Received March 6, 2020, accepted March 28, 2020, date of publication April 21, 2020, date of current version May 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989408

Utilizing an Autoencoder-Generated Item Representation in Hybrid Recommendation System

TAN NGHIA DUONG, TUAN ANH VUONG, DUC MINH NGUYEN, AND QUANG HIEU DANG
School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi 100000, Vietnam

Corresponding author: Tan Nghia Duong (nghia.duongtan@hust.edu.vn)

This work was supported by the Domestic Master's/Ph.D. Scholarship Programme of the Vingroup Innovation Foundation under Grant VINIF.2019.TS.43.

ABSTRACT While collaborative filtering (CF) is the most popular approach for recommendation systems, it only makes use of the ratings given to items by users and neglects side information about user attributes or item features. In this work, a natural language processing (NLP) technique is applied to generate a more consistent version of Tag Genome, a side information which is associated with each movie in the MovieLens 20M dataset. Subsequently, we propose a 3-layer autoencoder to create a more compact representation of these tags which improves the performance of the system both in accuracy and in computational complexity. Finally, the proposed representation and the well-known matrix factorization techniques are combined into a unified framework that outperforms the state-of-the-art models by at least 2.87% and 3.36% in terms of RMSE and MAE, respectively.

INDEX TERMS Collaborative filtering, matrix factorization, neighborhood-based, recommendation system, similarity measure.

I. INTRODUCTION

Nowadays, the habits of consumers have been greatly changed due to the rapid growth of information technology and networking. People have access to tremendous amount of online multimedia content, such as movies, music, news and articles. While this growth gives users more choices, it is more challenging for them to find relevant information. Or, in another perspective, it is critical that the system can provide automated and personalized recommendations to its users. Such systems are called recommendation systems (RS) these days [1]–[3].

In general, there are three main approaches for recommendation systems [4]: the content-based method, the collaborative filtering (CF) method, and the hybrid method. Content-based methods [5]–[9] suggest items based on the correlation between the item description and user's preference profile. This requires a substantial amount of item features and users' past behaviors. User preference models are then estimated by machine learning techniques such as stochastic gradient descent or mini-batch gradient descent. However, the main drawback of this content-based method is that the information representing item content is not always available or, if available, not reliable. In contrary, CF systems [10]–[14] generate recommendation of items based on the analogy of users with similar preference without making use of item content information. Furthermore, CF techniques can examine the similarity in preference between users based on their ratings on items. In more detail, CF methods can be classified into two groups: memory-based and model-based. Early implementations of RS are memory-based (aka neighborhood-based) where neighborhood algorithms are used to predict unknown ratings. Recent implementations of RS are more devoted to model-based techniques, after the success of matrix factorization model in Netflix Prize [15]. The fundamental idea of model-based approach is to learn a predictive model by analyzing the user-item interaction for estimation of missing ratings. Both types of CF often give better accuracy in prediction than content-based one due to the fact that the behavior of a specific user might be inferred from the behavior of users who share same tastes. Nevertheless, the main weakness

The associate editor coordinating the review of this manuscript and approving it for publication was Tossapon Boongoen.

of CF systems is that their performance decreases sharply when the rating matrices are very sparse. Unfortunately, this situation occurs frequently in practice because consumers are often not willing to provide their evaluation on items that they purchase or like. Furthermore, CF techniques are not capable of suggesting new items that have not yet any interaction with users, which is the cold-start problem. Consequently, hybrid methods [16]–[19] which utilize both side information and user preference appear to get the best of both worlds. The proposed model in this paper can be classified as a hybrid RS.

Hybrid methods can be categorized into two sub-classes: loosely coupled and tightly coupled methods [20]. Loosely coupled methods simply combine the outputs of individual content-based and collaborative filtering systems to make final ratings using a linear combination [21] or a voting scheme [22]. Tightly coupled methods are more sophisticated in integrating user-item ratings and auxiliary information to generate unified systems. In [17], authors incorporated user profiles, movie genres and past interaction data into a single model for predicting dyadic response in a generalized linear model framework. One limitation of this work is the usage of user profiles, which is currently a privacy issue that prevents users from providing their personal information. In [19], each item (scientific paper) is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from a dictionary of given unique words. Besides, there are citation (links) between the papers that make the dataset look like a social network which can be viewed as a graph in which the nodes represent the papers (objects) and the edges represent the links between objects. Making use of item content (word vector) and relationship between items (links), authors proposed a generalized latent factor model where content-related information is considered as features for the collaborative filtering methods. In [16], authors proposed a unified view of matrix factorization where additional sources of movie information (movie's genres, movie's actors) are crawled from the Internet Movie Database to augment the ratings in Netflix Prize data. Another model named Factorization Machines (FM) which combines the strength of matrix factorization and support vector machine techniques [23] is also capable of utilizing both rating and auxiliary information to make predictions. Nevertheless, the common point of these methods is that the features such as movie's genres or word vector for a scientific paper are considered as good representations of the item content. In practice, this assumption is not adequate for the recommendation task. It is often the case that raw content-based information needs to be processed carefully in order to be suitable for using in RS. This task is called feature engineering which is often performed manually and therefore tedious. It is even more challenging when the content of items is texts, images or videos, which always requires a hard and time-consuming work to discover good representation for items. Collaborative Filtering Regression (CTR) [18] is the state-of-the-art method leveraging textual information for recommendation, seamlessly integrates collaborative filtering and topic modeling. Although CTR is an appealing method which can produces interpretable recommendations with high accuracy, the representation capability of the model is limited to the topic model where items have a rich textual content information (the title and abstract of a scientific paper or the plot of a movie).

