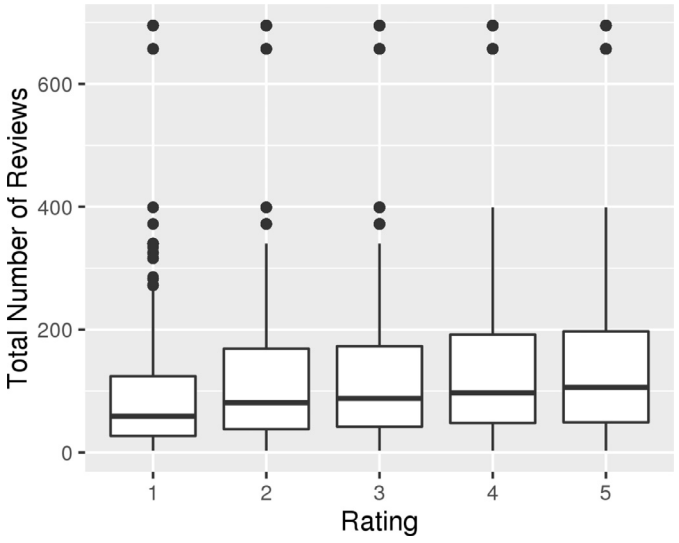

Fig. 5. Outliers in the number of reviews written per star rating.

is shown in Fig. 4 and the presence of the outliers for this feature is shown in Fig. 5.

Outliers are data objects that have some characteristics that are different from most of the other data objects, and they will have an influence on the accuracy of a supervised learning model (Rousseeuw & Leroy, 1987). To minimize the influence of these outliers on the model estimates, we transform the data using *winsorising*, that is, extreme values are clipped to a threshold percentile of 95 (Hastings et al., 1947).

The Yelp dataset has content available for users that have been writing reviews from October, 2005 through January, 2015; for the purpose of model development, the feature *yelping-since* is converted to number of days from January, 2015. It is observed that 82% of the users write 50 or fewer reviews and the average number of reviews written by a user is 10.2. The average rating provided by users that wrote at least 10 reviews but not more than 100 reviews is 3.8.

A striking observation from Fig. 6 is, the likelihood of a user providing a rating of 5 is high when they are new to the system or number of reviews written by the user is less than 5. Another explanation for this behavior is the likelihood of the presence of fake ratings and reviews in the system where users with an affiliation to the business provide high rating and do not participate in reviewing any other businesses. We are also interested in analyzing if there are any seasonal trends in the feature *yelping since* and it is observed that this feature has a reasonably uniform distribution across all months of the year, with a marginal 10% increase in users in July and 30% increase of users in December when compared against the number of active users in the previous months.

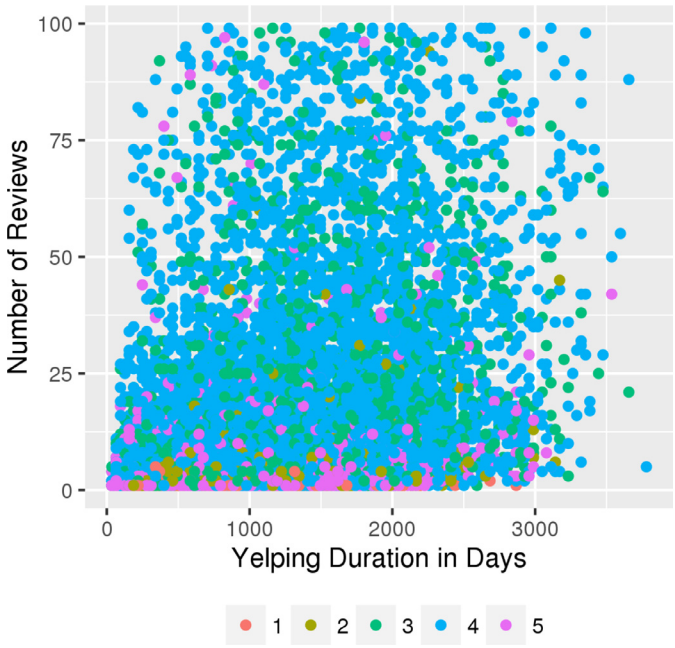

Fig. 6. Distribution of ratings for all users with a maximum of 100 reviews.

Table 2
Mis-classification rate per similarity metric for user based collaborative filtering model.

Similarity Metric	Misclassification (%)
Pearson	35.60
Cosine	57.16
Euclidean	74.43

In an ANN, if the range of a feature is too high, any change to the weights will have a relative influence on the features with larger magnitudes. Training of an ANN is more efficient with better predictors and objective functions, like gradient descent, work well and convergence is usually faster with *scaled* features. Saturating hidden nodes fail to identify and differentiate patterns in the data, which leads to little or no learning and scaling features prevents premature saturation of hidden nodes. As shown in Table 1, a number of features have a high range of values and will lead to inefficient training without transforming the data. There is no standard approach for scaling features but for this research we adopt *min-max normalization* technique because it linearly transforms the data and preserves the relationship from original data. This is a computationally simple technique that can fit data within a uniform range. To improve model performance and convergence rate, we perform *feature rescaling* using the formula [11] to standardize the data such that the feature vector is transformed to have values in the range [0, 1] (Table 2).

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{11}$$

Table 3
Confusion matrix for restaurants in PA using Pearson similarity.

Restaurants		Predicted rating				
		1	2	3	4	5
Actual rating	1	31	38	30	1	
	2	2	39	58	1	
	3		12	83	5	
	4			16	58	26
	5			1	21	78

Table 4
Confusion matrix for restaurants in PA using Cosine similarity.

Restaurants		Predicted rating				
		1	2	3	4	5
Actual rating	1	22	20	39	19	1
	2	2	14	48	35	1
	3		4	41	53	2
	4		2	23	67	8
	5		1	12	56	31

Table 5
Confusion matrix for restaurants in PA using Euclidean distance.

