Tag-based embedding representations in neural collaborative filtering approaches

Tahar-Rafik Boudiba[1,2], Taoufiq Dkaki[1]

[1]IRIS/IRIT, UMR 5505 CNRS, 118 Route de Narbonne, F-31062, TOULOUSE CEDEX 9, France
[2]ADBI Accelarator Data & Business Intelligence, 8 rue rossini 75009 Paris, France

Abstract

Learning user-item interactions in collaborative systems have become a promising method to improve the performance of collaborative filtering approaches. In such systems, contents surrounding users and items, particularly user tags, have a key role since they are leveraged with collaborative filtering approaches. Tags are commonly represented using the bag of words paradigm, although it is subject to ambiguity due principally to the poor semantic relation between tags. Recent methods suggest the use of deep neuronal architectures as they attempt to learn semantic and contextual word representations. On this basis, we have addressed how to integrate semantically such content into different neural collaborative filtering models for rating prediction. Based on effective models initially developed to learn user-item interaction, in this paper, we have extended different neural collaborative filtering models for rating prediction to evaluate the impact of using static or contextualized word embeddings within a neural collaborative filtering strategy. Moreover, the presented models use dense tag-based user and item representations extracted from pre-trained static Word2vec and contextual BERT. In addition, the paper emphasizes the impact of using contextualized tag embedding neighbors in a neural graph collaborative filtering approach that learns an aggregated function. Finally, to determine whether the use of different neural architectures can influence the recommendation quality, we adapt neural architectures, including three popular end-to-end learning models that are an MLP an autoencoder, and a Graph Neural Network. We evaluated and compared all the models with recent baselines on several MovieLens datasets.

Keywords

Learning representation, folksonomies, deep learning, word embedding, social tagging,

1. Introduction

Deep learning (DL) techniques are the milestones of several recent recommendation engines. Platforms such as Facebook[1] and Pinterest[2] have already shared their experience in using DL for recommender systems (RS). In such platforms, Collaborative Filtering (CF) approaches are mainly exploited. Such methods enable the users to get recommendations on favourite items. When such methods are put into practice in RS, it implies being able to predict how users will rate a particular item. Classical CF approaches are based either on Matrix Factorization (MF) techniques or on simple user-item vector similarity methods. However, these models

CIRCLE (Joint Conference of the Information Retrieval Communities in Europe) 2022
Tahar-rafik.Boudiba@irit.fr (T. Boudiba); Taoufiq.Dkaki@irit.fr (T. Dkaki)
0000-0002-0877-7063 (T. Boudiba); 0000-0001-7116-9338 (T. Dkaki)

[1]https://www.facebook.com/
[2]https://www.pinterest.com/

share the property of being essentially linear since they combine user and item latent factors linearly. In contrast, DL models for RS have the main property of learning multiple level of representation and hence have enabled the deep integration of several type of content. As result, recent neural collaborative filtering approaches capture more complex user-item interactions and enable high-level abstractions for content description. Such content often makes reference to user's tags since they are commonly used to describe items and users' profiles using the bag of words representation. Although such representations commonly appearing as one-hot vectors are efficient for computing user-item similarity, many problems such as ambiguity and vocabulary mismatch have been raised [1]. In this sens, common NLP techniques suggest the use of dense representations in the forme of eitheir user or item agregated semantic embedding vectors extracted from pre-trained Word2vec neural language model [2, 3]. However, how to include efficiently such embedding vectors at the top layer of a neural CF architecture? A design choice is to combine the two embedding vectors, then feed them through multiple fully connected layers to get the likelihood that a user interacts with an item. In that way, multiplying the embedding vectors element-wise with each other or simply concatenating them might be a raisonnable technique to integrate both user and item dense representations in a neural CF model. Some works have discussed text embedding aggregation techniques [4] others have suggested the concatenation of mean Word Embedding since they compute word average embedding representations [3]. Recent neural approaches for recommendation consider in addition other relationships such as neighborhood proximity among graph-based approaches. Such approaches have been proposed to explore multi-layer neighbor embedding representations. Since these embeddings are integrated with neural CF architectures this has resulted in Neural Graph CF (NGCF) approaches [5]. In this paper, we have considered tag embeddings as the starting point for integrating explicitly a tag-based vocabulary within neural collaborative filtering models. However, such initiative raises some research issues, such as determining the most efficient neural architecture to use or defining the best tag embedding representations. At this end, we handle dense tag-based representations that we exploit within effective neural CF models for rating prediction. We have developed several neural models that combine neural CF with tagging information integrated into a training process. For this purpose, we handled word vector representation to include more valuable tag' semantic and so to enhance neural CF models ability to generalize. We compared different tag embedding representations from pre-trained static (Word2vec) and contextual BERT models. Furthermore, we evaluated the impact of using such tag embeddings through several neuronal model's architecture that is an MLP, an autoencoder and a graph-based neural collaborative architecture. We provided empirical results from MovieLens Dataset 10 M, 20M et 25M. The main contributions of this paper are summarized as follows:

- Integrate efficiently tag-embedding representations into several neural CF models.
- Evaluate the impact of static/contextual embedding representations and comparing model architecture.
- Evaluate impact of multi-layer neighbor static/contextual embedding representations to be exploited in a neural graph CF model.
- Extensive series of experiments on real data from several MovieLens data sets.

The remaining of the paper is organized as follows. The next section presents some background and reviews recent research works related to content-based recommendation using neuronal networks and word vector representation. We gathered works that describe neural approaches from a collaborative filtering point of view, specifying the most used neural architectures. Section 3 highlights the basis of our proposed models. Section 4 details datasets, evaluation metrics, and experimental settings. Section 5 gives the evaluation results and discusses performance comparison with baselines. Following these sections, we will draw our conclusion in the final section.