Recently, deep learning models have proven great potential for learning effective representations and gained dominant performance across many domains such as computer vision, speech recognition or text processing [24]–[29]. Nevertheless, there is relatively little work on developing deep learning techniques for recommendation tasks in contrast to the enormous amount of researches on CF. Reference [30] uses restricted Boltzmann machines instead of the traditional matrix factorization formulation to perform CF and [31] leverages user-user and item-item correlations to extend the original work. Even though, these models actually belong to CF methods due to the fact that they do not incorporate content information into making recommendations. Some models using a convolutional neural network or deep belief network for content-based music recommendation are described in [32], [33]. However, models for predicting latent factors from music audio are trained using the latent factors learned by applying weighted matrix factorization to usage data as ground truth. In other words, neural network is linked directly to the rating matrix, which means the performance degrades significantly when the ratings are highly sparse and MF fails. Recently, Collaborative Deep Learning (CDL) [20] has been proposed for joint learning a stacked denoising autoencoder (SDAE) and CF, and proven encouraging performance. Its idea is trying to learn a representation from item content through some denoising criteria: firstly, a corrupted version of the input is fed to an AE to reconstruct the original input; then the response of the encoder part is used as features of the CTR model. This work improves the well-known model CTR for the particular problem of article recommendation by replacing its Topic Model component with a Bayesian AE. Collaborative Denoising Autoencoder (CDAE) [34] might be regarded as a generative version of CDL which addresses the general top-N recommendation problem and the inputs are user behaviors instead of article/item features. A drawback of CDAE is that it does not take into account of side information (item features or user attributes) which can be important for producing semantically meaningful models and deal with the cold-start problems. Besides, CDL and CDAE both make use of *implicit* data which indicates whether a user likes/purchases an item or not. It means that *explicit* data (item ratings) which is a highly valuable information on user preference is not fully utilized. Therefore, these models mainly focus on top-N recommendation task, not suitable for rating prediction task. A newly proposed model using an AE for rating prediction task is item-based AutoRec (I-AutoRec) which estimates missing values by applying one AE per item whose input size is the number of known ratings [35]. In contrary to CDL and CDAE, I-AutoRec directly handles explicit data to make reliable rating predictions. However, I-AutoRec only considers user-item interaction and ignores secondary data like side information which may lead to the difficulty in explaining the produced recommendations.

In this paper, our work concentrates on movie rating prediction, a classic research topic in recommendation systems since the Netflix Prize. Our empirical studies are conducted on the latest version of the MovieLens dataset released in October 2016. The MovieLens 20M dataset consists of 20,000,263 ratings and 465,564 tag applications across 27,278 movies created by 138,493 users. There is no information about user profile; however, the dataset includes a current copy of the Tag Genome which was based on user-contributed content including tags, ratings and textual reviews [36]. In other words, movies in this dataset are associated with secondary data reflecting content-related information. To address the challenges mentioned above, we propose methods to utilize side information of movies available in the MovieLens 20M dataset in order to improve the performance of the traditional recommendation systems. The main contributions of this paper are summarized as follows.

- Utilize **word2vec**, an NLP technique, to preprocess the raw data included in the Tag Genome to produce a more consistent description of each movie.
- Apply auto-encoder, a deep learning technique, on the cleaned version of Tag Genome to generate a more compact and accurate representation for each movie which not only reduces the error rates of predicted ratings but also speeds up the whole system.
- Integrate the output of matrix factorization model (**SVD++**) as the baseline estimate of user rating into the hybrid content- and neighborhood-based system to provide more precise recommendations over the state-of-the-art techniques.

The rest of the paper is organized as follows. Section II formalizes the problem and discusses existing solutions for rating prediction task. Section III summarizes our previous work. Experimental settings are described in Section IV. The proposed models and their performance are presented along with state-of-the-art techniques for comparison in Section V. We conclude with a summary of this work and discussion of future work in Section VI.

II. PRELIMINARIES

In this paper, u, v denote users and i, j denote items. The preference by user u for item i is denoted by r_{ui}, also known as the rating, where high values indicate strong preference. The (u, i) pairs for which r_{ui} is known are stored in the set $\mathcal{K} = \{(u, i) | r_{ui} \text{ is known}\}$. U_{ij} is the set of all users that rate both items i and j, and U_i is the set of all users that rate item i. The task is to predict the unknown rating $\hat{r}_{ui}$ if user u has not rated item i before. Two popular CF techniques for the rating prediction are briefly formulated as follows.

A. MEMORY-BASED CF

There are two types of memory-based (or neighborhood-based) CF: (i) user-oriented (or user-user) model [37] and (ii) item-oriented (or item-item) model [38], [39] of which the latter is gaining more successes in practice [2]. An item-item CF system (**ii-CF**) finds the most relevant items to the item which was purchased or liked by a specific user and recommend them to her. The central component of these systems is a measure indicating the similarity degree s_{ij} between two items which can be computed using common formulas such as cosine similarity function (**Cos**) or Pearson correlation coefficients (**PCC**) as follows.

$$s_{ij}^{Cos} = \cos(x_i, x_j) = \frac{\sum_{u \in U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U_i} r_{ui}^2} \sqrt{\sum_{u \in U_j} r_{uj}^2}} \quad (1)$$

$$s_{ij}^{PCC} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}} \quad (2)$$

where μ_i, μ_j are the average mean ratings of items i and j, respectively.

Recently, a modified version of (2) was proposed replacing μ_i, μ_j by baseline estimates b_{ui}, b_{uj} which account for the user and item effects:

$$\hat{\rho}_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - b_{ui}) \cdot (r_{uj} - b_{uj})}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{uj} - b_{uj})^2}} \quad (3)$$

Then a shrunk correlation coefficient which helps avoid overfitting when two items share only few common raters is integrated into (3) to create a new similarity measure named **PCCBaseline**:

$$s_{ij}^{PCCBaseline} = \frac{|U_{ij}| - 1}{|U_{ij}| - 1 + shrinkage} \cdot \hat{\rho}_{ij} \quad (4)$$

where $|U|$ is the number of common users between items i and j, and *shrinkage* is the shrunk correlation coefficient [40].

Let $S^k(i, u)$ denote the set of k most similar items to i rated by user u, then the predicted value of r_{ui} can be computed as a weighted average of the ratings of similar items (named **kNNBasic** model):

$$\hat{r}_{ui}^{kNNBasic} = \frac{\sum_{j \in S^k(i,u)} s_{ij} r_{uj}}{\sum_{j \in S^k(i,u)} s_{ij}} \quad (5)$$

or as a weighted average of the ratings of the similar items while adjusting for user and item effects through the baseline estimates (named **kNNBaseline** model [40]):

$$\hat{r}_{ui}^{kNNBaseline} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij} (r_{uj} - b_{uj})}{\sum_{j \in S^k(i;u)} s_{ij}} \quad (6)$$

B. MODEL-BASED CF

Latent factor models are typical model-based CF techniques aiming at uncovering latent features that explain the observed ratings, among which the matrix factorization ones have proved their superior accuracy and flexible scalability in the Netflix Prize [41]. By using **SVD** factorization, both users and items are mapped into a latent space of dimension k, where each user can be characterized by a user-factors vector

$p_u \in \mathbb{R}^k$ and each item by an item-factors vector $q_i \in \mathbb{R}^k$. The prediction is done by taking an inner product $\hat{r}_{ui} = q_i^T p_u$. An extended version of **SVD**, named **SVD++**, was proposed to improve the accuracy by taking into account *implicit* feedbacks for additional indication of user preferences. That is why a second set of item factors is added, relating each item i to a factor vector $y_i \in \mathbb{R}^k$. The predicted rating is computed as follows.

$$\hat{r}_{ui} = q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \tag{7}$$

where $R(u)$ contains the items rated by user u [42].

