dTrust: a simple deep learning approach for social recommendation

Quang-Vinh Dang, Claudia-Lavinia Ignat

dTrust: a simple 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*—Rating prediction is a key task of e-commerce recommendation mechanisms. Recent studies in social recommendation enhance the performance of rating predictors by taking advantage of user relationships. However, these prediction approaches mostly rely on user personal information which is a privacy threat.**

In this paper, we present dTrust, a simple social recommendation approach that avoids using user personal information. It relies uniquely on the topology of an anonymized trust-user-item network that combines user trust relations with user rating scores. This topology is fed into a deep feed-forward neural network. Experiments on real-world data sets showed that dTrust outperforms state-of-the-art in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) scores for 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 buyers a huge number of items [1] and a typical user experiences difficulties in evaluating the alternative items [2]. This is known as the *information overload* problem [3]. Recommender systems, defined as "software tools and techniques that provide suggestions for items that are most likely of interest to a particular user" [2], play an important role in modern e-commerce systems to cope with the information overload problem [4].

In this paper, we focus on e-commerce systems that integrate rating mechanisms where a user can provide a rating score to an item. We consider the five-star rating systems [5] where rating scores range from 1 (lowest) to 5 (highest). A rating score represents the preference of a user on a particular item [5]. We aim predicting future item rating scores provided by users and recommend users only highly rated items. The task is called *item-rating prediction*, or rating prediction.

Based on their underlying rating prediction solutions, recommender systems can be roughly divided into two categories: traditional recommender systems which do not consider user social relations and social recommender systems based on 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 R, the value of a cell r_{ij} represents the rating score the user i gave to the item j. A critical problem of traditional recommender systems is that they usually fail with cold-start problem, i.e. when facing new users or items in the system [6], [8]. The main reason is the sparsity of the dataset [9]: the number of items that a user can consume is typically very small compared to the total number of items [8]. Therefore, in the face of a new user or item, the recommender does not have enough information for prediction.

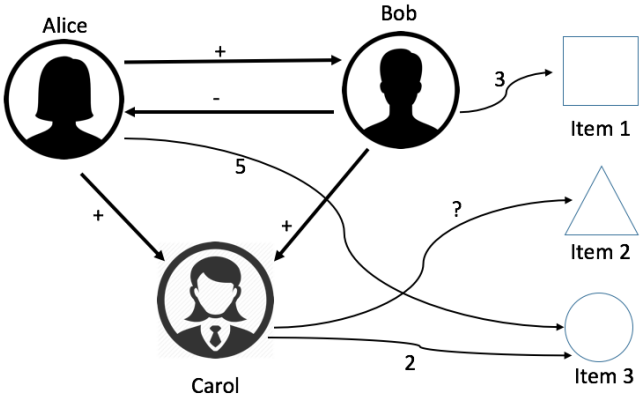


Fig. 1: A trust-user-item network.

Traditional recommender systems do not take relationship between users into consideration. Studies [10], [11], [8] suggested that user opinions are influenced by not only their own preferences but also their trusted friends. Social recommendation attracted a lot of attention in recent studies [6], [8], [12] and several e-commerce systems tried to leverage user social information to improve the quality of their recommender systems [11], [13].

Social recommendation relies on information on user relationship in addition to user-item rating. Users declared *trust* and *distrust* opinions on other users compose a *trust* network. Integration of users trust network with the rating scores given by users to items leads to the *trust-user-item* network. An example of such a *trust-user-item* network is shown in Figure 1 where the task of the rating predictor is to predict the rating score Carol will give to Item 2.

In fact, negative links are available in some particular networks such as Epinions or Slashdot [14], [15] but they are

missing in most popular social networks [16]. Furthermore, even if these negative links are available, they are usually hidden from public [17] as they might discourage users to participate to the network. 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 consistent with other studies [8], [12].