2. Background and related works

DL methods have made breakthroughs in data representation learning from various data sources. As result, recent neural recommendation models have been able to handle learning representations of user preferences, item features and textual interactions [6, 1]. Yet, neural recommendation models attempt to introduce in addition, tag semantic-aware representations based on distributional tag semantic used as features [6]. In this area, Musto et al., [7] exploit Word2vec approach to learn a low dimensional vector space word representation and exploited it to represent both items and user profiles in a recommendation scenario. Zhang et al., [8] proposed to integrate traditional matrix factorization with Word2vec for user profiling and rating prediction. Liang et al., [9] exploited pre-trained word embeddings from *Word2vec* to represent user tags and construct item and user profiles based on the items' tags set and users' tagging behaviors. They use deep neural networks (DNNs) and recurrent neural networks (RNNs) to extract the latent features of items and users to predict ratings. Moreover, TagEmbedSVD [10] uses pre-trained word embedding from *Word2vec* for tags to enhance personalized recommendations that are integrated to an SVD model in the context of cross-domain CF. Other works [11, 1] take advantage of network embedding techniques to propose embedding-based recommendation models that exploit CF approaches. Along with learning content representation for recommendation, exploiting rating patterns often require the use of a neural network-based embedding model that is first pre-trained. Features are extracted and integrated into a CF model by fusing those features with latent factors thanks to non-linear transformations that better leverage abstract content representations and so perform higher quality recommendations. Since pre-training word embedding from large-scale corpus became widely used in different information retrieval tasks, it was also exploited to generate recommendations by ranking user-item matrix from users' similar tags vocabulary. Models such as Word2vec [12] or GloVe [13] for instance learned meaningful user tag representations by modeling tag co-occurrences. However, these methods don't consider the deep contextual information that some single content words may suffer. Moreover, they do not handle unknown words. In contrast, contextualized word representations such as BERT [14], have been proposed to overcome the lack of static word embeddings, since it was shown that such contextual neural language model improves the performance of many downstream tasks. Yet, graph-based neuronal approaches[15, 16, 17] have considered heterogeneous graphs as they try to overcome the missing of relationship modeling in features-based neural recommendation models. Such approaches have been proposed to explore multi-layer neighbor embedding representations [18]. Neural graph network models

consider content information features extracted from either graph properties [19] or learned from node embedding representations [20]. Particularly, Neural Graph Collaborative Filtering (NGCF) approaches exploit feature representations of the user-item graph structure by propagating either user-based or items-based content embeddings on it [21]. Such process is often the result of learning aggregation functions that allow deep-based relationship modeling among both user-item interaction and content features. In this way, Graph Convolutional Networks (GCNs) have also been exploited through learning aggregator functions which required additional layers to obtain a convolution neighborhood aggregation by neighborhood's embeddings at these layers [22]. As result, deep semantic representations are extracted using embeddings propagation on user-item graph structure. An instance of such method is used in Ying et al,. [23] since it employs multiple graph convolution layers on an item-item graph in Pinterest [3] image recommendation.

In the following, we introduce some recommendation models of the literature that have handled neural CF approaches [24, 25, 26]. Those models resolved user rating prediction. Some of them have been adapted to include tagging content [27, 28, 29], they are mostly composite through which multiple neural building modules compose a single distinguishable function that is trained end-to-end. Here, we introduce some summary definitions related to tagging that will allow us to address later most common architectures and topologies giving recommendation strategies for each of them. A folksonomy F can be defined as a 4-tuple $F = (U, T, I, A)$, where U is the set of users annotating the set of items $I, U = \{u_1, u_2, ...u_M\}$ where each u_i is a user. T is the set of tags that includes the vocabulary expressed by the folksonomy. I is the set of tagged items by user $I = \{i_1, i_2...i_N\}$. $A = \{u_m, t_k, i_j\} \in U \times T \times I$ is the set of annotations of each tag t_k to an item i_j by user u_m. We have also considered R as the set of user ratings $r_{u,i}$.

2.1. MLP-based neural collaborative filtering for Recommendation

Approaches of neural collaborative filtering (NCF) for rating prediction often involves dealing with binary property of implicit data. Some works [30, 31, 26] have in addition discussed the choice of the neural architecture to be implemented. A possible instance of the neural CF approach can be formulated using a multi-layer perceptron (MLP). As addressed in [30] the input layer (the embedding layer) is a fully connected layer that maps the sparse representations to dense feature vectors. It consists of two feature vectors $v^f_{(u)}$ and $v^f_{(i)}$ that describe user $v^U_{(u)}$ and item $v^I_{(i)}$ represented initially through one-hot encoding. The obtained user (item) embedding can be seen as the latent vector for user (item). The user embedding and item embedding are then fed into neural CF layers to map the latent vectors to prediction scores. Final output layer is the predicted score $\hat{r}_{u,i}$, and training is performed by minimizing the point wise loss between $\hat{r}_{u,i}$ and its target value $r_{u,i}$. *NCF* predictive model can be formulated as:

$$\hat{r}_{u,i} = \mathbf{MLP}(P_u^T . v^f_{(u)}, Q_i^T . v^f_{(i)} | P_u, Q_i, \Gamma_{\mathbf{MLP}}) \quad (1)$$

$P_u \in \mathbb{R}^{M \times K}$ and $Q_i \in \mathbb{R}^{N \times K}$ are latent factor matrix for users and items respectively. Γ_{MLP} denotes the model parameters of the interaction function that is defined as a multi-layer

[3] https://www.pinterest.fr/

neural network.

2.2. Autoencoder-Based collaborative filtering for Recommendation

Another way to consider neural CF is to approach user-item rating as a matrix $X \in R^{m \times n}$ with partially observable row vectors that form a user $u \in$ the set of users $U = \{1...m\}$ given by the set of user ratings $r_{(u)} = \{X_{u1}...X_{um}\} \in R$ and column vectors from the set of items $i \in I = \{1...n\}$ also given by their corresponding ratings $r_{(i)} = \{X_{i1}...X_{in}\}$. An efficient neural method to encode each partially observed vector into law-dimensional latent space is to handle an autoencoder architecture as suggested in [25] that will reconstruct the output space to predict missing ratings for recommendation [25, 32, 24]. Given a set of rating vectors $r_{(u)}$ and $r_{(i)} \in \mathbb{R}^d$, the autoencoder solves:

$$min_\theta \sum_{r \in R} = ||r - h(r;\theta)||^2 \tag{2}$$

Where $h(r;\theta)$ is the reconstruction of input $r \in \mathbb{R}^d$ that is defined as:

$$h(r;\theta) = f(W.g(Vr + \mu) + b) \tag{3}$$

$f(.)$ and $g(.)$ are activation functions associated to the encoder and decoder respectively and θ gather model parameters; $W \in \mathbb{R}^{d \times k}$ and $V \in \mathbb{R}^{k \times d}$ are weight matrices and $\mu \in \mathbb{R}^k$, $b \in \mathbb{R}^N$ biases. In an item-based recommendation perspective, the autoencoder applies $r_{(i)}$ as the set of input vectors. Weights associated to those vectors are updating during backpropagation.