III. PREVIOUS WORK

In [43], we analyzed the distribution of the similarity scores calculated using two commonly used formulas: cosine similarity (**Cos**) and Pearson correlation coefficient (**PCC**) based on the rating information. Intensive experiments on the original MovieLens 20M dataset showed that the values of similarity degree between two arbitrary items are 97% distributed in the range of [0.85; 1] with a coefficient of variation of 4.83%. Such a small coefficient of variation makes it difficult to distinguish a pair of two relevant items from a pair of two irrelevant ones. This badly affects the item-oriented models which utilize the similarity degree between two items to make useful recommendations. Based on this observation, we proposed new similarity measures which could achieve a wider spectrum of the similarity degree by using the cubed version of the traditional formulas, named **cubedCos** and **cubedPCC**, as follows.

$$s_{ij}^{cubedCos} = (s_{ij}^{Cos})^3 \tag{8}$$

$$s_{ij}^{cubedPCC} = (s_{ij}^{PCC})^3 \tag{9}$$

where s_{ij}^{Cos}, s_{ij}^{PCC} are similarity measures calculated using **Cos** and **PCC**, respectively. Experimental results on the original MovieLens 20M dataset showed that newly proposed measures totally outperform their counterparts at accuracy: the item-oriented CF model using **cubedPCC** produces 6.4% lower RMSE than using **PCC**.

In [44], we noticed that similarity measures using the rating information faces some problems. Firstly, in practice the rating matrix is highly sparse (for example, 99.47% of the ratings in the MovieLens 20M dataset are missing); therefore, evaluating the relevance between two movies that have many ratings but share only few common users using above similarity measures is not reliable. Secondly, calculating similarity between two movies in practical recommendation systems is a time consuming task due to the large number of users (often in order of millions of users). To solve these problems, a novel similarity measure was proposed using Genome Tag instead of rating information. In more detail, each movie is characterized by a genome score vector $\mathbf{g} = \{g_1, g_2, \ldots, g_{1128}\}$ which encodes how strongly a movie exhibits particular properties represented by 1,128 tags [36], and the similarity s_{ij} between movies i and j is calculated as follows.

$$s_{\mathbf{g}_i,\mathbf{g}_j}^{Cos_{\text{genome}}} = \frac{\sum_{k=1}^{G} g_{i.k} g_{j.k}}{\sqrt{\sum_{k=1}^{G} g_{i.k}^2}\sqrt{\sum_{k=1}^{G} g_{j.k}^2}} \tag{10}$$

or

$$s_{\mathbf{g}_i,\mathbf{g}_j}^{PCC_{\text{genome}}} = \frac{\sum_{k=1}^{G}(g_{i.k} - \bar{g}_i)(g_{j.k} - \bar{g}_j)}{\sqrt{\sum_{k=1}^{G}(g_{i.k} - \bar{g}_i)^2}\sqrt{\sum_{k=1}^{G}(g_{j.k} - \bar{g}_j)^2}} \tag{11}$$

where $\bar{g}_i$ and $\bar{g}_j$ are the mean genome scores of vectors $\mathbf{g}_i$ and $\mathbf{g}_j$, respectively; and $G = 1128$ is the length of genome vectors. Experiments conducted on the preprocessed MovieLens 20M dataset (keeping only movies with Tag Genome) showed that the item-oriented CF models based on similarity measures $\mathbf{Cos_{genome}}$ and $\mathbf{PCC_{genome}}$ provide accuracy equivalent to the state-of-the-art CF models using rating information whilst performing at least 2 times faster.

TABLE 1. Summary of the original MovieLens 20M and the preprocessed dataset.

	# Ratings	# Users	# Movies	Sparsity
Original dataset	20,000,263	138,493	27,278	99.47%
Preprocessed dataset	19,793,342	138,185	10,239	98.97%

IV. EXPERIMENTAL SETUP

A. DATASET

In order to evaluate the performance of the models presented in this paper, the MovieLens 20M dataset is used as a benchmark. The dataset, released by GroupLens in 2016, originally contains 20,000,263 ratings and 465,564 tag applications across 27,278 movies created by 138,493 users (all selected users had rated at least 20 movies). The ratings are float values ranging from 0.5 to 5.0 with a step of 0.5. Different from the previously released datasets of GroupLens, this dataset includes a current copy of the Tag Genome which was computed on user-contributed content including tags, ratings, and textual reviews [36].

Because the proposed system in this work makes use of the information in tag genome vectors, it is necessary to apply a preprocessing step into the original dataset. In more detail, we firstly drop out the movies which do not have tag genome. After that, only movies and users with at least 20 ratings are kept. The preprocessed dataset now consists of 19,793,342 ratings (approximately 98.97% compared with the original dataset) given by 138,185 users for 10,239 movies.

B. EVALUATION SCHEME

After preprocessing, the dataset is split into 2 distinct parts: 75% ratings of each movie are used as the training set and the 25% remaining ratings as the testing set. To compare the overall performance between models, three indicators are used: RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) for accuracy evaluation, and Time [s] for timing evaluation. Here, RMSE and MAE are calculated using the following formulas.

$$RMSE = \sqrt{\sum_{u,i \in \text{TESTSET}} \left(\hat{r}_{ui} - r_{ui}\right)^2 / |\text{TESTSET}|} \tag{12}$$

$$MAE = \sum_{u,i \in \text{TESTSET}} \left|\hat{r}_{ui} - r_{ui}\right| / |\text{TESTSET}| \tag{13}$$

where |TESTSET| is the size of testing set, $\hat{r}_{ui}$ is the predicted rating estimated by the model, and r_{ui} is the actual rating made by user in the testing set. Timing is measured as the total duration for learning the model on the training set and predicting all samples in the testing set.

All experiments are carried out on a workstation consisting of an Intel®Xeon®Processor E5-2637 v3 3.50 GHz (2 processors), 32 GB RAM and no GPU.

C. BASELINES AND EXPERIMENTAL SETTINGS

In order to evaluate the overall performance of the proposed models in this paper, some popular methods for rating prediction are implemented as baseline models.

