dTrust: a deep learning approach for social recommendation

Quang-Vinh Dang, Claudia-Lavinia Ignat

dTrust: a deep learning approach for social recommendation

Quang-Vinh Dang
Université de Lorraine, LORIA, F-54506
Inria, F-54600
CNRS, LORIA, F-54506
quang-vinh.dang@inria.fr

Claudia-Lavinia Ignat
Inria, F-54600
Université de Lorraine, LORIA, F-54506
CNRS, LORIA, F-54506
claudia.ignat@inria.fr

***Abstract*—Recommender systems play an important role in modern e-commerce systems. Rating prediction is an important task for recommender systems. Recent studies in social recommendation enhance the performance of rating predictors by taking advantage of user relationship. However, these approaches mostly rely on user personal information to make a prediction. Due to privacy concerns, we should avoid using user personal information.**

In this paper, we present a rating prediction approach relying on deep learning. The approach is easy to implement and does not reveal any personal information. Experiments on real-world data sets showed that the approach outperforms state-of-the-art in both warm-start and cold-start problems.

***Index Terms*—social recommendation, social network analysis and mining, deep learning, trust, e-commerce**

I. Introduction

Modern e-commerce systems offer a huge number of items [1] which is known as the *information overload* problem [2]. A typical user does lack the sufficient personal experience to evaluate the alternative items [3]. Recommender systems, defined as "software tools and techniques that provide suggestions for items that are most likely of interest to a particular user" [3], play an important role in modern e-commerce systems [4].

In this paper, we focus on e-commerce systems that integrated rating system, i.e. a user can leave a rating score to an item. We consider the five-star rating system [5] in which the rating scores are represented as number of stars and range from 1 (lowest) to 5 (highest). The rating score represents user preference on a particular item [5]. In order to provide suggestions for items to users, we seek to predict the future rating score that a user will give to an item. The task is called *item-rating prediction*, or rating prediction.

A huge number of research studies in rating prediction were presented. The existing solutions can be roughly divided into two categories: traditional recommender systems which do not take social relations of users into consideration, and social recommender systems which utilize the social information to improve accuracy of predicting values [6].

Traditional recommender systems are mostly based on user-item rating matrix [7]. In a user-item matrix A, the value of a cell A_{ij} represents the rating score the user i gave to the item j. Traditional recommender systems could be divided further into three sub-categories: "content-based recommender systems", "collaborative filtering based recommender systems" and hybrid methods [6], [8]. Between three popular method, collaborative filtering based recommender systems dominated research in recommender systems for a long period of time [5]. However, a critical problem of traditional recommender systems is that they usually fail with cold-start problem, i.e. when they need to deal with new users or new items in the system [6], [9]. The main reason is the sparsity of the dataset [10]: the number of items that a user can consume is typically very small in comparison with the total number of items that the system can offer [9]. Therefore, when a new user or a new item is introduced to the system, the recommender does not have enough information for prediction.

Traditional recommender systems do not take relationship between users into consideration. Studies [11], [8], [9] suggested that, the user opinions are influence by not only their own preferences but also their trusted friends. Recommender systems that use social information to improve the predicting quality [12] are called "social recommenders" [6]. Social recommendation attracted a lot of attention in recent studies [6], [9], [13]. Several e-commerce systems tried to leverage the social information of users to improve the quality of their recommender systems [8], [14].

In social recommendation, along the user-item rating information, we are given user relation information. Users declare their *trust* and *distrust* opinions on other users. Together the users form a *trust* network. We can integrate the trust network of users and the rating scores given by users to items into a single network as visualized in Figure 1. We called this network as *trust-user-item* network. The task of a rating predictor is to predict the rating score Carol will give to the Item 2.

In fact, negative links are available in some particular networks such as Epinions or Slashdot [15], [16] but in general they do not exist in most of popular social networks [17]. Furthermore, even if these negative links are available, they are usually hidden from public [18] because the negative links might discourage users to involve to the network. Furthermore many users do not want to reveal their negative opinions on others. For the sake of generality, we consider only positive links in this study. The elimination of negative links is con-

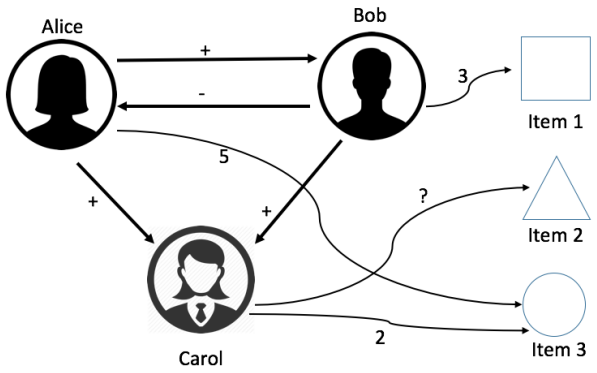


Fig. 1: A trust-user-item network.

sistent with other studies [9], [13].

In this paper we propose a rating prediction solution in trust-user-item network that satisfies the following requirements:

- The solution should scale well for large networks. In fact, several existing solutions that perform very well at small-scale feature high computational time when used in large networks [19].
- The solution should perform well in cold-start problem.
- The solution should be simple enough to be implemented in real-world systems. Simplicity is indeed an important requirement. For instance, the winning solution of Netflix competition was not implemented into Netflix system due to its complexity [20].
- The solution should not use personal information. Several studies [16], [17] claimed that predictions can be made easier if we can access user personal information such as trading history, gender, income and location. However, due to raising concerns about privacy, this information should not be used [9].
- Moreover, the solution should not use detailed information of the products bought by users. In many scenarios, users do not want to publicly reveal the products they are interested in.