Restaurants		Predicted rating				
		1	2	3	4	5
Actual rating	1	16	13	34	35	3
	2	1	8	31	55	5
	3		4	32	57	6
	4		7	41	47	5
	5		7	43	50	7

3. Experiments

3.1. Baseline model using collaborative filtering

Neighborhood based algorithms are a popular choice in collaborative filtering recommendation systems. To assess the performance and efficiency of an ANN based learning model, we compare it's performance against a reference model developed using user-based collaborative filtering algorithm for PA based restaurants. A neighborhood for two users is the common set of restaurants that both users have rated. Based on the common neighborhoods, pairwise user similarities are computed using Pearson correlation, cosine and euclidean distance. Predictions are computed as weighted average of users in the neighborhood such that similarity between users is the weight as shown in the Eq. 8.

We observed that Pearson correlation based model produced the best results with an accuracy of 64.40% and euclidean based model performed worse than both Pearson and cosine based models with an accuracy of only 25.57%. Since the Pearson based model has the best accuracy, we use this as a reference to compare and analyze ANN based model. For nearly 6% of the observations, Pearson correlation based model incorrectly predicted a rating of either *1 or 2 or 3* when actual rating is *4 or 5* – this can lead to a potential loss of revenue for businesses because users are not recommended restaurants that are of interest to them. On the other hand, close to 1% of predictions were either *4 or 5* when actual rating is *1 or 2 or 3* – this can result in users losing confidence in the recommendations. Refer to Tables 3–5 for confusion matrices of the models developed using *Pearson, cosine* and *euclidean* similarity metrics respectively where each cell represents the percentage for each actual rating.

3.2. Model parameters optimization

We adopt the following model naming convention: [activation-function]-Model[number of hidden nodes]-L[learning rate]-D[dropout] while discussing the results. For example, *T-Model16-L0.01-D0.2* is a model with activation function *tanh*, 16 hidden nodes with a learning rate 0.01 and dropout probability of 0.2 at each hidden layer.

We use the 13 features identified in Table 1 for building the ANN supervised learning model to predict the class of star rating for a business. Response variable of the model is the *star rating* and predictor variables are: business average rating, total reviews for business, total fans for a user, total friends for a user, number of days user has been yelping, total compliments received by the user, total votes received for user's reviews, total reviews written by the user, age of the review, total cool, funny, useful votes received by a review. Restaurants from *PA* are split into train/validation datasets such that 70% of the data is used to train the model and remaining 30% is used to validate model performance. At each epoch, model's performance is evaluated against the validation dataset and training stops, when the model performs adequately against the validation dataset. Using these model parameters, we further analyze the model's effectiveness, by assessing its performance against a test dataset of all restaurants outside *PA*. Note that, test dataset does not play a role in the model training. We further analyze the performance of trained model for *restaurants* in *PA* against other categories of businesses in different states and demonstrate that the concepts, observations and results observed for *restaurants* are valid and can be extended to any of the other 782 categories. From a total of 782 non-restaurant categories, we choose categories with at-least 2000 open businesses.

We proceeded to run experiments while adjusting following parameters to find the best model:

- Number of neurons per hidden layer
- Activation function
- Drop-out probabilities

The dataset was trained using 3 different network topologies such that each topology had 3 hidden layers with [4,4,4], [4,8,4] and [4,4,8] hidden nodes; all networks are fully connected and contain a *bias* node in the hidden layers. Each network was initialized with a random set of weights following a *uniform* probability distribution. For each of the network topologies, experiments were run using the activation functions: *rectified linear unit (relu), tanh, sigmoid* for the hidden layers and *softmax* activation function for the output layer. Standard neural network architectures such as fully connected multi-layer perceptron with a large number of hidden nodes are susceptible to a problem called *overfitting* (Geman, Bienenstock, & Doursat, 1992). Regularization, early stopping, and dropout are various techniques applied to prevent a network from overfitting.

A weight regularizer improves model fitting by applying penalties (*L1, L2, elastic net*) on layer parameters, and these penalties are incorporated into *logloss* function before computing the 1st derivative that is used by gradient descent. *L*1 regularization adds *L1-norm* penalty to the loss function and performs feature selection by forcing weak features to have zero coefficients. On the other hand, *L*2 regularization adds *L2-norm* penalty resulting in coefficients of the features to be more evenly spread out. Since the dataset has a reasonably small number of features and *L2-norm* ensures correlated features tend to produce similar coefficients, we chose *L*2 regularization with a parameter value of 0.01 for hidden nodes and the bias node. In addition to shuffling training data between epochs, a dropout technique is utilized since it prevents over-fitting a model and provides a mechanism for combining different neural network architectures in an efficient

manner (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). When dropout technique is enabled, hidden units and their connections are randomly dropped resulting in samples of sub-networks that are used in training. The effect of averaging the predictions from these sub-networks can be approximated by using the full-network with smaller weights. For each network topology, experiments were run for the dropout probabilities of: 0.1, 0.2, 0.3. Some of the commonly used criteria for early stopping of the model training are:

- Stop when generalization loss exceeds certain threshold
- Stop when the generalization loss increases for *n* successive steps
- Stop when the decrease in generalization loss is below certain threshold for *n* successive steps

In this research, we chose to define early stopping criteria based on the stability achieved by the model within a parameterized window size.

- For a window size of 50 epochs, stop when the ratio of minimum validation error (log loss) to the average validation error in a window is within the range [0.99, 1.01] for *3* successive windows.

First, we experimented and compared the model performance after changing the number of hidden nodes per layer (while the number of hidden layers is kept constant at 3). This experiment was intended to check if there is a significant improvement in the model performance by increasing number of nodes. We developed and validated models with 4, 8, 32 and 64 hidden nodes per layer and each model was trained for 3, 000 epochs. As the number of nodes increased, overall model performance and accuracy of rating predictions improved too but we did not observe a significant improvement in overall performance or validation loss between a model with 12 and 192 hidden nodes. In addition, increasing the number of hidden nodes to 192 raised the model training time from 10 to 12 min to 4 h. Hence, we continued our experiments with architectures containing no more than 24 hidden nodes. Second, we compared the performance of the models with different activation functions *relu, tanh, sigmoid* and observed that the activation function had a significant impact on the overall model performance and accuracy. Across all experiments, we observed that *relu* and *tanh* produced better models with higher accuracy when compared to *sigmoid*. We also observed the convergence rate and model consistency is much better when using *relu* or *tanh* compared to *sigmoid*.