- **ii-CF** [39]: **PCCBaseline** is used to measure the similarity between movies and the number of neighbors is set at 40.
- **SVD** [41] and **SVD++** [42]: both models are trained using 40 hidden factors with 100 iterations and step size of 0.002.
- **kNNBaseline$_{\textbf{genome}}$** [44]: **PCC$_{\textbf{genome}}$** is used as the similarity measure.
- **I-RBM** [30]: an item-based RBM is trained over 50 epochs with batch size of 1,000, learning rate of 0.01/batch size, momentum of 0.9 and a weight decay of 0.01.
- **FM$_{\textbf{genome}}$** [23]: each feature vector is composed of user and movie ID, movie genres and original genome scores associated with each movie; the model is trained with degree $d = 2$ and 50 iterations.
- **I-AutoRec** [35]: a 1-layer AE is trained using 600 hidden neurons, and the combination of activation functions is $(Identity, Sigmoid)$.

In our experiments, the optimal hyperparameters for each baseline methods are carefully chosen using 5-fold cross validation to guarantee fair comparisons. For **ii-CF** model, the number of neighbors is picked from $\{10, 20, 30, 40, 50, 100, 150\}$, and the similarity measures implemented are **Cos**, **PCC** and **PCCBaseline**. **SVD++** model is implemented with the number of latent factors $k \in \{10, 20, 30, 40, 60, 80, 100, 120\}$. For **I-AutoRec** model, the size of the hidden layer is set as $n \in \{200, 400, 600, 800, 1000\}$ units; and the choices of activation functions $f(\cdot), g(\cdot)$ are experimented with $(Identity, Identity)$, $(Identity, Sigmoid)$, $(Sigmoid, Identity)$, $(Sigmoid, Sigmoid)$. Finally, the regularization strength is tuned $\lambda \in \{0.001, 0.01, 0.1, 1, 10\}$ for all baselines.

V. PROPOSED MODEL

In this paper, we have three main contributions. Firstly, when investigating the genome scores information used in our previous article [44], we found that the total number of tags can be reduced by combining similar tags together while still getting competitive results. Secondly, we could even compress this information further by a method in deep learning called autoencoder to automatically learn the hidden representation of the genome scores. Finally, the resulting information can be combined with the global information caught by the state-of-the-art models like SVD and SVD++ to improve the overall performance of the neighborhood models.

A. CLUSTERING RELEVANT GENOME TAGS

We find from genome data that there are many tags which share the same meaning but have different names. This happens because GroupLens allows users to choose tags that they find most appropriate with the movie without any limitation. For instance, both user A and user B know that *Captain America: The Winter Soldier (2014)* is a super hero movie; however, user A can attach tag **superhero** for this movie while user B can choose tag **super-hero**. Other examples are displayed in Table 2. In theory, this would not be a problem if the relevance values corresponding to similar tags are the

TABLE 2. Six movies along with a group of closely related genome tags from the MovieLens 20M dataset. New tag is assigned a composite score calculated using the *mean* or *median* value of individual ones.

Genome Tag	The 40-Year-Old Virgin	Despicable Me	Grown Ups	Kick-Ass	Kung Fu Panda	Toy Story	Toy Story 3
fun	0.30	0.65	0.67	0.52	0.72	0.88	0.61
fun movie	0.26	0.61	0.49	0.68	0.62	0.82	0.64
funniest movies	0.80	0.05	0.10	0.11	0.06	0.04	0.05
funny	0.92	0.87	0.87	0.69	0.76	0.69	0.60
funny as hell	0.78	0.15	0.11	0.30	0.20	0.16	0.15
humor	0.84	0.64	0.46	0.89	0.66	0.63	0.57
humorous	0.83	0.70	0.53	0.69	0.72	0.69	0.59
fun_new$_{\text{mean}}$	0.68	0.52	0.46	0.56	0.54	0.56	0.46
fun_new$_{\text{median}}$	0.80	0.64	0.49	0.68	0.66	0.69	0.59
Difference	0.12 (17.65%)	0.12 (23.08%)	0.03 (6.52%)	0.12 (21.43%)	0.12 (22.22%)	0.13 (23.21%)	0.13 (28.26%)

TABLE 3. Clustering relevant tags together. For brevity, we just show several samples here, but there are 148 tags to be combined into 64 groups.

Original tag	New tag
007 007 (series)	007_new
soccer football	football_new
gangs gangster gangsters	gangster_new
good acting good action great acting	good_acting_new

same or at least close to each other so that the similarity calculation would not be affected. Nonetheless, when analyzing the dataset we see that these values are frequently distributed across a large range. For example, tag *fun movie* of movie *The 40-Year-Old Virgin* has a genome score of 0.26; at the meantime, tag *funny*'s score is 0.92, and other analogous tags such as *fun*, *funniest movies* and *funny as hell* have scores varying in a large range from 0.30 to 0.80. This situation occurs regularly as can be seen in Table 2. Underlined genome scores represents the extreme values in a typical group of relevant tags for six movies. Obviously, content-related information of a movie cannot be described exactly using such largely distributed values. This affects negatively the accuracy of evaluating the analogy between two movies using genome scores as in [44].

To eliminate the effect of freely user-created tags, we propose to apply a mapping process: original tags which share the common context are grouped into a new tag associated with a composite score. More specifically, a cleaning step including lemmatization and removing stop words and non-alphabetic characters is performed to generate appropriate form of raw tag genome. Then a natural language processing technique named **word2vec** [45] is used to cluster the same meaning tags. In this work, we use spaCy library[1] to implement pre-processing and calculate the semantic similarity between genome tags: two tags are considered to share analogous meaning if their similarity score is greater than a fixed threshold (chosen as 0.65 in our experiments). After clustering similar tags, the size of genome vector is reduced from 1,128 to 1,044. Table 5 demonstrates four of the newly combined tags while Figure 1 illustrates the strength of semantic relationship corresponding to each pair of these original tags.