Many existing social recommendation techniques rely on a predefined model. These models are defined based on the researchers experience and require an intensive manual engineering. Moreover, these approaches usually require a time consuming preprocessing phase such as computation of global trust [21], execution of a Principal Component Analysis (PCA) [22] or finding all the paths between two nodes to calculate their similarity [23]. These calculations might be feasible for a small network graph, but not for a large graph dataset [19].

In this paper, we present a rating prediction approach for trust networks that relies on deep learning [24]. Our solution is inspired by universal approximation theorem [25] which stated that a multi-layer neural network can approximate any function at any precision level. It relies on a multi-layer neural network for predicting the rating score. However, rather than mimicking a specific existing function, the neural network creates its own function.

In our solution the input data provided to the multi-layer neural network is extracted from the trust network such as the one illustrated in Figure 1 and consists of the relationships between users and items and of the trust relationships between users associated with the time the links were established. In order to preserve user privacy, each user and product is assigned an unique and random ID and the input data set contains uniquely relationships between these IDs. Therefore, it is impossible to know which item is recommended to which user.

The advantages of our solution are:

- No preprocessing step is required as the raw input data is directly fed into the neural network model.
- The solution is simple and easy to be implemented.
- The solution does not reveal any personal information of users or details about their products.
- On real-world and popular datasets, our solution outperforms other state-of-the-art approaches in both warm-start and cold-start predictions.

The paper is organized as follows. We describe existing techniques for item-rating prediction problem in Section II. We present the justification to select deep learning as our approach and detailed description of our model in Section III. The experiments performed are presented in Section V. Section VI discusses the results of our experiments. Conclusions and remarks are drawn in Section VII.

II. Related Work

As briefly mentioned in the previous section, recommender systems could be divided into two groups: traditional recommender systems, which do not consider the relationship between users, and social recommender systems, which take the relationship between users into consideration [6].

Traditional recommender systems include content based recommenders, collaborative filtering based recommenders and hybrid approaches. Content based recommender systems rely on the idea that users tend to like those items which are similar to the items they liked in the past. In collaborative filtering based recommender systems, the score of a particular item I rated by a particular user U is predicted as the weighted average of similar users (user-oriented), the weighted average rating of this user U on similar items (item-oriented), or based on patterns computed with machine learning techniques. The hybrid approach is a combination of content and collaborative filtering approaches, by adding content based features to collaborative filtering model, or vice versa [6].

Collaborative filtering (CF) is the most popular recommender system technique today [26], [27], but it usually fails with cold-start problem. There are several research studies to improve the performance of CF in cold-start problem, such as combining multiple CF techniques [28] or perform CF on a small cluster rather than the entire graph [29].

In opposition to traditional recommender systems, social recommender systems take into account user social relationships. Social relationships are defined as trust relations, friendships, memberships or following relations [30]. As suggested

by Guha et al. [15], if Alice *distrusts* Bob, Alice could deny all the judgments made by Bob, even if these two users are personally similar. It is well known in social recommendation research that the decisions of users are influenced by their social status and relationship [6], [11], [31].

One of the first approaches in social recommendation research is to replace "similar" users in traditional recommendation techniques by "related" users. This means that prediction of rating scores a particular user gave to an item is not based on other similar users, but on the connected friends of this user [18]. TidalTrust [32] and MoleTrust [33] are based on this mechanism.

Several research works claimed that, even users could trust other similar users on some topics, but not in a general context. For instance, Alice could trust Bob in selecting ingredients as Bob is a cooking expert, but Alice may not trust Bob in setting up a new computer system [34]. Tang et al. [34] designed a multi-faceted trust with latent factor model [35] where two users have different trust relationships on different domains that was used to predict trust relationships and item-rating in trust-based recommender systems such as Epinions and Ciao. Hwang et al. [36] used the definition of *category expert* to predict item rating scores in recommender systems.

Many traditional approaches in recommender system research assume a static social relationship between users as well as static user preferences, i.e. if a user liked an item, she will like this item forever [15]. However, trust relationships may change over a long period of time. For instance, users may change a lot the preferences they had ten years ago.

Tang et al. [37] used neural networks to analyze sentiments on reviews users gave on products and then used sentiment scores to predict future user ratings. [38] presented an approach that takes into account both user's preference and sentiment to predict the rating score. However, the approach of [38] on sentiment analysis is based on dictionary rather than a learning algorithm.

Kim and Phalak [39] claimed that, a Web of Trust such as Epinions, where users explicitly express their trust relations on other users, is not always available. They suggested that the review ratings should be taken into consideration in studying trust relationship between users.

Wang et al. [4] proposed to distinguish between *strong* and *weak* ties. The authors integrated the strength of links into Social Matrix Factorization (SMF) technique [40]. The updated model called PTPMF is presented in [9] wherein the authors considered the personalized preference over the tie strength.

Mei et al. [41] introduced a concept of *activeness* besides the trustworthiness as two sources of influence. The authors argued that, if Alice has more influence in the network than Bob, then a trust relation from Carol to Alice should be assigned a higher weighted factor than a trust relation from Carol to Bob. In other words a decision of Alice could have more influence to Carol than a decision of Bob. The authors presented a model that integrated the trustee-influence called MF-NTPR-I^t.

As discussed, manual feature engineering is critical as it is mostly based on researchers experience. Deng et al. [10] proposed using deep autoencoders for automatic feature selection. As opposed to this approach that used deep learning for feature selection, our proposed approach uses deep learning for the entire prediction process.

Negative links, such as distrust relationship between users, can play an important role in item-rating prediction [42]. However, negative links are not popular on the modern online social networks [17] such as Twitter, Instagram or Facebook that do not include the *non-friend relationship*. In this paper, we focus our study on online social networks with positive links.

Similar to our work, Covington et al. [43] also used multi-layer feed-forward neural networks for recommendation service in YouTube. However, as opposed to this approach that requires data preprocessing such as vector embedding, our solution simply takes the raw data as input without the need of a preprocessing step.