Fig. 7 demonstrates the performance of the model with *sigmoid* activation function. The model appears to be learning very little over 3, 000 epochs after achieving the lowest validation loss of 1.468223 at epoch 12, and this observation is further emphasized by reviewing the model's stability shown in Fig. 8. A well-performing, stable model would have training and validation curves hovering around 100% but the stability plot for *sigmoid* goes up towards 102% very early in the training process and fails to improve. On the other hand, models utilizing *relu* and *tanh* delivered results with better performance and accuracy than *sigmoid*. We also observed that *tanh* produces results that have fractionally better *loss* and *accuracy* compared to *relu*.

Finally, we ran experiments using dropout probabilities of 0.1, 0.2 and 0.3 but this parameter did not have a significant impact on the model performance.

3.3. Hyperparameters optimization

All experiments were run using the *stochastic gradient descent* optimization algorithm by varying the hyperparameters such as

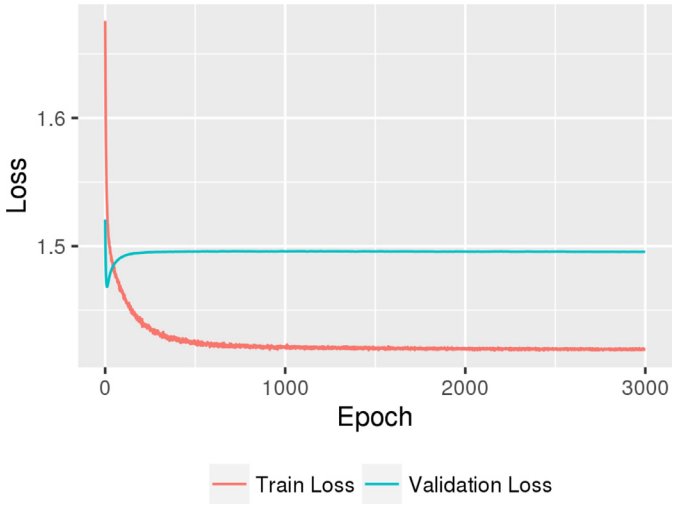

Fig. 7. Comparison of model performance between training and validation datasets for network with 12 hidden nodes (4,4,4) and learning rate of 0.0001 in Stochastic Gradient Descent using *sigmoid* activation function.

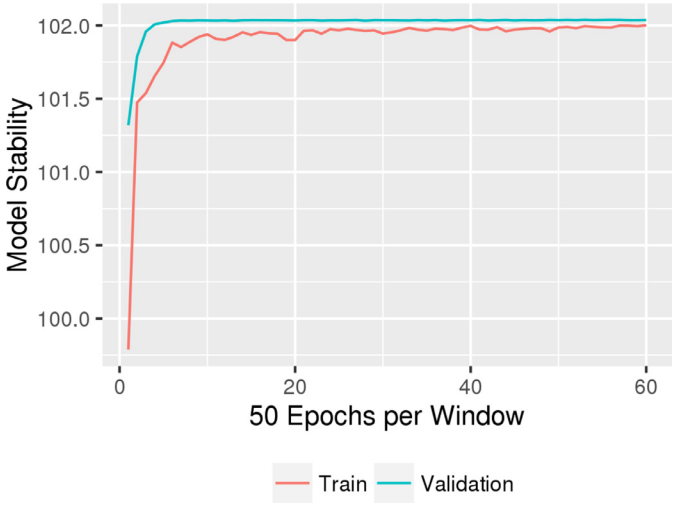

Fig. 8. Analysis of stability of the model with 16 (4,4,8) hidden nodes with dropout probability of 0.1 over *3000* epochs when using *sigmoid* with learning rate *0.0001*.

- Learning Rate: It is size of the step towards minimum values along the gradient and determines how fast or slow the model iterates towards the optimal weights
- Decay: It is the decrease in learning rate over iterations
- Momentum: Adds a fraction of the previous weights to the weights in current iteration by increasing size of steps towards minimum.

Experiments were run for decay in the range $[e^{-3}, e^{-6}]$ and momentum in the range [0.3, 1.0]; we did not find significant change in the model performance by varying these hyper-parameters. So, decay and momentum were set to e^{-6} and 0.9 respectively.

Learning rate is another parameter that we observed had strong influence on the model performance. Optimizing learning rate is more challenging compared to choosing an activation function or model architecture because of the inter-dependencies between these three parameters. Large values for learning will cause the algorithm to overshoot optimal solutions by taking huge steps in the weight space and on the other hand, small learning rate can increase overall training time of the model.

Experiments were also run for a learning rate within the range $[e^{-1}, e^{-4}]$ and we observed that the learning rate of e^{-1} is too high

Fig. 9. Comparison of model performance between training and validation datasets for network with 16 hidden nodes (4,4,8), learning rate of 0.1, dropout probability of 0.1 in Stochastic Gradient Descent using *tanh* activation function.