Finally, a composite score is assigned to the new tag. Two methods to calculate this score are deployed in this paper: *mean* and *median*. As can be seen in Table 2, (*fun*, *fun movie*, *funniest movies*, *funny*, *funny as hell*, *humor*, *humorous*) are considered as closely related tags and grouped into a new

[1]https://github.com/explosion/spaCy

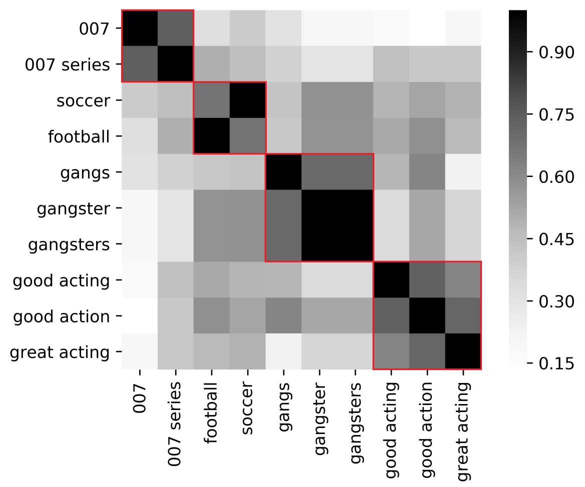


FIGURE 1. The magnitude of semantic similarity between some genome tags.

one named fun_{new}. Then a score is attached to fun_{new} using the mean/median value of the individual scores. Clearly, there is a significant disagreement between two methods: in this example, the relative difference is approximately 22% in most cases. The best choice is determined by substituting both values into (10) and (11) to calculate the similarity between two movies. In order to evaluate the effect of clustering similar tags, two baseline models utilizing the content-based information in the rating prediction are implemented: $\mathbf{kNNBaseline_{genome}}$ and $\mathbf{FM_{genome}}$. **kNNBaseline** model with different values of neighborhood size is implemented to evaluate the performance of new genome tags.

For the purpose of comparison, the error rates and complexity of $\mathbf{kNNBaseline_{genome}}$ and $\mathbf{FM_{genome}}$ models using original and newly generated tags are presented in Tables 4 and 5, respectively. Experimental results show that

TABLE 4. Performance of $\mathbf{kNNBaseline_{genome}}$ and $\mathbf{FM_{genome}}$ models using 1,128 original genome tags.

Model		RMSE	MAE	Time [s]
kNNBaseline (k=40)	$Cos^{original}_{genome}$	0.8202	0.6243	1,521
	$PCC^{original}_{genome}$	0.7912	0.6034	1,701
kNNBaseline (k=10)	$Cos^{original}_{genome}$	0.8037	0.6135	1,360
	$PCC^{original}_{genome}$	0.7905	0.6025	1,474
$FM^{original}_{genome}$		0.7918	0.6037	42,788

TABLE 5. Performance of kNNBaseline and $\mathbf{FM_{genome}}$ models using 1,044 new genome tags.

Model		RMSE		MAE		Time [s]
		Mean	Median	Mean	Median	
kNNBaseline (k=40)	Cos^{new}_{genome}	0.8123	0.8102	0.6187	0.6173	1,472
	PCC^{new}_{genome}	0.7894	0.7888	0.5992	0.5983	1,644
kNNBaseline (k=10)	Cos^{new}_{genome}	0.7992	0.7981	0.6102	0.6094	1,312
	PCC^{new}_{genome}	0.7886	**0.7875**	0.5982	**0.5975**	**1,398**
FM^{new}_{genome}		0.7906	0.7898	0.6025	0.5995	40,106

using median value for new tags is a better choice. The large gap between mean and median values, as seen in Table 2, is due to the appearance of abnormal scores which are much lower or higher than the majority. Therefore, a mean value is heavily affected by these outliers while a median one could effectively eliminate them. The benefit of clustering relevant tags has been demonstrated at both the accuracy and timing indicators of all models. It can be seen that **kNNBaseline$_{\mathbf{genome}}$ (k=10)** model using **PCC$_{\mathbf{genome}}$** as similarity measure works best with both original and new tags. However, substituting original tags with new ones helps lower RMSE by 0.38% and MAE by 0.83% whilst performing 5.16% faster. Obviously, combining the same meaning tags together not only creates a more precise representation for each movie but speeds up the process of measuring similarity degree due to using shorter genome vectors.

B. LEARNING NEW REPRESENTATION FOR EACH MOVIE WITH AN AUTOENCODER

Experiments in the previous section show that cleaning original data slightly improves the accuracy of the recommendation system. However, the number of new tags is rather large (reduced by about 7% from raw ones); more importantly, there may still exist groups of tags which are to some extent related to each other. In other words, combining genome tags based on only the semantic similarity may not explore hidden links between tags. It is desirable to generate a more concise and accurate representation for each movie which can capture concealed but valuable information about the relationship between tags.

Among current techniques for data engineering and learning representation, an autoencoder is widely used to discover latent features embedded in raw data. It not only eliminates the information redundancy but also generates new data representation which is more precise and efficient [24], [46]. The simplest form of an autoencoder is a feedforward, non-recurrent neural network which has an input layer and an output layer with the same number of nodes, and one or more hidden layers connecting them. An example of a 1-layer autoencoder is illustrated in Figure 2. This neural network is trained to minimize the difference between the input and the output. Therefore, it can be considered that an autoencoder is constituted by two main parts: an encoder that maps the input into the code, and a decoder that reconstructs the original input from the code. In practice, only the first part of this architecture is generally used to create a compressed representation of the input that preserves the most relevant information.

In this work, to keep reducing the dimension of genome tags and learn hidden structures we attempt to apply an autoencoder to newly created tags in the previous section. Firstly, a 1-layer autoencoder is implemented with input and output layers having 1,044 neurons corresponding to 1,044 new tags. **kNNBaseline** with $k = 10$ and **FM$_{\mathbf{genome}}$** are chosen to evaluate the performance of new representations. Furthermore, we also apply a 1-layer autoencoder with

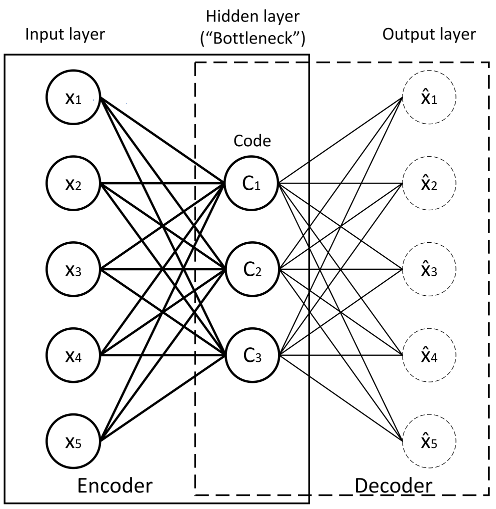


FIGURE 2. Illustration of an autoencoder with 1 hidden layer.

1,128 nodes at input and output layers to original genome tags for comparison. Similarity measure of **kNNBaseline** model is still implemented with two options **Cos$_{\mathbf{genome}}$** and **PCC$_{\mathbf{genome}}$** to determine the best method. The number of hidden units is decreased from $1,000$ to 300 with the step of 100 to find the optimal value. A grid search is performed showing that the hyperparameters $learning\_rate = 0.01$, $dropout = 0.2$, $num\_epochs = 50$, $regularization = 0.01$ gives good performance on the test set. Experimental results are displayed in Table 6 where the optimal model is highlighted.