2.3. Neural Graph Collaborative Filtering for Recommendation

NGCF approaches are particular in the sens that they exploit embeddings of users and items represented initially as a graph structure. Most of them adopt a user-item bipartite graph of as it much represents user-item interactions [15, 20, 16]. Promising recent methods suggest learning user and item representations from their bipartite associated graph by stacking multiple embedding propagation layers to allow high-order connectivity from user-item interactions [21]. Other works [15] learn aggregator functions that induce the embedding of a new node given its features and neighborhood. In the following we formalized what can be associated to a neural graph-based collaborative filtering approach for user rating prediction based on multiple embedding aggregation layers. This neural graph-oriented approach is designed to exploit node embeddings from neighborhood aggregation. Given a bipartite weighted graph of user-item $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A, \mathcal{X})$, with $\mathcal{V} = \{\mathcal{V}_u \cup \mathcal{V}_i\}$, $\mathcal{E}$ denotes the set of undirected weighted edges representing user ratings, A is the adjacency matrix and $\mathcal{X} \in \mathbb{R}^{m \times n}$ is defined as the node feature matrix.

Let $h_v^0 = x_v^u$ with $v \in \mathcal{V}_u$ be the user node feature at the 0th layer. Then, At the k-th layer :

$$h_v^k = \delta(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + A_k h_v^{k-1}) \tag{4}$$

h_v^{k-1} is the embedding of user node $v \in \mathcal{V}_u$ from previous layer. $|N(v)|$ is the number of the neighbors of node v. The sum expressed in the equation enables aggregate neighboring features of node v from previous layer. δ is the activation function (*Tanh*) that enables non-linearity. W_k and A_k are trainable parameters. The final embedding after *K* layers ($k \in \{1...K\}$) is extracted from the output layer: $z_v^u = h_v^K$ after *K* layers. This can be expressed as a matrix multiplication form for the whole graph as:

$$H^{l+1} = \delta(H^l W_0^l + \tilde{A} H^l W_1^l) \tag{5}$$

In such a way that $\tilde{A} = D^{-1/2} A D^{-1/2}$ with A represents adjacency matrix and D represents the degree matrix. Thereafter, after applying similar process to item nodes embeddings to get z_v^i with $v \in \mathcal{V}_i$, one way is to employ a concatenated operator $\oplus$ on both user and item final embeddings to obtain $z_e^{u \oplus i} = z_v^u \oplus z_v^i$ that represents the edge embedding $e_{u,i}$ between a user node v_u and item node v_i, with $e_{u,i} = [v_u, v_i]$. These edge embeddings are passed through a link regression layer to obtain predicted user-item ratings. The model is trained end-to-end by minimizing a regression loss function (RMSE or root mean square error between predicted and true ratings) using stochastic gradient descent (SGD) updates of the model parameters, with minibatches of user-item training edges fed into the model.

3. Overview of the proposed models

In this section, we introduce our tag-aware neural models for recommendation. More explicitly, we integrate tag-based embeddings into CF neural architectures, namely a Multilayer perceptron, an autoencoder and a neural graph based model. More explicitly, to integrate side information into predictive neural models a naive approach consists of appending additional user/item bias to the rating prediction. We estimate that computing those biases can be handled either by hand-crafted engineering or by implementing an appropriate CF strategy. A simple Neural collaborating filtering framework architecture implies considering the input layer(embedding layer) as a fully connected layer that projects sparse representation of users and items to dense vectors. To integrate explicitly tags vocabulary in a neural model for rating prediction, we have made use of feature vectors that we have considered as tag vector representations sharing a common embedding space using projection matrices. The obtained user (item) embedding can be seen as the latent vector for the user (item) in the tag latent space. Feature vectors $v_{(u)}^f$ and $v_{(i)}^f$ are reconsidered since we have projected tag representations into lower dimension using projected matrices $\mathbf{E}$ and $\mathbf{F}$. Consequently, tag-based vector representation is expressed as a user feature vector $v_{(\tilde{u})}^{\tilde{f}}$:

$$v_{(\tilde{u})}^{\tilde{f}} = \frac{1}{|T_u|} \sum_{t_k \in T_u} E(t_k) \tag{6}$$

Such as $t_k \in \mathbb{R}^c$ is the embedding vector associated with tag *k*, and *c* denotes the embedding dimension. E denotes the projection matrix with $E \in \mathbb{R}^{d \times c}$.

Similarly, if F denotes the projection matrix with $F \in \mathbb{R}^{d \times c}$, then the item feature vector $v_{(\tilde{i})}^{\tilde{f}}$ is expressed as:

$$v_{(\tilde{i})}^{\tilde{f}} = \frac{1}{|T_i|} \sum_{t_k \in T_i} F(t_k) \tag{7}$$

We denoted T_u the set of tags of a user u and T_i as the set of related tags describing a particular item. Moreover, we have obtained embeddings for tags from *Word2vec* and *BERT* pre-trained neural model by handling projection matrices $\mathbf{E}$ and $\mathbf{F} \in \mathbb{R}^{d \times c}$.

3.1. CF-based MLP model

Extended tag-based NCF predictive model can be reformulated relying on the previous NCF model that has been described in section 3.1 equation (1) as:

$$\hat{r}_{u,i} = \mathbf{MLP}(\, v_{(\tilde{u})}^{\tilde{f}},\, v_{(\tilde{i})}^{\tilde{f}}, \theta_{\mathbf{MLP}}) \tag{8}$$

The user and item embeddings can be fed into a multi-layer neural model.

Where, $\hat{r}(u, i)$ is the rating score for a user on an item. **Figure** $1_{(\mathcal{C})}$ details an instance of the model. Prediction Pipeline exploits user and item vectors extracted from dense space representation (***Figure*** $1_{(\mathcal{A})}$), hidden layers are added to learn interactions between user and item latent features, a regressor at the last hidden layer is set to produce the final rating. (***Figure*** $1_{(\mathcal{A})}$) is a dynamic module in which dense representations are computed through inner product of user and items embedding' representations. Tag embedding representations are extracted from neural pre-trained language model (***Figure*** $1_{(\mathcal{E})}$).