One of the major issues of recommender systems is the cold-start problem, i.e. dealing with new users or new items. Most of traditional recommender systems rely on the knowledge of existing users or items and fail in cold start problem [6]. Our experiments show that the approach we present can deal with this problem.

III. Learning model

A. Problem Definition

We define the problem as follows. Given a list of users U, a list of items I, a user-item rating matrix $R_{|U|\times|I|}$, a user-user trust relation matrix $T_{|U|\times|U|}$.

The task is to predict missing rating scores in the matrix R.

B. Motivation

As we described in Section II, several approaches for item-rating prediction using trust information have been presented in recent years. The input of the existing algorithms are the user-item rating matrix and user-user relation matrix. Hence, we can notate all the existing algorithms as a function $\hat{f}$ as follows:

$$\hat{r}_{i,j} = \hat{f}(R, T, i, j) \tag{1}$$

wherein $\hat{r}_{i,j}$ is the predicting rating score from user U_i to the item I_j. The definition of the function $\hat{f}$ depends on particular approach. We notate the *true* rating score from U_i to I_j as r_{ij}.

Our approach is inspired by the universal approximation theorem [25]. Originally the theorem stated that a feed-forward neural network can approximate any continuous function by using a bounded activation function. Recent studies proved that the universal approximation property of a multi-layer feed-forward neural network holds for unbounded activation function such as the *rectifier* function we used in this paper [44]. Studies also proved that a neural network can approximate a discontinuous function [45], and in practice the distinction between continuous and discontinuous functions is usually not

important for the approximation capacity of a neural network [46]. The approximation theorem states that, for any $\epsilon > 0$, we are able to construct a multi-layer feed-forward neural network F so that:

$$|F(R,T,i,j) - \hat{f}(R,T,i,j)| < \epsilon \forall R,T,i,j \tag{2}$$

Therefore, we argue that, if the function $\hat{f}$ can be defined manually in previous studies for rating prediction, we can also build a multi-layer neural network to do the same task. Our approach will be more simple in term of development cost. In fact, we do no try to approximate a particular existing function, but let the neural network to build its own function. As we will see in the following sections, the multi-layer neural network can predict the rating score better than state-of-the-art algorithms.

Furthermore, theoretically, if the function f exists, i.e. if there is a perfect predictor that predicts correctly all the time the rating score, we know that there exists also a multi-layer feed-forward neural network that can approximate this f at an arbitrary precision level.

C. Model Selection

As we discussed above, our approach is inspired by the universal approximation theorem. Furthermore, we chose deep learning approach for some other reasons due to the properties of the problem. In this section we discuss our selection of model.

Studies addressed several important properties of trust-user-item networks:

- **Large-scale**: Modern trust-user-item networks contain millions or billions of users and products [17].
- **Sparsity**: The trust-user-item networks are known as very sparse graphs, i.e. the number of established links is very small compared to the number of possible links [16]. Indeed, these networks usually contain millions or billions of nodes while a typical user can establish several hundreds of links.
- **Imbalance**: The social network data is also known as imbalanced data, as users generally establish more positive reviews than negative reviews [6], i.e. the number of rating scores 4 or 5 is much higher than the number of rating scores 1 or 2.

In order to address the above challenges, we chose deep learning as our approach for the item-rating prediction problem, due to several theoretical advantages [47]:

- Deep learning can learn complex nonlinear relationship between input and output data.
- Deep learning can learn without or with manual feature engineering, which is considered as one of the most difficult tasks in machine learning [48].
- Deep learning can scale well with a huge training set.
- Deep learning can learn well with unbalanced datasets and incorrect labelled data.

D. Deep Neural Network Model

In this paper, we focus on *deep neural network* (DNN) for the rating prediction problem. An artificial neural network includes at least one input and one output layer, and optionally one or many hidden layers where each layer includes one or many neurons. An artificial neural network with multiple hidden layers is called *deep neural network*, but so far there is not yet a standard definition of the minimum required number of hidden layers for an artificial neural network to be considered *deep* [49]. We follow [47], [50] by considering neural networks with at least two hidden layers as being deep learning models.

A visualization of a DNN is displayed in Figure 2. The layer k computes an output vector h^{k} using the output $h^{\mathrm{k\text{-}1}}$ of the previous layer, starting with the input layer $x = h^0$ as in Equation 3, where b^{k} is the offset vector and W^{k} is the matrix of weights.

$$h^{\mathrm{k}} = f(b^{\mathrm{k}} + W^{\mathrm{k}} h^{\mathrm{k\text{-}1}}) \tag{3}$$

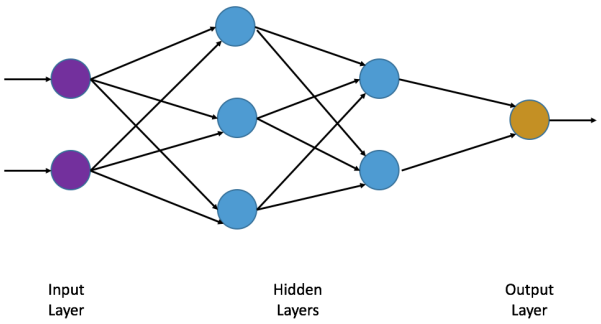


Fig. 2: A multi-layer feed-forward neural network.

The function f in Equation 3 is called *activation function*, which decides how each neuron calculates and transfers the signal to the neurons in subsequent layer. The advantage of deep neural networks is that, the calculation in deep neural networks usually requires simple mathematical activation functions. A popular activation function is called *rectifier*, which is the most simple non-linear function. Rectifier function is defined as:

$$rectifier(x) = max(0, x) \tag{4}$$

Even if the *rectifier* function is very simple, its prediction quality is very good [51], [43]. Due to its simplicity, the *rectifier* activation function has the advantage that it can be faster computed compared to other activation functions such as *tanh* or *sigmoid*. The *rectifier* function is the mostly used activation function today [52].