The advantage of the data cleaning process in the previous section is once again demonstrated in Table 6: taking the newly created genome tags as the input of the autoencoder generates a more precise representation for each movie than original ones in all cases. Hence, we only focus on the cleaned version of genome tags hereafter.

Figure 3 shows RMSE and MAE at different sizes of hidden layer. The best result in the previous section is used as reference (green lines **Ref.**). When the number of hidden units is in the range [1,000; 800], the accuracy of the proposed models has a modest improvement. However, when the hidden layer has smaller sizes error rates drop out sharply and reach the minimum value at 600. Compared to the reference model, we find that encoding 1,044 genome tags as a 600-element feature vector not only decreases the time complexity but enhances the accuracy of the recommendations. **kNNBaseline** model with **Cos$_{\mathbf{genome}}$** provides a 0.19% lower RMSE and a 0.12% lower MAE while performing 1.39 times faster. With **PCC$_{\mathbf{genome}}$**, the improvement over the reference model is most impressive: our model has a 2.15% lower RMSE and a 2.03% lower MAE while speeding up the whole system by 1.33 times. **FM$_{\mathbf{genome}}$** model ranks the second in terms of accuracy; nonetheless, it performs significantly slower than its counterparts. The lower error rates indicates that the autoencoder can find hidden relationships among

TABLE 6. Performance of kNNBaseline and FM$_{genome}$ models at different sizes of the hidden layer.

#Hidden units	Model		1,128 original tags			1,044 new tags		
			RMSE	MAE	Time [s]	RMSE	MAE	Time [s]
1,000	kNNBaseline	Cos$_{genome}$	0.7987	0.6098	1,311	0.7933	0.6049	1,296
		PCC$_{genome}$	0.7862	0.5969	1,330	0.7819	0.5939	1,314
	FM$_{genome}$		0.7880	0.5978	39,838	0.7854	0.5962	39,822
900	kNNBaseline	Cos$_{genome}$	0.7976	0.6085	1,226	0.7922	0.6042	1,203
		PCC$_{genome}$	0.7851	0.5960	1,295	0.7805	0.5922	1,274
	FM$_{genome}$		0.7865	0.5971	36,026	0.7833	0.5947	36,003
800	kNNBaseline	Cos$_{genome}$	0.7963	0.6074	1,160	0.7910	0.6033	1,138
		PCC$_{genome}$	0.7836	0.5949	1,132	0.7783	0.5898	1,210
	FM$_{genome}$		0.7852	0.5960	30,681	0.7818	0.5939	30,658
700	kNNBaseline	Cos$_{genome}$	0.7939	0.6056	1,101	0.7883	0.5980	1,075
		PCC$_{genome}$	0.7798	0.5916	1,154	0.7742	0.5880	1,129
	FM$_{genome}$		0.7828	0.5944	27,372	0.7792	0.5909	27,346
600	kNNBaseline	Cos$_{genome}$	0.7913	0.6034	1,032	0.7860	0.5968	1,003
		PCC$_{genome}$	0.7767	0.5890	1,078	**0.7706**	**0.5854**	**1,048**
	FM$_{genome}$		0.7781	0.5896	23,382	0.7743	0.5881	23,352
500	kNNBaseline	Cos$_{genome}$	0.7982	0.6094	1,013	0.7934	0.6050	980
		PCC$_{genome}$	0.7916	0.6036	1,106	0.7857	0.5964	1,073
	FM$_{genome}$		0.7935	0.605	21,135	0.7880	0.5978	21,102
400	kNNBaseline	Cos$_{genome}$	0.8036	0.6135	959	0.7988	0.6099	924
		PCC$_{genome}$	0.8003	0.6110	1,047	0.7942	0.6058	1,011
	FM$_{genome}$		0.8027	0.6129	15,912	0.7970	0.6082	15,876
300	kNNBaseline	Cos$_{genome}$	0.8080	0.6160	934	0.8036	0.6135	895
		PCC$_{genome}$	0.8061	0.6151	1,032	0.8002	0.6110	994
	FM$_{genome}$		0.8075	0.6156	8,794	0.8024	0.6127	8,756

the genome tags and learn a more accurate representation for each movie. Besides, the reduced computational complexity in neighborhood-based models is owing to describing a movie with an approximately 43% shorter feature vector which helps reduce the time to calculate the similarity degree between movies. However, all proposed systems work much worse if we keep decreasing the size more: compressing the data input to a very low dimension may cause a huge information loss which eventually leads to irrelevant suggestions.

Normally, a deep neural network often outperforms a shallow one due to the capability of exploring more latent features under the raw data. Therefore, we experiment with adding more hidden layers to the autoencoder and evaluate the performance changes. 3-, 5- and 7-layer autoencoders with the bottleneck of 600 units are deployed with the same hyperparameters as above. Table 7 shows that a 3-layer autoencoder which employs 2 layers at the encoder part generates a more robust representation for each movie than a simple 1-layer autoencoder. **kNNBaseline** with **PCC$_{genome}$** still works best in all experiments: compared to the reference model, its error rates are reduced by 2.32% and 2.24% in terms of RMSE and MAE, respectively. Deeper architectures with more layers in the encoding part does not help to improve the accuracy so a 3-layer autoencoder is regarded as the best choice.

TABLE 7. Performance of kNNBaseline and FM$_{genome}$ models when using a deeper AE.

#Layers	Model		RMSE	MAE	Time [s]
3	kNNBaseline	Cos$_{genome}$	0.7822	0.5942	1,046
		PCC$_{genome}$	**0.7692**	**0.5841**	**1,092**
	FM$_{genome}$		0.7702	0.5850	23,412
5	kNNBaseline	Cos$_{genome}$	0.7895	0.5992	1,116
		PCC$_{genome}$	0.7754	0.5890	1,230
	FM$_{genome}$		0.7776	0.5895	23,531
7	kNNBaseline	Cos$_{genome}$	0.7904	0.6024	1,238
		PCC$_{genome}$	0.7772	0.5894	1,371
	FM$_{genome}$		0.7801	0.5919	23,712

Empirical results also show that **PCC$_{genome}$** consistently works better than **Cos$_{genome}$**. A possible explanation is that **PCC$_{genome}$** applies a mean-centering procedure on vectors, thereby the calculation of similarity degree between two