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

- It should scale well for large networks. In fact, several existing solutions that perform very well at small-scale feature a high time complexity when used in large networks [18].
- It should perform well in cold-start problem.
- It 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 high complexity [19].
- It should not use personal information. Several studies [15], [16] 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 [8].
- It should not use information about 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 [20], execution of a Principal Component Analysis (PCA) [21] or finding all the paths between two nodes to calculate their similarity [22]. These calculations might be feasible for a small network graph, but not for a large graph dataset [18].

In this paper, we present a rating prediction approach for trust networks that relies on deep learning [23]. Our solution relies on a multi-layer neural network for predicting the rating score, being inspired by universal approximation theorem [24] which stated that a multi-layer neural network can approximate any function at any precision level. However, rather than mimicking a specific existing function, in our solution 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-user-item network 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:

- It does not require a preprocessing step because the raw input data is directly fed into the neural network model.
- It is simple and easy to be implemented.
- It does not reveal any personal information of users or details about their products.
- It outperforms other state-of-the-art approaches in both warm-start and cold-start predictions on real-world datasets.

The paper is organized as follows. We describe existing techniques for item-rating prediction problem in Section II. We motivate our solution based on deep learning and describe our model in Section III. We present the experiments we performed in Section IV. Section V discusses the results of our experiments. Finally, Section VI presents concluding remarks.

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 can be divided into three sub-categories: content-based, collaborative filtering based and hybrid ones [6], [11]. 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 [25], [26], 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 [27] or performing CF on a small cluster rather than the entire graph [28].

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 [29]. As suggested by Guha et al. [14], 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], [10], [30].

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 [17]. TidalTrust [31] and MoleTrust [32] are based on this mechanism.

Several research works claimed that 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 [33]. Tang et al. [33] designed a multi-faceted trust with latent factor model [34] where two users have different trust relationships on different domains. This multi-faceted trust was used to predict trust relationships and item-rating in trust-based recommender systems such as Epinions and Ciao.

Many traditional approaches in recommender system research assume static user social relationships and preferences [14]. However, user relationships and preferences may change over time.

Tang et al. [35] used neural networks to analyze sentiments on reviews users gave on products and then used sentiment scores to predict future user ratings. Ma et al. [36] presented an approach that takes into account both user preferences and sentiments to predict the rating score. Authors used a lexicon-based approach for sentiment analysis of user reviews. Kim and Phalak [37] suggested studying user trust relationships based on ratings of user reviews.

Wang et al. [4] proposed to distinguish between *strong* and *weak* ties and to integrate the strength of these ties into Social Matrix Factorization (SMF) technique [38]. The updated model called Personalized Social Tie Preference Matrix Factorization (PTPMF) is presented in [8] wherein authors considered the personalized preference over the tie strength.

Mei et al. [39] proposed a trustee-influence based trust model where a trustee activeness or trustworthiness is used to determine trust relationships. Activeness is defined as the number of ratings made by a user, and trustworthiness is defined as the number of trustors the user has. The proposed trust model is incorporated into a memory-based and matrix factorization recommender system.

The above mentioned social recommendation approaches rely on manually predefined features for trust prediction. Approaches using manual feature engineering are critical as they rely on researchers experience. Deng et al. [9] 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.

Similar to our work, Covington et al. [40] also used multi-layer feed-forward neural networks for recommendation service in YouTube. However, this approach requires data preprocessing such as vector embedding, while 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 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 our approach 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 $|U| \times |I|$ user-item rating matrix R and a $|U| \times |U|$ user-user trust relation matrix T, where the matrices R and T have missing elements, the task is to predict missing rating scores in the matrix R.

B. Solution Overview

As we described in Section II, several approaches for item-rating prediction using trust information have been proposed. The input of the existing algorithms is composed of the user-item rating matrix and user-user relation matrix. Hence, we can define the predicting rating score by means of the 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, where $U_i \in U$, with $1 \leq i \leq |U|$ and $I_j \in I$ with $1 \leq j \leq |I|$. The definition of the function $\hat{f}$ depends on particular approaches. We denote the *actual* rating score from U_i to I_j as r_{ij} and the *theoretical* perfect predicting function as f.