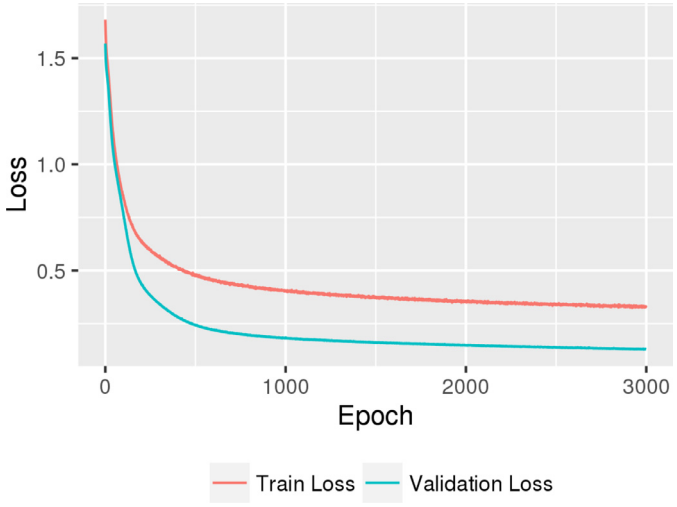

Fig. 10. Comparison of model performance between training and validation datasets for network with 16 hidden nodes (4,4,8) and learning rate of 0.0001 in Stochastic Gradient Descent using *tanh* activation function.

since it results in a model with high variation and severe oscillation in validation loss between epochs irrespective of the chosen architecture or activation function. Even though this behavior was less pronounced in the models with higher dropout probability of 0.3, none of the models had stable enough validation error to be considered for further evaluation. As the learning rate was lowered to e^{-3}, model stability and convergence rate increased resulting in a model that learns in a steady fashion with decreasing training and validation errors over 3000 epochs. Figs. 9 and 10 demonstrate the effect of varying learning rates on the model performance.

In Fig. 9, the model, *T-Model16-L0.1-D0.1* achieves the global minimum at the epoch 2800 and average error within 50 epoch window decreases from 1.2 to 0.9, however, there is no visible trend of decreasing error within a reasonably small window size. Experiments were further conducted by first decreasing learning rate to 0.01 and then to 0.001. With a learning rate of 0.01, it was observed that for models with more than 20 hidden nodes, the plot for training and validation loss is much smoother to indicate a model with steady learning properties. However, for models with less than 20 hidden nodes, learning rate of 0.01 still remained too

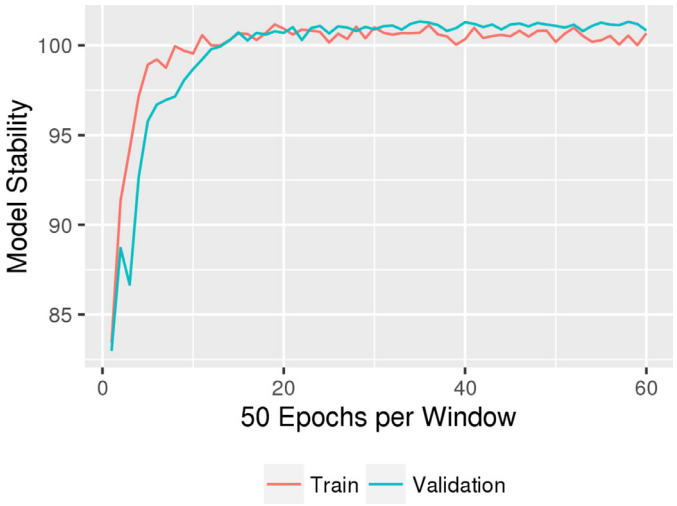

Fig. 11. Analysis of stability of the model with 16 (4,4,8) hidden nodes with dropout probability of 0.1 over *3000* epochs when using *tanh* with learning rate *0.0001*.

high. Finally, a learning rate of 0.0001 was found to produce a stable model as shown in Figs. 10 and 11.

4. Results and discussion

Results from the initial set of experiments are presented in Tables 6–8 sorted by increasing validation loss. *T-Model16-L0.0001-D0.1* has the best performance with lowest validation loss of 0.128386 achieved at epoch 2, 986 and the training error at this epoch is 0.329144. We also observed that validation loss continues to get better even at 3, 000 epochs as shown by the very high epoch at which lowest validation loss is observed. However, the rate of model improvement is reasonably stable once it is past 1, 000 epochs. As shown in Fig. 11, the model performance improves rapidly until it reaches epoch 1000 and becomes fairly stable state beyond that; hence, we chose to stop training the model at 1, 000 epochs and use the weight parameters at that epoch to predict *ratings* on the test dataset.

This model's performance is evaluated against test dataset of *restaurants* from the Yelp dataset. For this test data, model results in a loss of 0.142414 with an accuracy of 91.35%. Accuracy of a classification model is the percentage of correct predictions in the total population. Confusion matrix for the results from test dataset is shown in Table 9 and it is observed that the error rate is primarily contributed by misclassifying 8.65% of the ratings as *1* when the actual rating was *2*. Note that, for developing a RS, ratings *1, 2, 3* belong to the class of *not* recommending the business and misclassification of ratings within this group can be safely ignored. We infer the reason for incorrect classification of the rating *1* is, ratings in the dataset are skewed with over *65%* of the overall ratings classified as *4, 5* and only *7%* of businesses receiving a rating of *1%*; this makes it really difficult for the model to predict the rating *1* compared to other ratings.

Predicting ratings as a multi-categorical classification problem instead of a binary yes/no recommendation has its advantages. It allows the engine to perform deeper analysis on the predictions to build more efficient recommendation systems. For instance, an engine can rank the recommendations such that businesses with prediction rating of *5* are presented to the user ahead of the businesses with prediction rating of *4* which would not be possible if the model only presented *yes* or *no* prediction. It is also worth noting that, 15% of the predictions belong to the rating *3*, which can also be treated as neutral instead of a negative. In general, pre-

Table 6
Model performance comparison using *tanh* activation.

Architecture	Dropout	Validation loss (min)	Epoch	Train loss (at epoch)
3 layers (4-4-8)	0.1	0.128386	2986	0.329144
3 layers (4-4-4)	0.1	0.157782	2995	0.364658
3 layers (4-8-4)	0.1	0.162981	2997	0.365540
3 layers (4-4-8)	0.2	0.183043	2987	0.438139
3 layers (4-8-4)	0.2	0.210498	2990	0.464502
3 layers (4-4-4)	0.2	0.266450	2975	0.488699
3 layers (4-4-8)	0.3	0.311851	2971	0.533364
3 layers (4-8-4)	0.3	0.319071	2990	0.558623
3 layers (4-4-4)	0.3	0.389754	2991	0.578998

Table 7
Model performance comparison using *relu* activation.