As we observe in Equation 3, training and applying deep neural networks could be considered as a series of matrix calculation. These operations can be calculated efficiently using parallel computing techniques, such as Hadoop & MapReduce [53] or GPU-computing technology like CUDA [54].

Originally, neural networks are designed to process numerical input and output data. Indeed, the input and output of activation functions are numerical values. However, in item-rating prediction, users and items are categorical values. In order to apply deep neural networks in item-rating prediction, we need to convert categorical values to numerical values by using "one-hot encoding". The idea of "one-hot encoding" is to transfer categorical values into a matrix, where each row contains a single element with value 1, all other elements having the value 0. For instance, the one-hot encoding for the three items iPhone, iPad and iPod corresponds to the three vectors [1,0,0], [0,1,0] and [0,0,1] respectively.

The issue of one-hot encoding is that it increases the number of data dimension dramatically, which leads to "curse of dimensionality". Practically, many supervised algorithms fail in high dimensional space. In order to reduce the input size and allow the neural network to be trained effectively, we used feature hashing technique [55].

We note that, in this study we do not feed the exact numeric values of user ID or item ID into the model, which usually should be avoided in machine learning. We instead feed to the model the encoded values of these IDs, avoiding to reveal private information about users and items.

Selecting the number of hidden layers for DNN models is a very difficult task [43]. Learning complex data structure usually requires multiple hidden layers, but when the models become deeper, "it becomes more difficult to obtain good generalization" [56]. In fact, this problem is still an open theoretical problem [57]. In practice, as Professor Yoshua Bengio at Université de Montreal[1] stated, the best way of choosing the number of hidden layer in a DNN model is "just keep adding layers until the test error does not improve anymore"[2]. In this paper, we follow the approach suggested by Professor Bengio.

IV. Data Preparation

In this section we describe how the graph information such as the one illustrated in Figure 1, but where only positive links are considered, is fed into a DNN model.

A trust-user-item network is created over time, when links are added one by one. We can organize the network as in Table I that contains the rating information over time of users on products and in Table II that contains the positive link information between users over time.

User	Item	Time	Rating Score
Bob	Item 1	2010	3
Carol	Item 3	2012	2
Alice	Item 3	2016	5

TABLE I: User-item table.

To feed the input data into the DNN model, we combine Table I and Table II into one single table as visualized in Table III. In this combined table, we consider positive links between users having the value of 5, the highest possible rating score.

Trustor	Trustee	Time
Bob	Alice	2011
Alice	Carol	2013
Alice	Bob	2015

TABLE II: User-user table.

Link Source	Link Destination	Time	Link Value
Bob	Item 1	2010	3
Carol	Item 3	2012	2
Alice	Item 3	2016	5
Bob	Alice	2011	5
Alice	Carol	2013	5
Alice	Bob	2015	5

TABLE III: User-user-item table.

The final model for rating prediction is illustrated in Figure 3. Note that the figure displays only few connections between layers, but our experiments use the fully connected DNN.

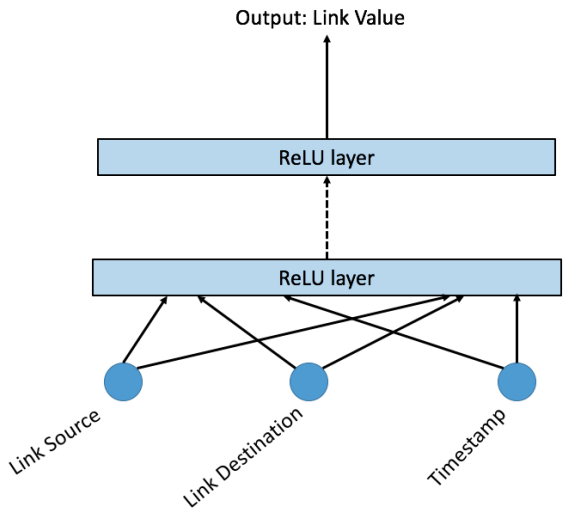


Fig. 3: Rating prediction with DNN. Despite the visualization, fully-connected DNNs are used. The link sources and destinations are fed as one-hot encoding vectors using hash function for dimension reduction [55]. ReLU stands for Rectified Linear Unit.

Feeding raw data directly into a fully connected DNN was also used widely and successfully in different tasks such as classification in computer vision or graph-based data [57], [50]. The advantage of a DNN is that the training time is much faster than other complex algorithms such as ConvNet which is used in other graph analysis tasks [50].

V. Experiments

A. Dataset

In order to evaluate our proposed solution we used three real-world datasets collected from two review websites which

[1]http://www.iro.umontreal.ca/~bengioy/

[2]https://www.quora.com/Artificial-Neural-Networks-How-can-I-estimate-the-number-of-neurons-and-layers/answer/Yoshua-Bengio?srid=hqJd

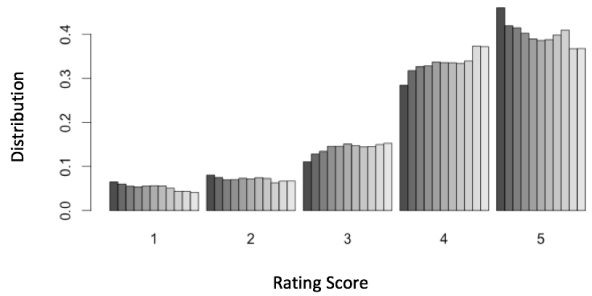


Fig. 4: Distribution of rating scores over 11 years in Epinions2 dataset. Each column represents one-year data.

are Epinions[3] and Ciao[4] where users can rate items and also can declare trust on other users.

- **Epinions1**: The dataset is collected and presented in [33]. The dataset is available at http://www.trustlet.org/.
- **Epinions2 & Ciao**: The two datasets are collected and presented in [34]. The datasets are available at http://www.jiliang.xyz/trust.html.