Our approach is inspired by the universal approximation theorem [24]. 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 [41]. Studies also proved that a neural network can approximate a discontinuous function [42], and in practice the distinction between continuous and discontinuous functions is not an important factor for the approximation capacity of a neural network [43]. The approximation theorem states that, for any $\epsilon > 0$, we are able to construct a multi-layer feed-forward neural network F such that:

$$|F(R, T, i, j) - \hat{f}(R, T, i, j)| < \epsilon, \quad \forall i, j, 1 \leq i \leq |U|, 1 \leq j \leq |I| \tag{2}$$

Therefore, we argue that, if the function $\hat{f}$ was manually defined in previous rating prediction studies, a multi-layer neural network can also define this function. Our proposed approach has a low development cost. In fact, we do no try to approximate a particular existing function, but let the neural network 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, a multi-layer feed-forward neural network can approximate this f at an arbitrary precision level.

C. Deep Neural Network Model

In this subsection we motivate the selection of deep neural network model for our proposed solution and we describe this 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 [16].
- **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 [15]. 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 based our solution on deep learning models due to their following advantages [44]:

- They can learn complex nonlinear relationships between input and output data.
- They can learn without or with manual feature engineering, which is considered as one of the most difficult tasks in machine learning [45].
- They can scale well with a huge training set.
- They can learn well with unbalanced datasets and incorrect labelled data.

In what follows we briefly describe basic notions about *deep neural networks* (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* [46]. We follow [44], [47] by considering neural networks with at least two hidden layers as being deep learning models.

We visualize a DNN in Figure 2. The k^{th} layer 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.

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

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

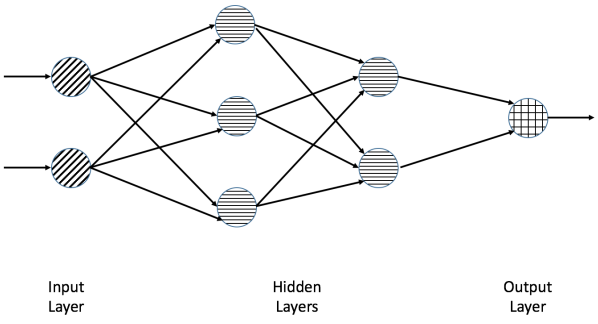


Fig. 2: A multi-layer feed-forward neural network.

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 [48], [40]. 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 [49].

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 [50] or GPU-computing technology such as CUDA [51].

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 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 [52] where input data is fed to a hash function to reduce the number of data dimensions.

In this study we do not feed the exact numeric values of user ID or item ID into the model, but 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 [40]. Learning complex data structure usually requires multiple hidden layers, but when the models

become deeper, "it becomes more difficult to obtain good generalization" [53]. In fact, this problem is still an open theoretical problem [54]. We follow the practical approach where we choose the number of hidden layers in a DNN model by "just keep adding layers until the test error does not improve anymore" [55].

IV. Experiments

In this section we present details about our experiments including adaptation of input data for DNN models, the dataset, description of the implementation, the metrics used for comparison with other baseline models and the results obtained.

A. Data Preparation

In this section we describe how information about a trust-user-item network containing positive links is fed into a DNN model.

The trust-user-item network is created over time, when links are added one by one. We can represent the trust-user-item network shown in Figure 1 as in Table I that contains user ratings on products over time 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 relations.

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

TABLE II: User-user relations.

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

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. The *Link Source* and *Link Destination* are fed as one-hot encoding vectors using hash function for dimension reduction [52].

Feeding raw data directly into a fully connected DNN was also successfully used in different tasks such as classification in computer vision or graph-based data [54], [47]. An advantage of using DNN is that the training time is much shorter than other complex algorithms such as ConvNet used for graph analysis [47].