Architecture	Dropout	Validation loss (min)	Epoch	Train loss (at epoch)
3 layers (4-4-8)	0.1	0.326469	2989	0.613846
3 layers (4-4-8)	0.2	0.460897	2980	0.702346
3 layers (4-4-8)	0.3	0.483404	3000	0.816316
3 layers (4-8-4)	0.1	0.520402	2993	0.801872
3 layers (4-4-4)	0.1	0.627786	2983	0.860996
3 layers (4-4-4)	0.3	0.696202	2967	0.944039
3 layers (4-8-4)	0.2	0.717184	2990	1.001955
3 layers (4-4-4)	0.2	0.964594	2991	0.997226
3 layers (4-8-4)	0.3	1.073402	2983	1.244359

Table 8
Model performance comparison using *sigmoid* activation.

Architecture	Dropout	Validation loss (min)	Epoch	Train loss (at epoch)
3 layers (4-4-4)	0.1	1.468223	12	1.529468
3 layers (4-4-4)	0.2	1.468750	12	1.540765
3 layers (4-4-4)	0.3	1.469371	12	1.555953
3 layers (4-8-4)	0.3	1.480628	63	1.511215
3 layers (4-8-4)	0.2	1.482010	63	1.504822
3 layers (4-4-8)	0.3	1.483327	26	1.529057
3 layers (4-8-4)	0.1	1.483435	63	1.497427
3 layers (4-4-8)	0.2	1.483929	26	1.517042
3 layers (4-4-8)	0.1	1.484672	26	1.507077

Table 9
Confusion matrix of test dataset for restaurants *not* in PA (each cell represents number of observations).

Restaurants		Predicted rating				
		1	2	3	4	5
Actual rating	1		72,631			
	2		80,608			
	3			127,798		
	4				270,205	
	5					288,426

dicting ratings provides flexibility for the engine to interpret and generate a recommendation customized to the line of business.

This model is trained on restaurants and also tested using businesses in the same category but to build an efficient recommendation engine, a trained model should be able to generalize and perform well against categories not observed during training phase. To further evaluate performance of the model and assess its ability to generalize towards any business, it is evaluated against categories other than *restaurants*. For this purpose, we shortlist categories with atleast 2000 open businesses and are least related to a restaurant type of business. Categories that satisfied these conditions are: *Active Life, Automotive, Beauty Spas, Event Planning Services, Fashion, Health Medical, Home Services, Local Services, Night Life, Shopping*. Performance of the model against each category of business is presented in Figs. 12, 13 and Table 10.

We observed that businesses categorized as *Active Life* have the lowest misclassification rate of 7.96% and *Home Services* have the highest misclassification rate of 23.53%. Confusion matrix for each of the test categories is presented in Fig. 12, where each cell represents the percentage of observations. We observed that error rate in all categories of businesses is the result of incorrect classification of rating *1* as *2*. We believe, model performance can be improved further by balancing the number of observations for rating *1* in the training dataset. However, a recommendation engine will classify both ratings *1, 2* as "not-recommended", hence, the cost of misclassifying a rating *1* as *2* is negligible.

Finally, we compare the results between hybrid recommendation system using ANN and baseline model developed using user-based collaborative filtering with Pearson similarity. ANN had a misclassification rate of only 8.65% on the *test* dataset whereas baseline model's misclassification rate was as high as 35.60%; this represents a 75% improvement in predicting user ratings when using the hybrid ANN. It is also very important to note that, in the collaborative model, just over 7% of the total predictions were misclassified when the predicted ratings are transformed to yes/no recommendation. Whereas, in the hybrid ANN model, there were not a single instance of such misclassification. The main drawback with collaborative filtering algorithm is the lack variety in the data input to the model; it depends entirely on rating to make the predictions. On the other hand, ANNs find patterns within the training data represented by model weights and utilize this information during predictions to produce very accurate results.