All these datasets are provided in a same format discussed in Section IV. Each dataset is composed of two files: one containing users rating score on items and one containing trust relations between users.

The distributions of rating score in the three datasets are displayed in Figure 5. We could see that all datasets are imbalanced: most of established rating scores are 4 or 5. We also observed that the distribution of rating score in the three datasets is similar even if data are collected at different times and from different websites.

We also displayed the distribution of number of trustors that trust a trustee and number of trustees that are trusted by a trustor in Figure 6. Most of the time, a trustor trusts four trustees, and similarly a trustee is trusted by four trustors. Similar distributions are obtained throughout the three datasets.

The datasets were established over a long time period. However, the independent and identical distribution (i.i.d.) assumption holds, i.e. distribution of rating scores does not change during the period. This means that the training and testing datasets are pulled out from a same distribution. As an example, we displayed the distribution of rating scores in Epinions2 dataset over 11 years in Figure 4. We can see that the distribution does not change over time. Similar distributions are found in the other two datasets.

B. Implementation

We implemented our solution in R language[5]. We used *h2o* package[6] for building and training deep neural networks. *h2o* package performs deep neural network training on Hadoop and MapReduce to utilize the availability of parallel programming with neural networks[7].

[3]http://epinions.com/
[4]http://ciao.com/
[5]https://www.r-project.org/
[6]http://www.h2o.ai/

	Epinions1	Epinions2	Ciao
# of Users	49,290	22,166	12,375
# of Products	139,738	296,277	106,797
# of Rating	664,824	922,267	284,086
# of Links	487,181	355,813	237,350
First Rating on	1999	1999	2000
Last Rating on	2003	2011	2011
Mean Rating	3.99	4.05	4.21

TABLE IV: Statistics of datasets

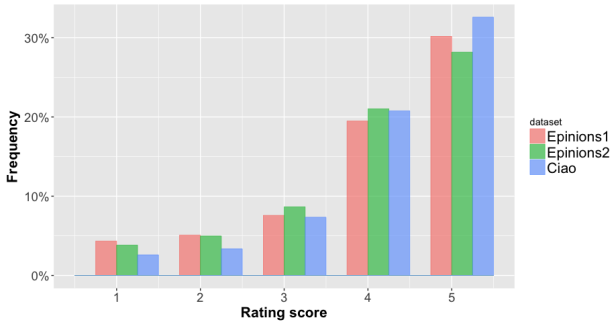


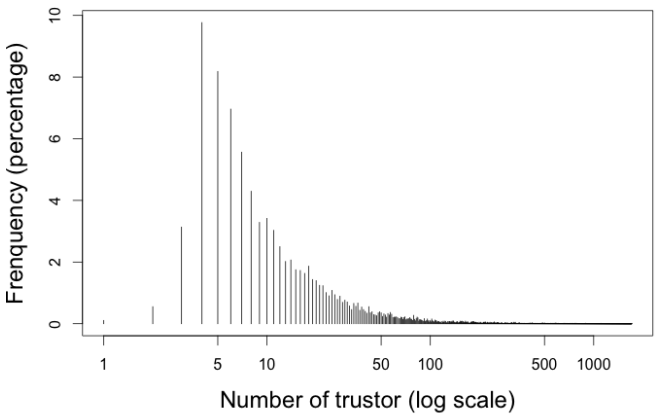
Fig. 5: Distribution of Rating Score in three datasets.

We studied the performance of deep neural networks in three predicting problems: general item-rating prediction, *warm-start* and *cold-start* problems. We used the same definition of *cold-start* problem as in [16]: *cold-start* problem assumes that items or users appear on the testing set, but not in the training set. The definition of *warm-start* is simply the reversion of *cold-start* problem.

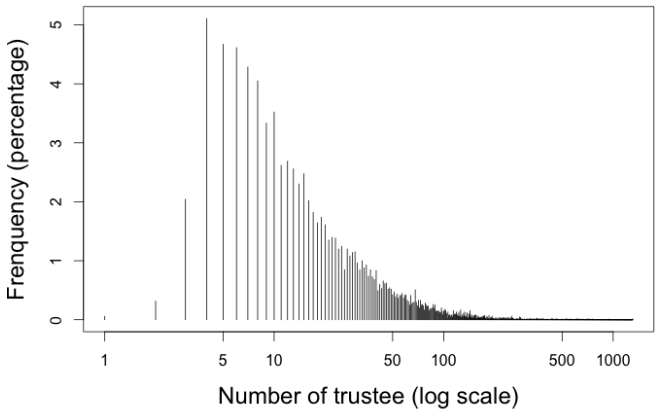
In order to make a fair comparison, we performed two different experiments, using two different training/testing division strategies. In the first experiment, we used the same training/testing division with previous studies, but the rating scores and trust relations in training set are established before the scores and relations in testing set, i.e. we used the historical data to predict the future. Similar to [58], in the second experiment, we used leave-one-out cross validation. We compared the performance of our DNN models with the performance metrics presented in corresponding papers.

As suggested by Professor Bengio, we performed the prediction on item-rating by using different neural network models with an increasing number of hidden layers. The number of neurons in each layer is optimized by using *grid search*. However, we found that a small change on number of neurons does not have a big effect on the performance of the deep neural network models. We used the *pyramid* architecture [24], meaning that the number of neurons is reduced in deeper layers, similar with other previous studies [43]: we started with 2048 neurons in the first layer, then reduced the number of neurons by half in next layer, i.e. we used 1024 neurons

[7]The code and data to reproduce our results are available at http://bit.ly/2nQWviH



(a) Distribution of number of trustor on trustee



(b) Distribution of number of trustee on trustor

Fig. 6: Distribution of number of trustors and trustees

in the second layer, then 512 neurons in the third layer if this layer was included and so on.