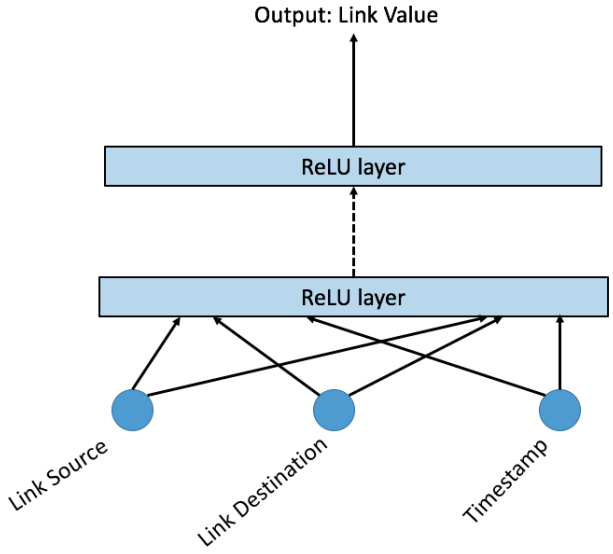


Fig. 3: Rating prediction using DNN. ReLU stands for Rectified Linear Unit.

B. Dataset

We evaluate our solution using the following three real-world datasets collected from Epinions[1] and Ciao[2] review websites where users can rate items and declare trust on other users:

- **Epinions1**: The dataset available at http://www.trustlet.org/ is collected and presented in [32].
- **Epinions2 & Ciao**: The two datasets available at http://www.jiliang.xyz/trust.html are collected and presented in [33].

Statistics about these datasets are presented in Table IV. All these datasets are organized in the format discussed in Section IV-A. Each dataset is composed of two files: one containing users rating score on items and one containing trust relations between users.

	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

The distributions of rating scores in the three datasets are displayed in Figure 4. We can see that all datasets are imbal-

[1]http://epinions.com/
[2]http://ciao.com/

anced: most of established rating scores are 4 or 5. Moreover, the distributions of rating scores in the three datasets are similar even the three datasets are collected at different times and from different websites.

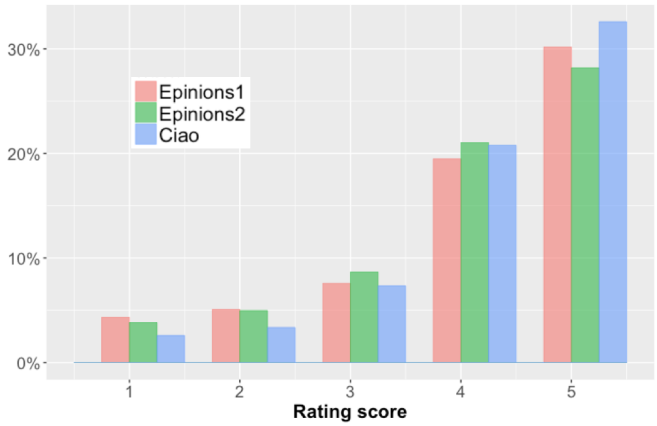


Fig. 4: Distribution of Rating Score in the three datasets.

We present in Figure 5 the distributions in the Epinions2 dataset of the number of trustors that trust a trustee and the number of trustees that are trusted by a trustor. Most of the time, a trustor trusts four trustees, and similarly a trustee is trusted by four trustors. Similar distributions are obtained throughout the other 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 6. We can see that the distribution does not change over time. Similar distributions are found in the other two datasets.

C. Implementation

We implemented our solution in R language[3]. We used *h2o* package[4] for building and training deep neural networks. *h2o* package performs deep neural network training on Hadoop and MapReduce [5].

We evaluated the performance of DNN models for general item-rating prediction, *warm-start* and *cold-start* problems. We used the same definition of *cold-start* problem as in [15]: *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.

Existing solutions use different division strategies for the training and testing datasets. In order to compare our approach with existing solutions, we performed two different experiments using the two main training/testing division strategies.