3.2. CF-based Autoencoder model

Following the autoencoder paradigm, instead of encoding user vectors containing user ratings to be predicted like in Autorec [25], we have extended a multilayered autoencoder architecture to integrate element wise product of pre-trained tag-based embeddings. Such embeddings are concatenated with the user rating representations and are projected on a dimensional latent (hidden) space. As such, user' rating $r(u_m, i_l)$ of a particular user is reconstructed using an objective function θ that minimizes :

$$\sum ||r(u_m, i_l) \oplus (v_{(\tilde{u})}^{\tilde{f}} \otimes v_{(\tilde{i})}^{\tilde{f}}) - h(r(u_m, i_l) \oplus (v_{(\tilde{u})}^{\tilde{f}} \otimes v_{(\tilde{i})}^{\tilde{f}}); \theta)||^2 \tag{9}$$

Where $(r(u_m, i_l), \theta)$ is the reconstruction of the input $r(u_m, i_l) \in \mathbb{R}^d$. The operator $\otimes$ denotes element-wise multiplication between user and item feature vectors. The operator $\oplus$ denotes a concatenation operator. $tanh$ is the selected activation function. **Figure** $1_{(\mathcal{B})}$ presents a detailed instance of the model. Prediction Pipeline exploits user and item vectors extracted from dense space representation. Such representations are concatenated with user rating and fed as input of the autoencoder model. Layers are added to learn interactions between user and item latent features to be compressed in a dense space. User's ratings reconstruction from the dense space produce the final rating.

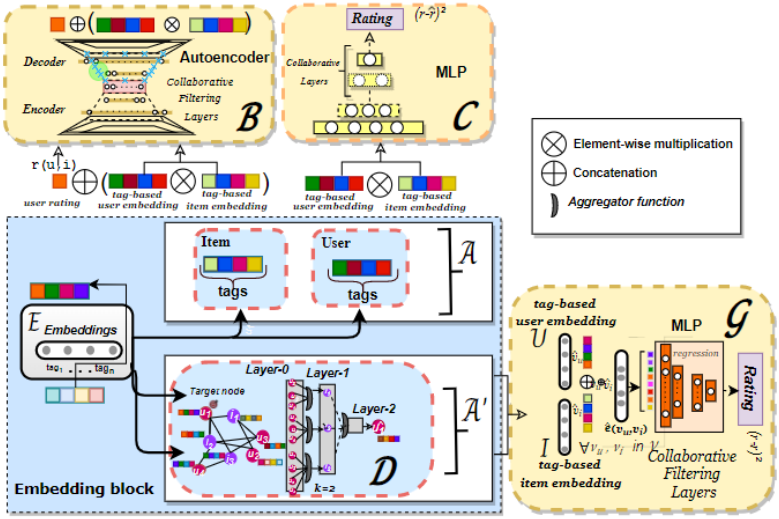

Figure 1: Extended NCF based on an MLP (on the right) and an Autoencoder (on the left), Graph-based NCF architecture based on tag feature embeddings and aggregator functions

3.3. Neural graph CF-based model

As part of collaborative filtering approaches, neural graph- based networks consider for the most [20, 19, 15] bipartite graphs of users and items in a recommendation context, where edges represent the rating interactions between the users and the items. From the bipartite graph G defined in section 2.1.3 where nodes' classes are derived from the set of user nodes $\mathcal{V}_u$ and the set of item nodes $\mathcal{V}_i$ respectively. Each edge corresponds to whatever user's rates an item. Each edge $e_{u,i} \in \mathcal{E}$ is associated to a value $r_{(u,i)} \in \{0, 1\}$.In order to learn the topological structure of each class of node neighborhoods, the idea is to aggregate feature information from node's local neighborhood [15], however in this paper we handled node's features from pre-trained static and contextual tag embeddings model. Users' nodes features are taken from mean average users' tags embedding vectors, equivalently items' nodes features are represented throws the mean average of their tag embeddings vectors. We have previously explored a simple neighborhood aggregation process in section 2.0.3. By defining a neighborhood function $N(v)$, that is set to a fixed-size (in our experiments *K=2*), the bipartite graph is sampled as the model learn a function that generates aggregates from tag-based textual feature node neighbors. This method can be generalized by applying different aggregation methods to nodes $\in G$ by concatenating the features with the nodes itself. For this purpose, we have associated each node $v \in \{\mathcal{V}_u \cup \mathcal{V}_i\}$ to features from word vector representation by joining tag-based vector representation $v^{\tilde{f}}_{(\tilde{u})}$ and $v^{\tilde{f}}_{(\tilde{i})}$ (***Figure*** $1_{(\mathcal{G})}$). We have designed a mean aggregation function that is commonly used since it imply element wise mean of the feature vectors in h_u^{k-1}. We have also designed a convolution aggregator function that we have detailed next.

3.3.1. Mean aggregator function

Since the rating interactions between users and items are represented as a bipartite graph $G = (U, V, E)$, $\mathcal{V}_u$ and $\mathcal{V}_i$ corresponds respectively to users and items sets. Thus, aggregation mean tag embedding features from the neighbors of the node $v \in \{\mathcal{V}_u \cup \mathcal{V}_i\}$ is processed given the following update rule (***Figure1*** $\mathcal{D}_{(\mathcal{A}')}$):

$$h^k_{N(v)} = \frac{1}{|N(v)|} D_p[h^{k-1}_v]$$

We give the forward pass through layer k as follows:

$$h^k_v = \delta(concat[W^k_I D_p[h^{k-1}_v], W^k_{Neighbor} h^k_N(v)] + b^k)$$

Where, h^k_v is the output node v at layer k,
W^k_I and $W^k_{Neighbor}$ are trainable parameters, b^k is an optional bias, d^k is node feature dimensionality at layer k, δ is a non linear activation function (*Tanh*), D_p is a random dropout with probability p applied to its argument vector used to reduce model's over-fitting. $N(v)$ represent the neighborhood of a node $v \in \{\mathcal{V}_u \cup \mathcal{V}_i\}$. The number of trainable parameters in layer k for the mean aggregator is $d_k.d_{k-1} + d_k$.

3.3.2. Convolutional aggregator function

To generalize the collaborative filtering process from a graph convolutional network perspective, we adopted a GCN aggregator [15] (***Figure1*** $\mathcal{D}_{(\mathcal{A}')}$), that concatenates nodes from the previous layer representation h^{k-1}_v with the aggregated neighborhood vectors $h^k_{N(v)}$. Features are updated given the following equation:

$$h^k_{N(v)} = \frac{1}{|N(v)| + 1}(h^{k-1}_v + \sum_{v \in N(v)} h^{k-1}_v) \tag{10}$$

Forward pass through layer k is defined as:

$$h^k_v = \delta(W^k.h^k_{N(v)} + b^k) \tag{11}$$

Where, W^k, is a trainable weight matrix, shared between all nodes $v \in \{\mathcal{V}_u \cup \mathcal{V}_i\}$. The size of W^k is given as $d_k \times d_{k-1}$. The number of trainable parameters in layer k for the *GCN* aggregator is $d_k.d_{k-1} + d_k$.

4. Experiments

In this section, we have conducted experiments intending to answer the following research questions:

RQ1: Are tag-based contextual embeddings efficient representations to be used in a neural CF model compared to static tag-based embedding representations?