C. Metrics

We use two popular metrics to evaluate the performance of our solution, which are *Root Mean Square Error* (RMSE) and *Mean Absolute Error* (MAE). RMSE and MAE are considered as suitable metrics for evaluating predicting task of a recommender system [59].

Both two metrics are defined on two vectors, one is $A = A_1, A_2, ..., A_n$ representing the *actual* values, and the other one is $P = P_1, P_2, ..., P_n$ representing the *predicting* values of a predictor on n items.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(A_i - P_i)^2}{n}} \tag{5}$$

$$MAE = \frac{\sum_{i=1}^{n}|A_i - P_i|}{n} \tag{6}$$

RMSE punishes the large error more than MAE, therefore RMSE is more suitable for measuring the performance of recommender systems [31]. However, we present both metrics for our results for a comparison with existing studies.

D. Data Division

Two main approaches exist for dividing datasets into training and testing sets.

The first approach can be called as "sequential division", meaning that we train the model using historical data and predict on future data [34], [16]. In sequence, the testing data can be divided further into two parts that are the cold-start part which considers users or items that appear only in testing data but not in training data, and warm-start part that is simply the reversion of cold-start data.

The second division approach is a popular k-fold cross-validation method. In this method, we divided the training dataset (just the user-item table) into equal k parts. Then the training and testing steps are repeated k times. For each time, we take a different part as the testing data and the remaining $k-1$ parts as the training data. The method is applied in [58], [31].

E. Baseline Models

In this study, we compare our solution with several state-of-the-art studies displayed in Table V.

In order to avoid re-implementation of existing solutions in a non optimal way, we refer to the scores reported in the corresponding publications. We set up experiments with the same settings as reported in the literature and we compared our solution with published results using the same metrics.

Solution	Year	Metric	Datasets
mTrust [34]	2012	RMSE	Epinions2, Ciao
DynFluid [58]	2015	RMSE	Epinions2, Ciao
eTrustRec [60]	2015	RMSE	Epinions2
DLMF [10]	2016	RMSE	Epinions1
Davoudi [7]	2016	RMSE, MAE	Epinions2
Costa [61]	2016	RMSE	Epinions2, Ciao
TrustSVD [31]	2016	RMSE, MAE	Epinions1, Ciao
PTPMF [9]	2017	RMSE, MAE	Epinions1
MF-NTPR-I^t [41]	2017	RMSE	Epinions2

TABLE V: Baseline models used for performance comparison.

Furthermore, we compare our solution with two popular baseline models:

- **Mean**: the rating of a product is predicted as the mean of known previous ratings of this product. For new items we simply predict the rating score as average rating score of all other products.
- **NN**: the nearest neighbor based method which predicts rating score of a product as the average rating score of their trusted friends.

F. Results

In this section we present the experimental results and comparison with baseline models. RMSE and MAE values for different experiments and the comparison with corresponding state-of-the-art models are presented in Table VI, VII, VIII and IX.

We displayed the distribution of predicting error of dTrust in all experiments with different hidden layers in Figure 7. We can see that, dTrust with 2 layers provides a *sharper* prediction than dTrust with 1 or 3 layers. It means that, while dTrust with 1 hidden layer is too shallow to capture the relationship between users and items, dTrust with 3 hidden layers on the other hand is too deep and the last layer does not contribute much for the prediction [57].

Algorithm	Warm-start	Cold-start	All
Mean	1.1054	1.1562	1.1106
NN	1.1092	1.1566	1.1148
dTrust (1)	1.0378	1.0876	1.076
mTrust	1.0566	1.1375	1.0646
eTrustRec2	1.0228	1.0672	1.0272
dTrust (2)	1.0033	1.067	1.022
dTrust (3)	1.005	1.067	1.023
dTrust (4)	1.049	1.13	1.076

TABLE VI: RMSE of different predictors in the first experiment (using historical data to predict future) on Epinions2 dataset. The values inside parentheses represent the number of hidden layers we used with *dTrust*.

Algorithm	RMSE
Mean	1.17
NN	1.09
MF-NTPR-I^t	1.079
dTrust (1)	1.066
DynFluid	1.05
dTrust (2)	1.028
dTrust (3)	1.03
dTrust (4)	1.092

TABLE VII: RMSE in the second experiment (cross-validation) on Epinions2 dataset.

Algorithm	Epinions1	Ciao
Mean	1.21	1.06
NN	1.17	1.01
mTrust	-	0.97
DynFluid	-	0.97
DLMF	1.07	-
[61]	1.16	0.94
TrustSVD	1.043	0.956
dTrust (1)	1.045	0.95
dTrust (2)	1.039	0.93
dTrust (3)	1.049	0.94
dTrust (4)	1.06	0.96

TABLE VIII: RMSE on Epinions1 and Ciao datasets in the first experiment (using historical data to predict future).

VI. Discussion

Experimental results showed that our deep learning solution outperforms other existing methods in rating prediction for both *warm-start* and *cold-start* problems.

Algorithm	Epinions1	Epinions2	Ciao
Mean	0.92	0.89	0.82
NN	0.89	0.88	0.8
[7]	-	0.87	-
TrustSVD	0.8	-	0.72
PTPMF	0.82	-	-
dTrust (1)	0.82	0.86	0.73
dTrust (2)	0.8	0.87	0.72
dTrust (3)	0.85	0.87	0.73
dTrust (4)	0.89	0.93	0.76

TABLE IX: MAE of different predictors.