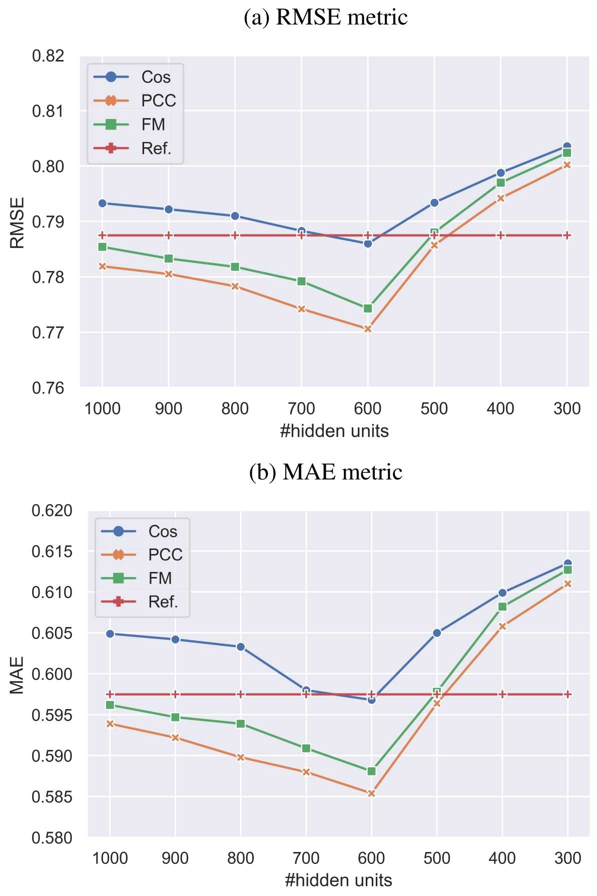


FIGURE 3. Plot of error rates with respect to the number of hidden units. Reference model (not using autoencoder) is depicted as the horizontal line.

TABLE 8. Comparison between the proposed model and baseline models.

Model	RMSE	MAE	Time [s]
ii-CF	0.8046	0.6140	2,486
SVD	0.7922	0.6042	14,892
SVD++	0.7894	0.5992	124,224
I-RBM	0.7951	0.6065	96,455
I-AutoRec	0.7808	0.5931	69,860
kNN-Content^{AE}	**0.7692**	**0.5841**	**1,092**

vectors does not take into account their analogy in absolute values, only considers if they vary in the same way. We name our selected model, **kNNBaseline** with $k = 10$ using **PCC$_{genome}$** on 600-element feature vectors compressed from 1,044 new genome tags by a 3-layer AE, as **kNN-Content^{AE}** and then compare it with baseline methods to evaluate the overall performance. Experimental results in Table 8 demonstrates the superior of our proposed model against the state-of-the-art techniques. Compared to **SVD++**, our model totally outperforms in terms of accuracy and time complexity: **kNN-Content^{AE}** not only achieves 2.56% lower RMSE and 2.52% lower MAE but works 113.76 times faster. **I-AutoRec**, an AE-based recommendation system, also produces 1.51% higher RMSE and 1.54% higher MAE but requires approximately 64x time complexity than the proposed model. Compared to another popular deep learning-based system, **I-RBM**, our model even operates more impressively: the error rates are 3.26% and 3.69% lower in terms of RMSE and MAE, respectively, while the duration for both training and testing phases is 88.33 times shorter.

C. INTEGRATING WITH MATRIX FACTORIZATION TECHNIQUES

Up to now, the proposed model can be regarded as a combination of content- and item-based neighborhood models: raw information indicating the content of a movie is compressed using an autoencoder to generate a feature vector which is used in the process of measuring the similarity between two movies. While neighborhood-based models could capture local-level information and make reasonable recommendations promptly, their matrix factorization counterparts are capable of extracting global-level information embedded in the rating matrix in order to produce more accurate suggestions at the cost of computational complexity. To enhance the accuracy of the proposed hybrid model, a solution is to integrate global-level characteristics explored by matrix factorization methods into the system.

Recalling the rating prediction of **kNNBaseline** model in Section II-A:

$$\hat{r}_{ui}^{kNNBaseline} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij} \left(r_{uj} - b_{uj} \right)}{\sum_{j \in S^k(i;u)} s_{ij}} \tag{14}$$

where b_{ui} is the baseline estimate of the preference by user u for item i and calculated as:

$$b_{ui} = \mu + b_u + b_i \tag{15}$$

The parameters b_u and b_i correspond to the observed deviations of user u and item i, respectively. These parameters b_u and b_i can be estimated from the least squares problem as in [42]. The unknown rating r_{ui} is composed of two parts: the former is a coarse estimate and the latter serves as a fine tuning against the former to generate a superior prediction. Obviously, b_{ui} is the *bottleneck* of the prediction: an imprecise, or even not good enough, baseline estimate will lead to an incorrect final rating.

We propose to replace b_{ui} by the rating produced by matrix factorization methods. Therefore, the final results can have the advantages of both content-based model and collaborative filtering model including neighborhood and matrix factorizaton methods. To evaluate the performance, the outputs of **SVD** and **SVD++** models are in turn used as the baseline estimate in (14). As shown in Table 9, substituting b_{ui} with the output of **SVD++** provides lower RMSE and MAE than of **SVD**. This is because originally the accuracy of **SVD++** is superior to the one of its counterpart. Moreover, combining the strengths of matrix factorization model with the model proposed in the previous section constitutes a hybrid recommendation system which outperforms each individual in

TABLE 9. Comparison between the proposed hybrid content-based and CF model and baseline models.

Model	RMSE	MAE	Time [s]
SVD++	0.7894	0.5992	124,224
kNN-Content^{AE}	0.7692	0.5841	1,092
I-RBM	0.7951	0.6065	96,455
I-AutoRec	0.7808	0.5931	69,860
kNN-Content^{AE}-SVD	0.7634	0.5796	15,412
kNN-Content^{AE}-SVD++	**0.7584**	**0.5732**	**124,856**

terms of the accuracy. More specifically, Table 9 shows that **kNN-Content^{AE}-SVD++** gains:

- 3.93%-lower RMSE and 4.34%-lower MAE than **SVD++**.
- 2.87%-lower RMSE and 3.36%-lower MAE than **I-AutoRec**.
- 1.40%-lower RMSE and 1.87%-lower MAE than **kNN-Content^{AE}**.

However, there is a trade-off between the accuracy and the computational complexity. The ultimate hybrid model makes better rating predictions than its separate components at the cost of requiring more time to learn from data and make recommendations. Indeed, the final rating is achieved after a consolidation stage of outputs from individual models.