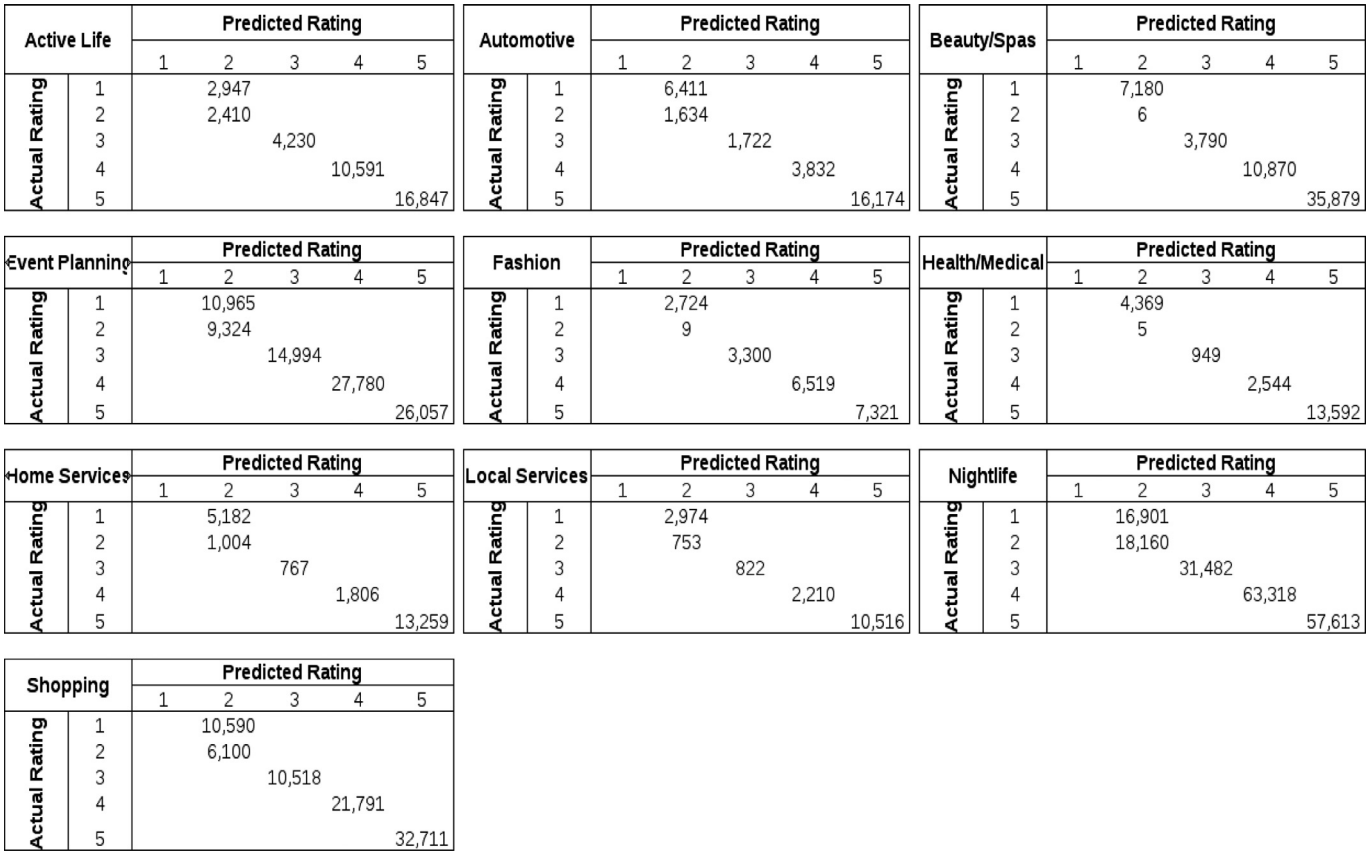

Fig. 12. Confusion matrix for test categories.

Table 10
Model evaluation against test categories.

Category	Number of businesses	LogLoss	Misclassification (%)
Active life	2266	0.242648	7.96
Night life	3531	0.296292	9.02
Beauty spas	4425	0.274124	11.74
Event planning services	2278	0.347313	12.3
Fashion	2289	0.33538	12.52
Shopping	8022	0.325736	12.96
Local services	2062	0.339758	17.22
Health medical	3102	0.372714	19.41
Automotive	2841	0.414752	21.53
Home services	2805	0.429974	23.54

5. Conclusions and future work

In this section, we summarize the contributions of the research work and present future research problems within the domain of recommendation systems.

5.1. Conclusions

In this research, we proposed a novel approach for predicting the ratings using an artificial neural network framework, where both collaborative and content-based features are used to train the model. This methodology brings together content (user and business), collaborative (review and votes) and metadata associated with ratings under the framework of a unified supervised learning model that produces better prediction results than memory-based collaborative filtering recommendation systems. We showed that because of the effective learning capabilities of an ANN framework, from the collaborative and content-based features we can significantly improve the overall effectiveness of the recommendation system. We also demonstrated that an ANN trained for a particular category of business can generalize well and act as the recommendation system for other categories and any user in the system – which significantly improves the scalability of the application.

We performed a number of experiments using the Yelp academic dataset by varying the model hyper-parameters, such as the number of hidden layers, number of nodes, number of epochs, learning rate, and dropout probability. We optimized these parameters by validating the model performance against test and validation datasets such that the misclassification error rate was reduced to less than 9% and none of the predictions crossed boundaries between the binary classification category of recommending (ratings 4 or 5) and not recommending (ratings 1 or 2 or 3) a business to the user. We further validated the ability of this model to generalize to categories other than restaurants and observed that it generalizes well. In spite of an increase in the misclassification error rate in ratings for most of the non-restaurant related categories, we did not find any instance of misclassification across binary classification categories.

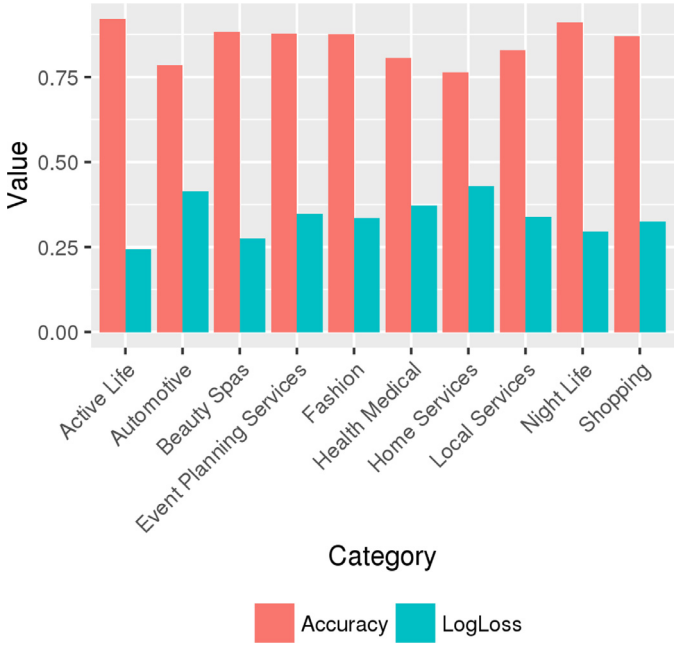

Fig. 13. Comparison of model performance against categories other than *restaurants*.

Unlike memory-based recommendation systems, this approach only requires model parameters to be saved in memory, thus, significantly lowering memory footprint and improving scalability. Scalability and accurate prediction results makes this approach a very attractive proposition to deploy in production systems. This approach can also be implemented to support real-time recommendations, improving overall user experience. We also showed this hybrid model can be applied towards a variety of business categories; hence, this methodology can be easily extended towards other domains.