RQ2: Which extended neural collaborative architecture perform significant improvement and ranking quality for a rating prediction task?

From there, an underlying research question can be derived, it concerns the various methods used for aggregating tag embeddings. Assuming that, the methods used for aggregating tag embeddings may affect the performance of recommendation models.

RQ3: Are contextual neural graph embeddings more efficient representations to be used in a neural collaborative filtering architecture ? regarding such process, which aggregator function should leads to better recommendation performance? A mean aggregator function? a convolutional aggregator function?

4.1. Experimental Settings

1. **Datasets**: The data sets describe 5-stars ratings and free-text tagging from *MovieLens*, a movie recommendation service. We extracted user annotations from the ML-10M, ML-20M, and ML-25M data sets. Only users that have annotated and rated at least 20 movies were selected. We observed from **Table** 1 an unequal distribution of user rating classes, because of users trend scoring items with good rating values. This can lose models capacity to generalize. To overcome, we over-sample minority classes [33] by duplicating samples from the minority class and adding them to the training data.

2. **Hyper-parameters**: After splitting the data in each dataset into random 90%, 10% training and testing sets, we hold 10% of the training set for hyper-parameters tuning. Then, we conducted 5 cross-fold validation strategy in each dataset and averaged RMSE measure. We have applied a grid search for hyper-parameters tuning such as the learning rate that we tuned among values $\in \{0.0001, 0.0005, 0.001, 0.005\}$, latent dimensions $\in \{100, 200, 300, 400, 500, 1000\}$ for both autoencoder and MLP architecture. We handled the Neural Collaborative Autoencoder with a default rating of 2.5 for testing set without training observations. Graph neuronal and convolutional models handled same dataset, except that models derived from these approaches handle edges prediction throw bipartite graph samples. We tuned the dropout ratio [4] from values $\in \{0.0, 0.1, , 0.8\}$, we have also defined the neighbor nodes embeddings features at a particular layer of 2. The models were optimized thanks to the well known Adam optimizer.

3. **Evaluation Metrics**: We have evaluated rating prediction using two metrics: Mean Absolute Error (MAE) and Root Mean Square Error(RMSE). Both of them are widely used for rating prediction in recommended systems. Given a predicted rating $\hat{r}_{u,i}$ and a ground-truth rating $r_{u,i}$ from the user u for item i, the RMSE is computed as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(r_{u,i} - \hat{r}_{u,i})^2} \tag{12}$$

   Where N indicates the number of ratings between users and items.

[4]The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent over-fitting

MAE is computed as follows:

$$MAE = \frac{1}{N} \sum_{u,i} |r_{u,i} - \hat{r}_{u,i}| \tag{13}$$

Indeed, we have also evaluated ranking accuracy using NDCG (Normalized Discounted Cumulative Gain [34]) at 10. For this purpose, we assumed rating values at 5 as being a good appreciation of a user regarding a movie. In contrast, rating values under 3 are considered as bad. Hence, the rating value of each movie is used as a gained value for its ranked position in the result. The gain is summed from the ranked position from 1 to n. To compute $NDCG$, relevance scores are set to six(5) points scale from 1 to 5 and denotes the relevance score from low to strong relevance. We set the Ideal DCG for user movies ranked in decreasing order of their ratings. NDCG values presented further are averaged over user testing set.

4.2. Tag-based embedding representations

We have considered tag-based embeddings thanks to word vector representations. We have extracted such tag-based embedding representations from pre-trained neural language models. Owing to the users' writing discrepancy, users' tags semantic meaning is often ambiguous. Tags can be composed of several words and may contain subjective expressions. They can also be unique words which can occasionally lead to a lack of context. That makes it difficult to integrate tags explicitly in an effective neural CF architecture. Our main objective is to map users, items and their tags' interaction in the same latent space. Rather to exploit straightly dimensional latent space representations of users and items like in most neural collaborative approaches [30, 35], we propose to project first both users' and items' representations into a dense tag space representation. Both previous neural approaches are somehow representative of our objective since they are from CF. We assume that users and items are represented by their corresponding tags. Particularly, they are represented from the aggregate average of their tag embedding representations.

1. Static *Word2vec* tag-based embdddings: We have handled static tag-based embedding vectors from Word2vec. We have exploited pre-trained vectors trained on part of Google News dataset (about 100 billion words) and have extracted user's tags embedding by associating them to a vector of a well known fixed size for each tag. However, we found that some tags were out of tag vocabulary, since those user tags represent respectively 8%, 5%, 5% of our Movielens Datasets 10M, 20M and 25 millions ratings. We fixed this issue by initiate those samples with random vector values. The inability to handle unknown or out-of-vocabulary words is one of limitation encountered when using such pre-trained model. Finally, each set of tags per user is represented through a multidimensional vector of $dim = 300$.
2. Contextualized *BERT* tag-based embdddings: We have addressed extracting contextualised embeddings from *BERT* neural language model. For this purpose, we have assumed that the fist token which is '[CLS]' that captures the context is treated as sentence embeddings [36]. The word embedding sequence corresponding to each set of tags is entered into

the pre-trained model. We have then handled the activation from the last layers of BERT model since the features associated with the activation in these layers are far more complex and include more contextual information. These contextual embeddings are used as input to our proposed models. Thus, each set of tags per user is represented through a multidimensional embedding vector of $dim = 768$. We have implemented the pre-trained bert-base model [5] (12 blocks of hidden dimension 768, 12 heads for attention) and defining the '[CLS]' which indicates the beginning of a sequence as well as the '[SEP]' that we used as a separation between two tags of a same sequence.

Collection	***10M***	***20M***	***25M***
Number of users	71567	*138000*	*162541*
Number of movies	10681	*27000*	*62423*
TAS(Tag assignment)	95580	*465000*	*1093360*
Ratings	10000054	*2000000*	*25000095*
Nodes	7114	20555	35363
Edges	24564	126080	210725
Period	*Dec-2015*	*Oct-2016*	*Nov-2019*

Table 1
Statistical details of the 10M, 20M and 25M collections from MovieLens

5. Evaluation and Performance comparison

First, to solve the **RQ1**, we extended neural models [30, 25] by handling static and contextual tag-based embedding representations. We compared those models with recent neural models from CF that we set as baselines. We evaluated rating score accuracy using RMSE (Root Mean Square Error) and MAE (Mean Absolut Error). Then, to address **RQ2**, we have implemented an MLP and an autoencoder-based CF architecture then, we compared the performance of each neural model according to tag-based embedding representations with which such models were integrated. Moreover, ranking accuracy metric was carried out among the different neural models using NDCG (Normalized Discount Cumulative Gain) at 10. Finally, to answer **RQ3**, we managed to exploit user/item based tag embeddings thanks to an aggregate function that is learned from training samples of user-item graphs. Such function operates either by performing element wise multiplication between the tag embedding neighbor vectors of a given node or by concatenating tag embedding vectors with their tag embedding neighbor vectors to get the embedding of that node.