VI. CONCLUSION

In this paper, we first introduced an NLP-based cleaning process to eliminate the redundancy and conflict from 1,128 original genome tags which helped generate a more precise description consisting of 1,044 new tags for each movie. Then in order to discover the latent characteristics under the genome tags and create a more concise representation, a 3-layer autoencoder was utilized to compress the newly generated tags into a 600-element vector. The new representation not only helped produce a 2.32% lower RMSE and a 2.24% lower MAE but speeded up the whole system by 1.28 times compared to the reference model using 1,044 genome tags. Finally, we proposed to integrate the strengths of the new representation for each movie and the common CF techniques into a unified framework which outperformed the state-of-the-art models by at least 2.87% and 3.36% in terms of RMSE and MAE, respectively. This improvement was achieved at the cost of increasing the computational complexity because the final rating was predicted using the outputs from individual models.

REFERENCES

[1] A. Rae, V. Murdock, A. Popescu, and H. Bouchard, "Mining the Web for points of interest," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2012, pp. 711–720.

[2] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2015, pp. 1–34.

[3] T. Chen, X. He, and M.-Y. Kan, "Context-aware image tweet modelling and recommendation," in *Proc. 24th ACM Multimedia Conf. (MM)*, 2016, pp. 1018–1027.

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[5] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 331–339.

[6] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting Web sites," *Mach. Learn.*, vol. 27, no. 3, pp. 313–331, 1997.

[7] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 73–105.

[8] X. Li, M. Cheung, and J. She, "Connection discovery using shared images by Gaussian relational topic model," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 931–936.

[9] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, and G. Semeraro, "Concept-based item representations for a cross-lingual content-based recommendation process," *Inf. Sci.*, vol. 374, pp. 15–31, Dec. 2016.

[10] D. Billsus and M. J. Pazzani, "Learning collaborative information filters," in *Proc. ICML*, vol. 98, 1998, pp. 46–54.

[11] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*. [Online]. Available: https://arxiv.org/abs/1205.2618

[13] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, Oct. 2009, Art. no. 421425.

[14] Y. Wang, J. Deng, J. Gao, and P. Zhang, "A hybrid user similarity model for collaborative filtering," *Inf. Sci.*, vols. 418–419, pp. 102–118, Dec. 2017.

[15] J. Bennett and S. Lanning, "The netflix prize," in *Proc. KDD Cup Workshop*, New York, NY, USA, 2007, p. 35.

[16] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 650–658.

[17] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 19–28.

[18] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 448–456.

[19] W.-J. Li, D.-Y. Yeung, and Z. Zhang, "Generalized latent factor models for social network analysis," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 1705–1710.

[20] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2015, pp. 1235–1244.

[21] T. Miranda, M. Claypool, A. Gokhale, T. Mir, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *Proc. ACM SIGIR Workshop Recommender Syst.*, 1999.

[22] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artif. Intell. Rev.*, vol. 13, nos. 5–6, pp. 393–408, Dec. 1999.

[23] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[24] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[25] P. N. Huu, V. Tran-Quang, and T. Miyoshi, "Energy threshold adaptation algorithms on image compression to prolong WSN lifetime," in *Proc. 7th Int. Symp. Wireless Commun. Syst.*, Sep. 2010, pp. 834–838.

[26] P. Nguyen Huu, V. Tran-Quang, and T. Miyoshi, "Video compression schemes using edge feature on wireless video sensor networks," *J. Electr. Comput. Eng.*, vol. 2012, Oct. 2012, Art. no. 421307.

[27] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 809–817.

[28] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, *arXiv:1404.2188*. [Online]. Available: http://arxiv.org/abs/1404.2188

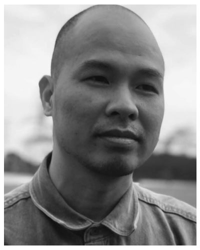
[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 791–798.
[31] K. Georgiev and P. Nakov, "A non-IID framework for collaborative filtering with restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1148–1156.
[32] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.
[33] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. ACM Int. Conf. Multimedia (MM)*, 2014, pp. 627–636.
[34] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for Top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2016, pp. 153–162.
[35] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.
[36] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, 2016.
[37] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 1999, pp. 230–237.
[38] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.
[39] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, vol. 1, 2001, pp. 285–295.
[40] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, p. 1, 2010.
[41] S. Funk. (2006). *Netflix Update: Try This at Home*. [Online]. Available: http://sifter.org/simon/journal/20061211.html
[42] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.
[43] T. N. Duong, V. D. Than, T. H. Tran, Q. H. Dang, D. M. Nguyen, and H. M. Pham, "An effective similarity measure for neighborhood-based collaborative filtering," in *Proc. 5th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Nov. 2018, pp. 250–254.
[44] T. N. Duong, V. D. Than, T. A. Vuong, T. H. Tran, Q. H. Dang, D. M. Nguyen, and H. M. Pham, "A novel hybrid recommendation system integrating content-based and rating information," in *Proc. Int. Conf. Netw.-Based Inf. Syst.* Cham, Switzerland: Springer, 2019, pp. 325–337.
[45] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[46] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

TAN NGHIA DUONG received the B.Sc. degree in electronics and telecommunications and the M.Sc. degree from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2011 and 2014, respectively, where he is currently pursuing the Ph.D. degree. Since 2012, he has been working as a Researcher and Lecturer with the Hanoi University of Science and Technology. His major research fields include signal processing for ultra wideband communication (UWB), fingerprint recognition, image processing, and recommendation systems.

TUAN ANH VUONG received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2019. He is currently pursuing the M.Sc. degree with Vietnam National University, Hanoi. His research interests include recommendation systems and natural language processing.

DUC MINH NGUYEN received the Ph.D. degree in electrical engineering from the University of Kaiserslautern, Germany, in 2009. He worked as a Scientific Staff Member of the University of Kaiserslautern. He is currently a Researcher and Lecturer with the School of Electronics and Telecommunications, Hanoi University of Science and Technology, Vietnam. His research activities involve digital hardware design, embedded system design, formal verification of digital design, embedded systems, and recommendation systems.



QUANG HIEU DANG was born in Hai Duong, Vietnam, in October 28, 1976. He received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 1999, and the M.Sc. degree from the Delft University of Technology, Delft, The Netherlands, in 2003, where he is currently pursuing the Ph.D. degree. From 1999 to 2001, he was a Research and Teaching Assistant at the Hanoi University of Science and Technology, where he has been working as a Researcher and Lecturer, since 2009. His research interests include signal processing for long-code WCDMA and ultra wideband communication (UWB), image processing, and recommendation systems.