5.2. Limitations

This research did not use geospatial data of the businesses but it can play an important role in the development of location aware recommendation systems. Geospatial data adds another dimension to the problem and it can be extremely useful in developing a user-specific, real-time recommendation system by leveraging geospatial data from a users' GPS enabled device. Review texts can also play an important role in understanding users' interests and can be more useful compared to star rating. Applying *Natural Language Processing* methods, we can identify latent factors within the review text to further enhance the quality of recommendations.

5.3. Future work

During the last few years, there has been significant technological developments in the domain of distributed storage and computation using big data analytic frameworks like Hadoop, MapReduce and Spark. Most of the recommender systems available today are designed to run on a single server; it is an interesting idea to analyze and compare the cost, performance and efficiency of a recommendation system while utilizing a distributed processing framework.

In future work, we would like to enhance the model training and predictive ability by adding additional content-based features from the Yelp academic dataset, as well as validate the model's performance against other publicly available datasets. Another dimension that we would like to consider in future research is the *social network* effect. Advanced probabilistic graphical modeling methods like Markov networks or Bayesian networks can be used to build the recommendation engine while leveraging the social components available in the Yelp academic dataset.

Finally, it will be interesting to analyze the performance of an ANN using other publicly available datasets such as Amazon reviews, MovieLens and EachMovie databases. These datasets have been extensively studied but analyzing the performance of a general purpose RS using these datasets will be useful.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*(6), 734–749. doi:10.1109/TKDE.2005.99.

Adomavicius, G., & Tuzhilin, A. (2011). In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Context-aware recommender systems* (pp. 217–253). Boston, MA: Springer US. doi:10.1007/978-0-387-85820-3_7.

Amatriain, X. (2013). Big & personal: data and models behind netflix recommendations. In *Proceedings of the 2nd international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications* (pp. 1–6). ACM.

Amini, M., Nasiri, M., & Afzali, M. (2014). Proposing a new hybrid approach in movie recommender system. *International Journal of Computer Science and Information Security, 12*(8), 40–45.

Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66–72. doi:10.1145/245108.245124.

Blattner, M., & Medo, M. (2012). Recommendation systems in the scope of opinion formation: A model. arXiv.org, 1206.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*. In *UAI'98* (pp. 43–52). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370. doi:10.1023/A:1021240730564.

Carrillo, D., López, V. F., & Moreno, M. N. (2013). *Multi-label classification for recommender systems* (pp. 181–188)). Cham: Springer International Publishing. doi:10.1007/978-3-319-00563-8_22.

Champiri, Z. D., Shahamiri, S. R., & Salim, S. S. B. (2015). A systematic review of scholar context-aware recommender systems. *Expert Systems and Applications, 42*(3), 1743–1758. doi:10.1016/j.eswa.2014.09.017.

Chen, L., Chen, G., & Wang, F. (2015). Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction, 25*(2), 99–154. doi:10.1007/s11257-015-9155-5.

Chen, W., Niu, Z., Zhao, X., & Li, Y. (2014). A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web, 17*(2), 271–284. doi:10.1007/s11280-012-0187-z.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Civco, D. L., & Waug, Y. (1994). Classification of multispectral, multitemporal, multisource spatial data using artificial neural networks. *Proceedings of ASPRS/ACSM, 1994*, 123–133.

Cochocki, A., & Unbehauen, R. (1993). *Neural networks for optimization and signal processing* (1st). New York, NY, USA: John Wiley & Sons, Inc.

Curry, H. B. (1944). The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics, 2*(3), 258–261.

DeCarlo, L. T. (1997). On the meaning and use of kurtosis. *Psychological Methods, 2*(3), 292307.

Dooms, S., Pessemier, T., & Martens, L. (2015). Offline optimization for user-specific hybrid recommender systems. *Multimedia Tools and Applications, 74*(9), 3053–3076. doi:10.1007/s11042-013-1768-2.

Duch, W., & Jankowski, N. (1999). Survey of neural transfer functions. *Neural Computing Surveys, 2*, 163–213.

Edmunds, A., & Morris, A. (2000). The problem of information overload in business organisations: A review of the literature. *International Journal of Information Management, 20*(1), 17–28. doi:10.1016/S0268-4012(99)00051-1.

Felfernig, A., Isak, K., Szabo, K., & Zachar, P. (2007). The vita financial services sales support environment. In *Proceedings of the 19th national conference on innovative applications of artificial intelligence - volume 2*. In *IAAI'07* (pp. 1692–1699). AAAI Press.

Ge, X., Liu, J., Qi, Q., & Chen, Z. (2011). A new prediction approach based on linear regression for collaborative filtering. In *2011 eighth international conference on fuzzy systems and knowledge discovery (FSKD): 4* (pp. 2586–2590). doi:10.1109/FSKD.2011.6020007.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation, 4*(1), 1–58. doi:10.1162/neco.1992.4.1.1.

Gunawardana, A., & Meek, C. (2009). A unified approach to building hybrid recommender systems. In *Proceedings of the third acm conference on recommender systems*. In *RecSys '09* (pp. 117–124). New York, NY, USA: ACM. doi:10.1145/1639714.1639735.

Guo, S. (2011). *Bayesian recommender systems: Models and algorithms*. Ph.D. thesis. The Australian National University.

Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., & Zadeh, R. (2013). Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on world wide web*. In *WWW '13* (pp. 505–514). New York, NY, USA: ACM. doi:10.1145/2488388.2488433.

Hariri, N., Mobasher, B., & Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth acm conference on recommender systems*. In *RecSys '12* (pp. 131–138). New York, NY, USA: ACM. doi:10.1145/2365952.2365979.

Hariri, N., Mobasher, B., Burke, R., & Zheng, Y. (2011). Context-aware recommendation based on review mining. In *Ijcai' 11, proceedings of the 9th workshop on intelligent techniques for web personalization and recommender systems* (pp. 30–36).

Hastings, C., Mosteller, F., Tukey, J. W., & Winsor, C. P. (1947). Low moments for small samples: A comparative study of order statistics. *Annals of Mathematical Statistics, 18*(3), 413–426. doi:10.1214/aoms/1177730388.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender systems: An introduction* (1st). New York, NY, USA: Cambridge University Press.