We have detailed bellow all the models that are included in the neural models Comparative study.

- **Neural GMF-MLP**[30]: Is a neural CF approach that exploits a multi-layer perceptron (MLP) to learn the user–item interaction function. The bottom input layer consists of two

[5]BERT was pre-trained on a corpus composed of 11,038 unpublished books belonging to 16 different domains and 2,500 million words from English Wikipedia text passages

vectors that describe user *u* and item *i* in a binarized sparse vector (one-hot encoding), such model employ only the identity of a user and an item as input feature.

- **Neural CF-MLP^{++}**: Is an extension of Neural CF-MLP, the model integrates in the bottom input layer two feature vectors that are described as tag embedding features of users and items. These features are extracted from word vector representation. User and item feature vectors are extracted from tag-based embeddings, with 300-dimensional word vectors from pre-trained Word2vec model Neural CF-MLP$^{++}{}_{Word2vec}$ and Neural CF-MLP$^{++}{}_{BERT}$ that exploits a 768-dimensional word vectors from pre-trained BERT model.
- **U-Autorec** [25] U-AutoRec is a neural CF framework for rating prediction that exploits an autoencoder architecture. It takes user vectors as input and reconstructs them in the output layer. The values in the reconstructed vectors are the predicted value of the corresponding position.
- **CF-Autoencoder^{++}** Our autoencoder-based neural collaborative approach that integrates as input tag embedding features by performing element-wise multiplication on their word vector representations and do concatenate such representations with user/item rating vectors to get the reconstructed ratings. We have termed the autoencoder-based model using static tag vector representations as CF-Autoencoder $^{++}{}_{Word2vec}$ meanwhile CF-Autoencoder$^{++}{}_{BERT}$ stands for autoencoder-based model using contextual tag vectors.
- **CF-GNN$^{++}_{Mean\ Agg_{(k=2)}}$** Our NGCF tag-based predictive model that generates node embeddings by sampling and aggregating features (tag embeddings) from nodes local neighborhood using a mean aggregation function that operates at neighborhood of $k = 2$. We distinguish between the NGCF model that handles features extracted from tag-based embeddings using 300-dimensional tag vectors extracted from pre-trained Word2vec model and that we term CF-GNN$_{Mean\ Agg_{(k=2)}Word2vec}$ and CF-GNN$_{Mean\ Agg_{(k=2)}Bert}$ that exploits 768-dimensional tag vectors from pre-trained BERT model.
- **CF-GCN$^{++}_{Mean\ Agg_{(k=2)}}$** We do consider this NGCF model as being convolutional since it learn convolutional aggregator function that concatenate the node's previous layers representations with the aggregated neighborhood vectors. We differentiate between the model that handles features extracted from tag-based static embeddings with 300-dimensional tag vectors from pre-trained Word2vec model and that we term CF-GCN$_{Mean\ Agg_{(k=2)}Word2vec}$ and CF-GCN$_{Mean\ Agg_{(k=2)}Bert}$ that exploit 768-dimensional tag vectors from pre-trained BERT model.
- **Hinsage** [15] is a model that employs a technique for computing node representations in an inductive way. This method operates by sampling a fixed-size neighborhood of each user/item node and then performing a specific aggregator over all the sampled neighbors' feature vectors. This model learns general-purpose node embeddings that use the graph structure and particularly node features. It was evaluated for a rating prediction task using demographic users information (no tags information).
- **TRSDL** [9]: Tag-aware recommender system that uses a deep neural networks (DNNs) and recurrent networks (RNNs) to extract latent features of both users and items. In their model Liang et al., [9] use Word2Vec for mapping user tags to k-dimensional dense

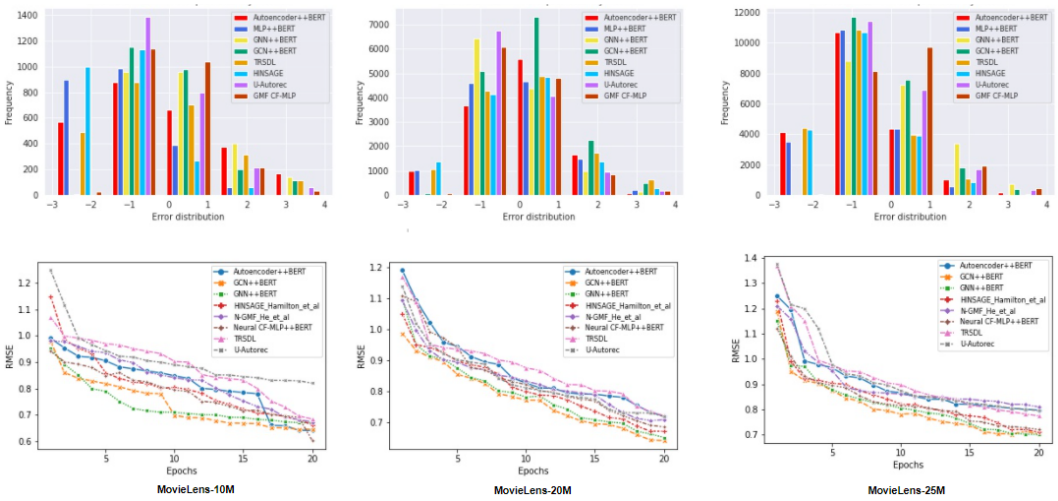

Figure 2: At the top of this figure, we presented each neural model's error distribution. At the bottom, we gave model's validation scores after 20 epochs

vectors in order to represent tags with word embeddings. Their model have the ability to construct item and user profiles based on the item's tags and the user's tagging behaviors. They then utilizes deep neural networks (DNNs) and recurrent neural networks (RNNs) to extract the latent features of the item and the user, respectively.