[3]https://www.r-project.org/

[4]http://www.h2o.ai/

[5]The code and data of our experiments are available at https://github.com/vinhqdang/dTrust

In the first experiment we employed the "sequential division" where we train the model using historical data and predict future data [33], [15]. In sequence, the testing data can be divided further into two parts that are the cold-start part and warm-start part. In the second experiment, we use leave-one-out cross validation. In this validation 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. At each time, we consider a different part as the testing data and the remaining $k-1$ parts as the training data. This method was used in [56], [30]. We compared the performance of our DNN models with the performance metrics presented in corresponding papers.

As suggested by Bengio [55], 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 make a significant effect on the performance of the deep neural network models. We used the *pyramid* architecture [23], meaning that the number of neurons is reduced in deeper layers. The same architecture is used in previous studies [40]. 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 in the second layer, then 512 neurons in the third layer if this layer was included and so on.

We used adaptive learning rate with the initial value as 0.001. We set the output size of the feature hashing function as $5,000$.

D. Metrics

We used 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 [57].

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.

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

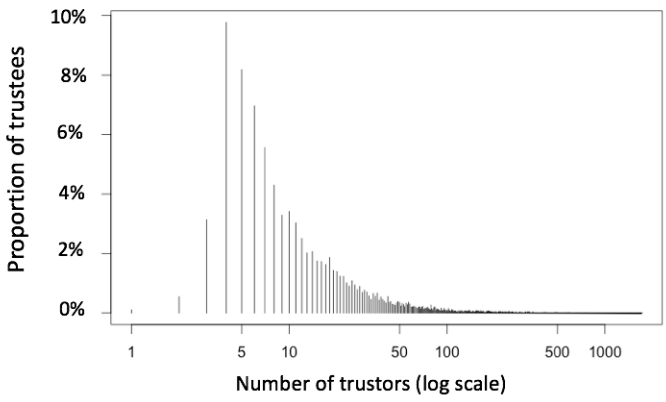
$$MAE = \frac{\sum_{i=1}^{n}|A_i - P_i|}{n} \tag{6}$$

RMSE punishes the large error more than MAE, therefore RMSE is more suitable for measuring the performance of recommender systems [30]. However, we present both metrics for our results for a comparison with existing studies.

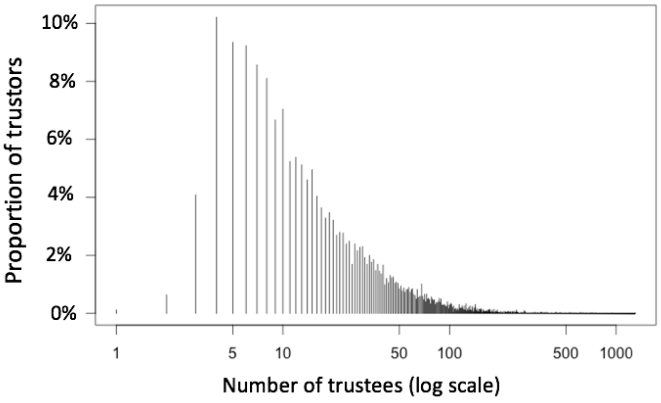
E. Baseline Models

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



(a) Distribution of number of trustors on trustees



(b) Distribution of number of trustees on trustors

Fig. 5: Distribution of number of trustors and trustees

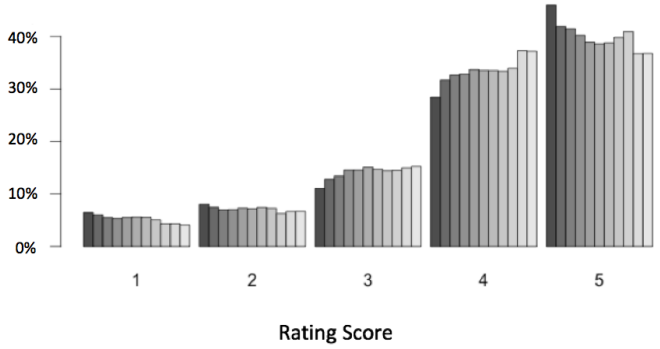


Fig. 6: Distribution of rating scores over 11 years in Epinions2 dataset. Each column represents one-year data.