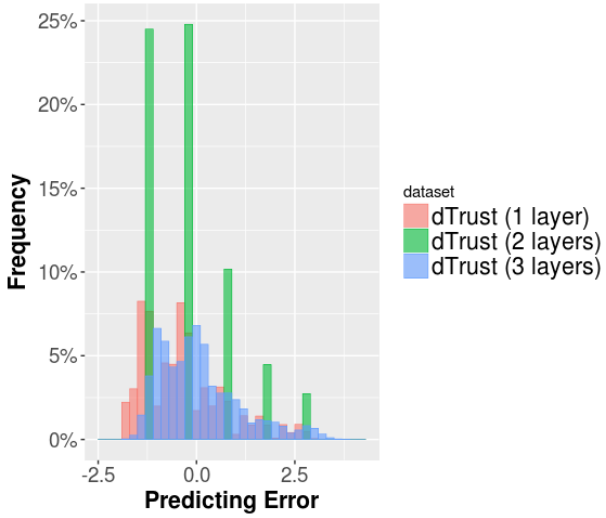


Fig. 7: Distribution of predicting errors

As we could expect, all approaches performed best in *warm-start* problem. The performance is reduced dramatically if we switch the predicting task to the *cold-start* problem. The RMSE values when we predict the entire testing dataset are between the RMSE values of *cold-start* and *warm-start* problems. This is explained by the fact that the entire testing set is a combination of *warm-start* and *cold-start* testing sets.

We performed the predicting tasks by different neural network models with increasing number of hidden layers as suggested in practice and literature [43]. We found that the neural networks with two hidden layers perform best in comparison with other models for the rating prediction problem that we studied. The result is expected as DNN models become unstable when more hidden layers are added [56], leading to the consequence that a DNN model cannot be extended forever [57]. We recall that, a neural network with two hidden layers is considered as a deep neural network by many research studies [56], [57], [50].

In order to verify the improvement in predicting item-rating scores between our deep learning approach and existing methods, we performed *t-tests* on the difference of rating scores between our best method, i.e. a neural network with two hidden layers, and the second best method in each experiment. The fact that p-values of all *t-tests* are less than 0.05 allows us to confirm the significant improvement of our approach

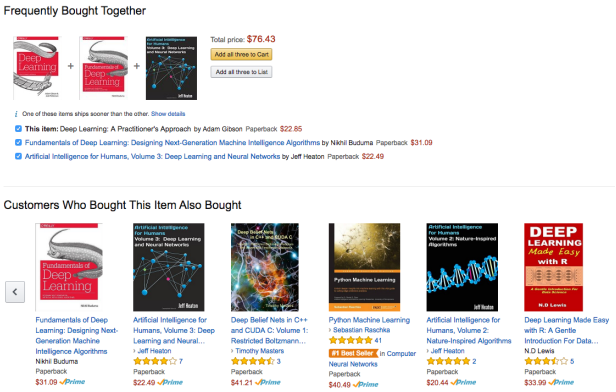


Fig. 8: A screenshot of Amazon's recommender system. Only few products are recommended to users.

compared to other methods.

Due to the huge size of the testing set in all experiments, improvements in term of RMSE and MAE are of a small order. This is however an improvement over the state-of-the-art. We recall that, in Netflix competition, the difference between the first and second best approaches was only 0.18%. As claimed in [16], in rating prediction, small absolute improvements in RMSE can lead to a significant impact on quality of the top few recommendations. For example, [35] showed that, the improvement of 0.0155 on absolute value of RMSE led to 50% relative improvement of top few recommendations. Improvement in top recommendations is very important as recommender systems in modern e-commerce only provide top few recommendations to users as shown in Figure 8.

Besides better predicting accuracies than state-of-the-art approaches, dTrust has also the advantage that the dataset is fed directly into the model without feature engineering. Generally speaking, feature engineering is a very difficult task in machine learning and requires a lot of human effort and expertise. Avoiding this step also means that we can save the preprocessing time.

VII. Conclusion & Future Works

In the book "The Paradox of Choice: Why More is Less" [62], Barry Schwartz claimed that increasing the selling offer in terms of the number of products does not make buyers happier as they do not know what to choose. Recommender systems tackle this problem by suggesting customers few items among billions of them.

Item rating prediction is definitely an important problem in recommender system research field. In this paper, we aimed improving the quality of rating prediction by combining user relations with rating historical data using a deep feed-forward neural network. Experiments showed the improvement of our method compared to other existing methods in term of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). Our experimental results suggest that given enough training data, deep neural network models could outperform other solutions based on complex mathematical models.

Our research study shows the potential of using deep learning in studying users' behavior in trust-based online social networks. Deep learning could also avoid manual feature engineering, which is currently a standard step required in machine learning. The future research plan includes optimizing the proposed neural network model and extending it to capture other types of users' behavior.

References

[1] P. Grey, "How many products does amazon sell?" December 2015. [Online]. Available: https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015/

[2] M. M. Azadjalal, P. Moradi, A. Abdollahpouri, and M. Jalili, "A trust-aware recommendation method based on pareto dominance and confidence concepts," *Knowl.-Based Syst.*, vol. 116, pp. 130–143, 2017.

[3] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*, 2015.

[4] X. Wang, W. Lu, M. Ester, C. Wang, and C. Chen, "Social recommendation with strong and weak ties," in *CIKM*. ACM, 2016, pp. 5–14.

[5] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, 2016.

[6] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Netw. Analys. Mining*, 2013.

[7] A. Davoudi and M. Chatterjee, "Modeling trust for rating prediction in recommender systems," in *SIAM MLRec*, 2016.

[8] Y. Li, C. Wu, and C. Lai, "A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship," *Decision Support Systems*, vol. 55, no. 3, pp. 740–752, 2013.

[9] X. Wang, S. C. H. Hoi, M. Ester, J. Bu, and C. Chen, "Learning personalized preference of strong and weak ties for social recommendation," in *WWW*. ACM, 2017, pp. 1601–1610.

[10] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE TNNLS*, 2016.

[11] C. A. Yeung and T. Iwata, "Strength of social influence in trust networks in product review sites," in *WSDM*, 2011.