Kaminsky, E. J., Barad, H., & Brown, W. (1997). Textural neural network and version space classifiers for remote sensing. *International Journal of Remote Sensing, 18*(4), 741–762. doi:10.1080/014311697218737.

Kim, S.-M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006). Automatically assessing review helpfulness. In *Proceedings of the 2006 conference on empirical methods in natural language processing*. In *EMNLP '06* (pp. 423–430). Stroudsburg, PA, USA: Association for Computational Linguistics.

Lee, M., Choi, P., & Woo, Y. (2002). A hybrid recommender system combining collaborative filtering with neural network. In *Proceedings of the second international conference on adaptive hypermedia and adaptive web-based systems*. In *AH '02* (pp. 531–534). London, UK: Springer-Verlag.

Li, M., Dias, B., El-Deredy, W., & Lisboa, P. J. G. (2007). A probabilistic model for item-based recommender systems. In *Proceedings of the 2007 acm conference on recommender systems*. In *RecSys '07* (pp. 129–132). New York, NY, USA: ACM. doi:10.1145/1297231.1297253.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80. doi:10.1109/MIC.2003.1167344.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115–133. doi:10.1007/BF02478259.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on artificial intelligence* (pp. 187–192). Menlo Park, CA, USA: American Association for Artificial Intelligence.

Moshfeghi, Y., Piwowarski, B., & Jose, J. M. (2011). Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th international acm sigir conference on research and development in information retrieval*. In *SIGIR '11* (pp. 625–634). New York, NY, USA: ACM. doi:10.1145/2009916.2010001.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. The MIT Press.

Musat, C.-C., Liang, Y., & Faltings, B. (2013). Recommendation using textual opinions. In *Proceedings of the twenty-third international joint conference on artificial intelligence*. In *IJCAI '13* (pp. 2684–2690). AAAI Press.

Nanas, N., Vavalis, M., & Houstis, E. (2010). Personalised news and scientific literature aggregation. *Information Processing & Management, 46*(3), 268–283. doi:10.1016/j.ipm.2009.07.005.

O'Mahony, M. P., & Smyth, B. (2009). Learning to recommend helpful hotel reviews. In *Proceedings of the third ACM conference on recommender systems*. In *RecSys '09* (pp. 305–308). New York, NY, USA: ACM. doi:10.1145/1639714.1639774.

Ozkan, C., & Erbek, F. S. (2003). The comparison of activation functions for multispectral and satellite image classification, *69*(11), 122–234.

Papagelis, M., Rousidis, I., Plexousakis, D., & Theoharopoulos, E. (2005). In M.-S. Hacid, N. V. Murray, Z. W. Raś, & S. Tsumoto (Eds.), *Incremental collaborative filtering for highly-scalable recommendation algorithms* (pp. 553–561)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pero, Š., & Horváth, T. (2013). In S. Carberry, S. Weibelzahl, A. Micarelli, & G. Semeraro (Eds.), *Opinion-driven matrix factorization for rating prediction* (pp. 1–13)). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pham, M. C., Cao, Y., Klamma, R., & Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. *j-jucs, 17*(4), 583–604.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Network, 12*(1), 145–151. doi:10.1016/S0893-6080(98)00116-6.

Raghavan, S., Gunasekar, S., & Ghosh, J. (2012). Review quality aware collaborative filtering. In *Proceedings of the sixth ACM conference on recommender systems*. In *RecSys '12* (pp. 123–130). New York, NY, USA: ACM. doi:10.1145/2365952.2365978.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM, 40*(3), 56–58. doi:10.1145/245108.245121.

Ricci, F., Rokach, L., & Shapira, B. (2011). In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Introduction to recommender systems handbook* (pp. 1–35)). Boston, MA: Springer US.

Rousseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. New York, NY, USA: John Wiley & Sons, Inc.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international acm sigir conference on research and development in information retrieval*. In *SIGIR '02* (pp. 253–260). New York, NY, USA: ACM. doi:10.1145/564376.564421.

Sorensen, S. (2012). Accuracy of similarity measures in recommender systems. *IEEE Intelligent Systems, 22*(3), 4855.

Speier, C., Valacich, J. S., & Vessey, I. (1999). The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences, 30*(2), 337–360. doi:10.1111/j.1540-5915.1999.tb01613.x.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*(1), 1929–1958.

Theano (2015). Deep learning tutorial. [Online; posted 01-September-2015]http://deeplearning.net/tutorial/deeplearning.pdf.

Theano Development Team (2016). Theano: A python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688.

Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology, 49*(11), 1225–1231. http://dx.doi.org/10.1016/S0895-4356(96)00002-9.

Vickery, A., & Vickery, B. C. (2005). Information science in theory and practice. *De Gruyter*.

Wu, M.-L., Chang, C.-H., & Liu, R.-Z. (2014). Integrating content-based filtering with collaborative filtering using co-clustering with augmented matrices. *Expert Systems with Applications, 41*(6), 2754–2761. doi:10.1016/j.eswa.2013.10.008.

Yelp (2015). Yelp academic dataset. https://www.yelp.com/dataset/.

Zhang, K., Narayanan, R., & Choudhary, A. (2010). Voice of the customers: Mining online customer reviews for product feature-based ranking. In *Proceedings of the 3rd wonference on online social networks*. In *WOSN'10*. Berkeley, CA, USA: USENIX Association. 11–11

Zhang, T., & Iyengar, V. S. (2002). Recommender systems using linear classifiers. *Journal of Machine Learning Research, 2*, 313–334. doi:10.1162/153244302760200641.

Zheng, Y. (2014). Semi-supervised context-aware matrix factorization: Using contexts in a way of "latent" factors. In *Proceedings of the 29th annual ACM symposium on applied computing*. In *SAC '14* (pp. 292–293). New York, NY, USA: ACM. doi:10.1145/2554850.2555195.