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 [33]	2012	RMSE	Epinions2, Ciao
DynFluid [56]	2015	RMSE	Epinions2, Ciao
eTrustRec [58]	2015	RMSE	Epinions2
DLMF [9]	2016	RMSE	Epinions1
Davoudi [7]	2016	RMSE, MAE	Epinions2
Costa [59]	2016	RMSE	Epinions2, Ciao
TrustSVD [30]	2016	RMSE, MAE	Epinions1, Ciao
PTPMF [8]	2017	RMSE, MAE	Epinions1
MF-NTPR [39]	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 our experimental results. Comparison of our approach with corresponding state-of-the-art models in terms of RMSE and MAE values is presented in Tables VI, VII, VIII and IX. The values inside parentheses represent the number of hidden layers we used with *dTrust*.

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.

Algorithm	RMSE
Mean	1.17
NN	1.09
MF-NTPR	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.

Figure 7 displays the distribution of predicting error of dTrust in all experiments with different hidden layers. 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 is too

Algorithm	Epinions1	Ciao
Mean	1.21	1.06
NN	1.17	1.01
mTrust	-	0.97
DynFluid	-	0.97
DLMF	1.07	-
[59]	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).

Algorithm	Epinions1	Epinions2	Ciao
Mean	0.92	0.89	0.82
NN	0.89	0.88	0.80
[7]	-	0.87	-
TrustSVD	0.80	-	0.72
PTPMF	0.82	-	-
dTrust (1)	0.82	0.86	0.73
dTrust (2)	0.80	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.

deep and the last layer does not contribute much for the prediction [54].

V. Discussion

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

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. When predicting the entire testing dataset, the RMSE values 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 [40]. 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 [53], leading to the consequence that a DNN model cannot be extended forever [54].

In order to verify the improvement in predicting item-rating scores between our deep learning approach and existing methods, we performed *t-test* on the prediction errors of different methods, i.e. we performed *t-test* on two vectors E_1 and E_2, where E_1 is the prediction error of dTrust and E_2 is the prediction error of the second best method in each experiment. All obtained *p-values* are less than 0.05, so we confirm the significant improvement of our approach compared to other methods.