Models	Evaluation measures								
	ML-10M			ML-20M			ML-25M		
	MAE	RMSE	ndcg@10	MAE	RMSE	ndcg@10	MAE	RMSE	ndcg@10
Neural CF-MLP$^{++}_{W2v}$	0.77	0.98	0.43	0.88	0.96	0.381	0.84	1.01	0.42
Neural CF-MLP$^{++}_{Bert}$	**0.72**	0.93	0.46	0.791	0.86	0.42	0.791	0.83	0.46
CF-Autoencoder$^{++}_{W2v}$	0.83	1.1	0.411	0.85	0.97	0.39	0.80	1.02	0.42
CF-Autoencoder$^{++}_{Bert}$	0.76	0.96	0.42	0.811	0.89	0.44	0.798	0.865	0.445
U-Autorec [25]	0.82	1.09	0.38	0.84	1.07	0.37	0.81	1.01	0.40
Neural CF-MLP[30]	0.73	0.98	0.44	0.89	1.025	0.39	0.87	0.92	0.43
CF-GNN$^{++}_{Mean\ Agg_{(k=2)}W2v}$	0.88	1.10	0.47	0.80	1.02	0.49	0.82	1.04	0.44
CF-GNN$^{++}_{Mean\ Agg_{(k=2)}Bert}$	0.774	0.89	0.451	0.78	0.85	0.441	0.772	**0.799**	0.471
CF-GCN$^{++}_{conv\ Agg_{(k=2)}W2v}$	0.798	0.821	0.47	**0.74**	0.838	**0.464**	0.79	0.801	0.465
CF-GCN$^{++}_{conv\ Agg_{(k=2)}Bert}$	**0.715**	**0.791**	**0.48**	**0.723**	**0.782**	**0.47**	**0.712**	**0.787**	**0.48**
***HINSAGE* [15]**	0.75	0.85	**0.48**	0.771	**0.801**	0.448	**0.74**	**0.791**	**0.475**
***TRSDL* [9]**	0.73	**0.810**	0.45	0.74	0.820	**0.461**	0.75	0.87	0.44

Table 2
A synthesis of RMSE and MAE values for each model including $ndcg@10$ scores, the best scores are in bold.

5.1. Effects on recommendation quality and ranking (RQ1)

Results of our experiments are synthesized in **Table** 2. Initially, as regards to ML-10M dataset, top RMSE and MAE scores are valued from CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ model with $MAE = 0.715$ and $RMSE = 0.791$. Our proposed contextual tag embeddings based NGCF model has also achieved top quality ranking to reach $NDCG@10 = 0.48$. We have noticed that the static tag-based embedding extension of this model that is CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}W2V$ has also achieved good results outperforming most of the baselines except TRSDL model [9] that has reached $MAE = 0.73$, $RMSE = 0.810$ with a ranking metric of $NDCG@10 = 0.45$. Regarding to Hinsage model [15] that reached $MAE = 0.75$, $RMSE = 0.85$ with a ranking score of $NDCG@10 = 0.48$ and CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ model that reached $MAE = 0.774$ and $RMSE = 0.89$ with a ranking quality that achieved $NDCG@10 = 0.451$, we might be tempted at first sight to claim that NGCF approaches describe strong performance compared with other neural collaborative approaches no matter which tag embeddings we have integrated to the models. However, by considering the significant performance of the neural models that integrate contextualized tag embeddings such as Neural CF-MLP$^{++}_{Bert}$ that has achieved scores valued to $RMSE = 0.72$ and $MAE = 0.93$ or even the autoencoder model CF-Autoencoder$^{++}_{Bert}$ that has risen $RMSE = 0.96$ and $MAE = 0.76$, our thoughts then focused to determine which model's architecture performs better among all the proposed neural architectures that effectively do integrate static/contextual tag embedding representations or those who additionally have aggregated tag-based neighborhood embeddings.

Furthermore, in ML-20M dataset, the same NGCF model named CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ has shown top RMSE and MAE score with $MAE = 0.723$ and $RMSE = 0.802$. This confirms the performance of NGCF approaches combined with contextualized tag embeddings. It also appeared that such models reach top quality ranking, additionally, ranking metric score shown that the most competitive baseline is Hinsage [15] with a ranking quality that does not exceed $NDCG@10 = 0.448$. Both CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ and CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ models have the highest ranking scores with $NDCG@10 = 0.47$ and $NDCG@10 = 0.441$ respectively. This is the case even if those models do not use the same aggregation technique nor the same tag embeddings process. In this regard, we found that mean aggregator function which is operated with static tag embeddings in a NGCF process named CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}W2V$ has performed well and obtained $MAE = 0.80$, $RMSE = 0.94$ with a ranking quality of $NDCG@10 = 0.464$ which is a score that outperforms the autoencoder-based model extension named CF-Autoencoder$++_{Bert}$ with $NDCG@10 = 0.44$ since this model has already achieved $MAE = 0.811$ and $RMSE = 0.89$.This demonstrates the efficiency of such aggregation function.

Finally, in ML-25M dataset, impact of contextualized tag embeddings on models is definitely established since both RMSE and MAE scores have shown significant improvements compared to baselines. Such is the case for Neural CF-MLP$^{++}_{Bert}$ model that has reached $MAE = 0.791$, $RMSE = 0.83$ for a quality ranking of $ndcg@10 = 0.46$. It is likewise for CF-Autoencoder$++_{Bert}$ model with RMSE and MAE scores to $MAE = 0.79$, $RMSE = 0.86$ and a ranking quality to $NDCG@10 = 0.445$. On top of that, impact of aggregator functions are also distinguishable through NCGF model scores since we noticed that results were

much improved using a convolutional aggregator function applied to contextualized tag embeddings. CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ model performed best RMSE and MAE scores comparing to CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ model which exploits a mean aggregator function despite such model integrates contextualized tag embeddings. We ensure that those results can be strengthened by increasing the training data.

5.2. Effects on error distribution (RQ2)

In the following, we have discussed the effectiveness of our approaches on predicting user ratings with an acceptable amount of error. We highlighted impact of exploiting contextualized tag-based embedding representations through studying error distribution when predicting user ratings. Such impact is summarized at the top of **Figure** 2. Error distribution values have been presented among testing sets of the data sets *ML-10M, ML-20M and ML-25M*. This is to propose an overview of the error distributions resulted from baselines compared with those from our predictive models that do integrate tag-based static or contextualized embedding representations and describe specific architectures for each model.

First, in ML-10M dataset we observe that error distribution values from the models exploiting contextual tag embeddings such as CF-MLP$^{++}_{Bert}$ and CF-Autoencoder$^{++}_{Bert}$ are most located in the interval $\in [-1, 1]$ compared to the error distribution values of the other baselines. We also observe that the NGCF models that are CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ and CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ outperforming all other models with a number of 980 and 890 accurate predictions respectively. Secondly, in ML-20M we notice that CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ model conduct to a large number of accurate predictions which are estimated to be 7220. Such performance is closely followed by the CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ with a number of 4250 accurate predictions. Lastly, in ML-25M the same models reached 7980 and 7740 accurate predictions respectively.