[12] P. Gao, H. Miao, J. S. Baras, and J. Golbeck, "STAR: semiring trust inference for trust-aware social recommenders," in *RecSys*, 2016.

[13] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1633–1647, 2017.

[14] Y. Dou, H. Yang, and X. Deng, "A survey of collaborative filtering algorithms for social recommender systems," in *SKG*. IEEE Computer Society, 2016, pp. 40–46.

[15] R. V. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *WWW*, 2004.

[16] J. Tang, H. Gao, H. Liu, and A. D. Sarma, "eTrust: understanding trust evolution in an online world," in *KDD*. ACM, 2012, pp. 253–261.

[17] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Comput. Surv.*, 2016.

[18] P. Massa and P. Avesani, "Trust-aware recommender systems," in *RecSys*, 2007.

[19] L. Duan, C. Aggarwal, S. Ma, R. Hu, and J. Huai, "Scaling up link prediction with ensembles," in *WSDM*. ACM, 2016, pp. 367–376.

[20] N. T. Blog, "Netflix recommendations: Beyond the 5 stars," April 2012. [Online]. Available: https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

[21] W.-P. Lee and C.-Y. Ma, "Enhancing collaborative recommendation performance by combining user preference and trust-distrust propagation in social networks," *Knowledge-Based Systems*, 2016.

[22] M. Pozo, R. Chiky, and E. Métais, "Enhancing collaborative filtering using implicit relations in data," *Trans. Comput. Collective Intelligence*, 2016.

[23] W. Zhang, B. Wu, and Y. Liu, "Cluster-level trust prediction based on multi-modal social networks," *Neurocomputing*, 2016.

[24] H. Wang, B. Raj, and E. P. Xing, "On the origin of deep learning," *CoRR*, vol. abs/1702.07800, 2017.

[25] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, 1991.

[26] I. Barjasteh, R. Forsati, D. Ross, A. Esfahanian, and H. Radha, "Cold-start recommendation with provable guarantees: A decoupled approach," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1462–1474, 2016.

[27] J. Liu, Y. Jiang, Z. Li, X. Zhang, and H. Lu, "Domain-sensitive recommendation with user-item subgroup analysis," in *ICDE*, 2016.

[28] E. Q. da Silva, C. G. Camilo-Junior, L. M. L. Pascoal, and T. C. Rosa, "An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering," *Expert Syst. Appl.*, vol. 53, pp. 204–218, 2016.

[29] C.-L. Liao and S.-J. Lee, "A clustering based approach to improving the efficiency of collaborative filtering recommendation," *Elec. Com. Res. and Appl.*, 2016.

[30] I. King, M. R. Lyu, and H. Ma, "Introduction to social recommendation," in *WWW*, 2010.

[31] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *IEEE TKDE*, 2016.

[32] J. Golbeck, "Combining provenance with trust in social networks for semantic web content filtering," in *IPAW*, 2006.

[33] P. Massa and P. Avesani, "Trust metrics on controversial users: Balancing between tyranny of the majority," *IJSWIS*, 2007.

[34] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *WSDM*, 2012.

[35] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD*, 2008.

[36] W. Hwang, H. Lee, S. Kim, Y. Won, and M. Lee, "Efficient recommendation methods using category experts for a large dataset," *Inf. Fusion*, 2016.

[37] D. Tang, B. Qin, T. Liu, and Y. Yang, "User modeling with neural network for review rating prediction," in *IJCAI*, 2015.

[38] X. Ma, X. Lei, G. Zhao, and X. Qian, "Rating prediction by exploring users preference and sentiment," *Multimedia Tools and Applications*, pp. 1–20, 2017.

[39] Y. A. Kim and R. Phalak, "A trust prediction framework in rating-based experience sharing social networks without a web of trust," *Inf. Sci.*, 2012.

[40] M. Jamali and M. Ester, "*TrustWalker*: a random walk model for combining trust-based and item-based recommendation," in *KDD*. ACM, 2009, pp. 397–406.

[41] J. Mei, H. Yu, Z. Shen, and C. Miao, "A social influence based trust model for recommender systems," *Intell. Data Anal.*, vol. 21, no. 2, pp. 263–277, 2017.

[42] R. Forsati, I. Barjasteh, F. Masrour, A. Esfahanian, and H. Radha, "Pushtrust: An efficient recommendation algorithm by leveraging trust and distrust relations," in *RecSys*, 2015.

[43] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *RecSys*, 2016.

[44] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, 2015.

[45] B. Llanas, S. Lantarón, and F. J. Sáinz, "Constructive approximation of discontinuous functions by neural networks," *Neural Processing Letters*, vol. 27, no. 3, pp. 209–226, 2008.

[46] M. A. Nielsen, *Neural networks and deep learning*. Determination Press USA, 2015.

[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[48] A. Ng, "Machine learning and ai via brain simulations," 2013.

[49] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[50] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, 2017.

[51] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*. Omnipress, 2010, pp. 807–814.

[52] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms*. O'Reilly Media, 2017.

[53] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*. USENIX Association, 2004, pp. 137–150.

[54] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *CoRR*, vol. abs/1410.0759, 2014.

[55] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *ICML*, 2009.

[56] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[58] H. Zheng and J. Wu, "Dynfluid: Predicting time-evolving rating in recommendation systems via fluid dynamics," in *TrustCom*, 2015.

[59] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *Journal of Machine Learning Research*, vol. 10, pp. 2935–2962, 2009.

[60] J. Tang, H. Gao, A. D. Sarma, Y. Bi, and H. Liu, "Trust evolution: Modeling and its applications," *IEEE TKDE*, 2015.

[61] G. Costa and R. Ortale, "Model-based collaborative personalized recommendation on signed social rating networks," *Trans. on Int. Tech.*, 2016.

[62] B. Schwartz, "The paradox of choice: Why less is more," *New York: Ecco*, 2004.