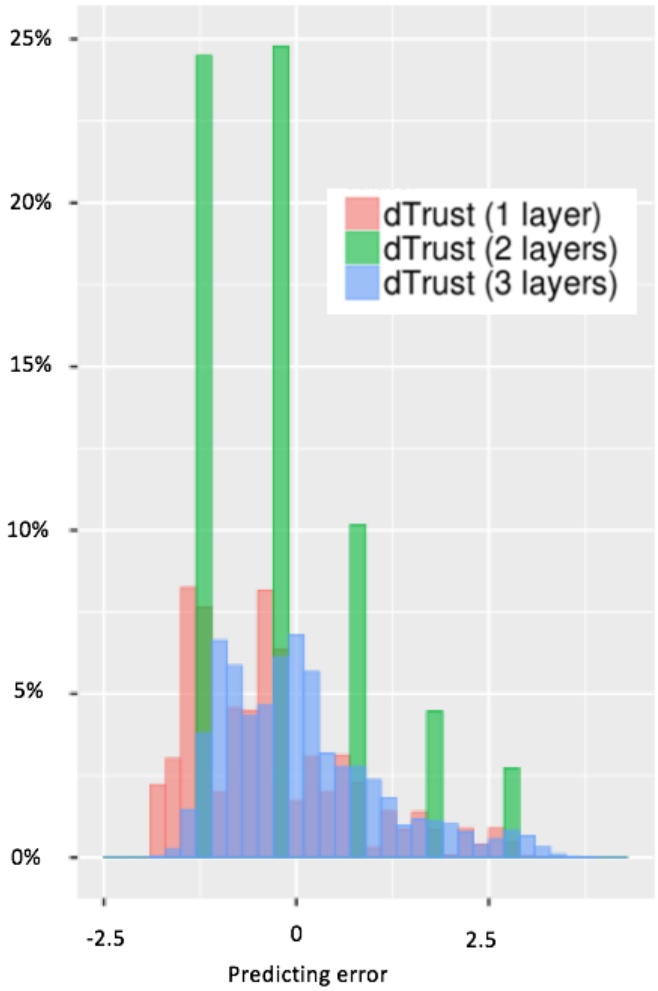


Fig. 7: Distribution of predicting error

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 [15], in rating prediction, small absolute improvements in RMSE can lead to a significant impact on quality of the top few recommendations which is critical for e-commerce recommender systems. For example, [34] showed that, the improvement of 0.0155 on absolute value of RMSE led to 50% relative improvement of top few recommendations.

Besides higher accuracy scores in prediction compared to state-of-the-art approaches, dTrust also benefits from the fact that we can feed the data directly into the model without feature engineering. Feature engineering is a very difficult task in machine learning which requires a lot of human effort and expertise [45]. Avoiding this step leads to preprocessing time improvement.

VI. Conclusion

Schwartz [60] claimed that increasing consumer choices can greatly increment shoppers anxiety. Recommender systems tackle this issue by suggesting customers few items among billions of them. Item-rating prediction is a critical issue in recommender systems.

In this paper, we improve the quality of rating prediction task by combining user relations with user-item rating 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. The solution does not require user personal information. In fact, the only information required by dTrust is the topology of the trust-user-item network where the IDs of users and items can be anonymized.

Our research study demonstrates the potential of using deep learning in studying user 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. We plan optimizing the proposed neural network model and extending it to capture other types of user 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] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*, 2015.

[3] 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.

[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] 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.

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

[10] C. A. Yeung and T. Iwata, "Strength of social influence in trust networks in product review sites," in *WSDM*. ACM, 2011, pp. 495–504.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] P. Massa and P. Avesani, "Trust-aware recommender systems," in *RecSys*. ACM, 2007, pp. 17–24.

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

[19] 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

[20] 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.

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

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

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

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

[25] 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.

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

[27] 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.

[28] 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.

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

[30] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1607–1620, 2016.

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

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

[33] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *WSDM*. ACM, 2012, pp. 93–102.

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

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

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *RecSys*. ACM, 2016, pp. 191–198.

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

[42] 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.

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

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

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

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

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

[48] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*. Omnipress, 2010, pp. 807–814.
[49] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-generation Machine Intelligence Algorithms*. O'Reilly Media, 2017.
[50] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *OSDI*. USENIX Association, 2004, pp. 137–150.
[51] 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.
[52] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *ICML*, 2009.
[53] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[55] Y. Bengio, "How can I estimate the number of neurons and layers?" May 2013. [Online]. Available: https://www.quora.com/Artificial-Neural-Networks-How-can-I-estimate-the-number-of-neurons-and-layers
[56] H. Zheng and J. Wu, "Dynfluid: Predicting time-evolving rating in recommendation systems via fluid dynamics," in *TrustCom/BigDataSE/ISPA (1)*. IEEE, 2015, pp. 1–8.
[57] 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.
[58] J. Tang, H. Gao, A. D. Sarma, Y. Bi, and H. Liu, "Trust evolution: Modeling and its applications," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1724–1738, 2015.
[59] G. Costa and R. Ortale, "Model-based collaborative personalized recommendation on signed social rating networks," *ACM Trans. Internet Techn.*, vol. 16, no. 3, pp. 20:1–20:21, 2016.
[60] B. Schwartz, "The paradox of choice: Why less is more," *New York: Ecco*, 2004.