5.3. Impact of learning aggregated tag-based functions (RQ3)

We have given for each model the validation scores after 20 epochs, this allows us to estimate the model's capacity to generalize past the data that it was trained on. From the bottom of the figure 2 we have Analyzed which models perform optimal convergence rate. It appears that among the three collections that are *ML-10M, ML-20M and ML-25M*, the convergence rate of the models are clearly more significant when it comes from neural graph approaches. Particularly, CF-GNN$^{++}{}_{Mean}$ Agg$_{(k=2)}Bert$ and CF-GCN$^{++}{}_{conv}$ Agg$_{(k=2)}Bert$ that are our NGCF models that exploit fine-tuned tag embedding representations. This leaves us to believe that when contextualized tag embeddings are aggregate throw neighborhood embeddings they give more effective representations of users and items and enhance recommendation quality. We argue that our NGCF approaches catch the multiple semantic dimensions that a tags can take have including the abstract formalization of tag neighborhood embeddings that have conducted to fine-gained representations.

6. Conclusion

Following the experiments, we came to the conclusion that exploiting neural graph models to learn aggregation functions has enabled us to gain quality recommendations and improve ranking quality. We have shown that handling a convolutional aggregator function can generalize an efficient graph-based neural collaborative filtering process. It concatenates contextualized tag embedding representations of user/item nodes from previous layer representations. This has enabled us to gain more refined embedding features and achieved to catch non-trivial tagging behavior.

References

[1] H. A. M. Hassan, G. Sansonetti, F. Gasparetti, A. Micarelli, Semantic-based tag recommendation in scientific bookmarking systems, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 465–469.

[2] J. Manotumruksa, C. Macdonald, I. Ounis, Modelling user preferences using word embeddings for context-aware venue recommendation, arXiv preprint arXiv:1606.07828 (2016).

[3] A. Rücklé, S. Eger, M. Peyrard, I. Gurevych, Concatenated power mean word embeddings as universal cross-lingual sentence representations, arXiv preprint arXiv:1803.01400 (2018).

[4] L. Wu, C. Quan, C. Li, Q. Wang, B. Zheng, X. Luo, A context-aware user-item representation learning for item recommendation, ACM Transactions on Information Systems (TOIS) 37 (2019) 1–29.

[5] Y. E. Wang, C.-J. Wu, X. Wang, K. Hazelwood, D. Brooks, Exploiting parallelism opportunities with deep learning frameworks, ACM Transactions on Architecture and Code Optimization (TACO) 18 (2020) 1–23.

[6] H. Liu, Y. Wang, Q. Peng, F. Wu, L. Gan, L. Pan, P. Jiao, Hybrid neural recommendation with joint deep representation learning of ratings and reviews, Neurocomputing 374 (2020) 77–85.

[7] C. Musto, G. Semeraro, M. De Gemmis, P. Lops, Word embedding techniques for content-based recommender systems: An empirical evaluation., in: Recsys posters, 2015.

[8] W. Zhang, Q. Yuan, J. Han, J. Wang, Collaborative multi-level embedding learning from reviews for rating prediction., in: IJCAI, volume 16, 2016, pp. 2986–2992.

[9] N. Liang, H.-T. Zheng, J.-Y. Chen, A. K. Sangaiah, C.-Z. Zhao, Trsdl: Tag-aware recommender system based on deep learning–intelligent computing systems, Applied Sciences 8 (2018) 799.

[10] M. Vijaikumar, S. Shevade, M. N. Murty, Tagembedsvd: Leveraging tag embeddings for cross-domain collaborative filtering, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2019, pp. 240–248.

[11] L. Guo, Y.-F. Wen, X.-H. Wang, Exploiting pre-trained network embeddings for recommendations in social networks, Journal of Computer Science and Technology 33 (2018) 682–696.

[12] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International conference on machine learning, 2014, pp. 1188–1196.
[13] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[14] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
[15] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in neural information processing systems, 2017, pp. 1024–1034.
[16] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[17] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Tag-aware recommender systems: a state-of-the-art survey, Journal of computer science and technology 26 (2011) 767.
[18] H. Dai, B. Dai, L. Song, Discriminative embeddings of latent variable models for structured data, in: International conference on machine learning, 2016, pp. 2702–2711.
[19] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.
[20] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 891–900.
[21] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019, pp. 165–174.
[22] L. Wu, L. Chen, P. Shao, R. Hong, X. Wang, M. Wang, Learning fair representations for recommendation: A graph-based perspective, in: Proceedings of the Web Conference 2021, 2021, pp. 2198–2208.
[23] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.
[24] Y. Ouyang, W. Liu, W. Rong, Z. Xiong, Autoencoder-based collaborative filtering, in: International Conference on Neural Information Processing, Springer, 2014, pp. 284–291.
[25] S. Sedhain, A. K. Menon, S. Sanner, L. Xie, Autorec: Autoencoders meet collaborative filtering, in: Proceedings of the 24th international conference on World Wide Web, 2015, pp. 111–112.
[26] W. Chen, F. Cai, H. Chen, M. D. Rijke, Joint neural collaborative filtering for recommender systems, ACM Transactions on Information Systems (TOIS) 37 (2019) 1–30.
[27] G. K. Dziugaite, D. M. Roy, Neural network matrix factorization, arXiv preprint arXiv:1511.06443 (2015).
[28] Y. Zuo, J. Zeng, M. Gong, L. Jiao, Tag-aware recommender systems based on deep neural networks, Neurocomputing 204 (2016) 51–60.
[29] L. Zheng, V. Noroozi, P. S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the tenth ACM international conference on web

search and data mining, 2017, pp. 425–434.
[30] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on world wide web, 2017, pp. 173–182.
[31] W. Chen, H.-T. Zheng, X.-X. Mao, Extracting deep semantic information for intelligent recommendation, in: International Conference on Neural Information Processing, Springer, 2017, pp. 134–144.
[32] R. Huang, N. Wang, C. Han, F. Yu, L. Cui, Tnam: A tag-aware neural attention model for top-n recommendation, Neurocomputing 385 (2020) 1–12.
[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.
[34] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of ir techniques, ACM Transactions on Information Systems (TOIS) 20 (2002) 422–446.
[35] X. He, X. Du, X. Wang, F. Tian, J. Tang, T.-S. Chua, Outer product-based neural collaborative filtering, arXiv preprint arXiv:1808.03912 (2018).
[36] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).