IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 7, 2020, accepted October 4, 2020, date of publication October 14, 2020, date of current version October 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3030693

EnsVAE: Ensemble Variational Autoencoders for Recommendations

AHLEM DRIF[1], HOUSSEM EDDINE ZERRAD[2], AND HOCINE CHERIFI[3]
[1]Networks and Distributed System Laboratory, Faculty of Science, Ferhat Abbas University, Setif 19000, Algeria
[2]Computer Science Department, Ferhat Abbas University, Setif 19000, Algeria
[3]LIB, University of Burgundy, 21078 Dijon, France

Corresponding author: Ahlem Drif (adrif@univ-setif.dz)

ABSTRACT Recommender systems are information software that retrieves relevant items for users from massive sources of data. The variational autoencoder (VAE) has proven to be a promising approach for recommendation systems, as it can explore high-level user-item relations and extract contingencies from the input effectively. However, the previous variants of VAE have so far seen limited application to domain-specific recommendations that require additional side information. Hence, The Ensemble Variational Autoencoder framework for recommendations (EnsVAE) is proposed. This architecture specifies a procedure to transform sub-recommenders' predicted utility matrix into interest probabilities that allow the VAE to represent the variation in their aggregation. To evaluate the performance of EnsVAE, an instance — called the ''Ensemblist GRU/GLOVE model'' — is developed. It is based on two innovative recommender systems: 1-) a new ''GloVe content-based filtering recommender'' (GloVe-CBF) that exploits the strengths of embedding-based representations and stacking ensemble learning techniques to extract features from the item-based side information. 2-) a variant of neural collaborative filtering recommender, named ''Gate Recurrent Unit-based Matrix Factorization recommender'' (GRU-MF). It models a high level of non-linearities and exhibits interactions between users and items in latent embeddings, reducing user biases towards items that are rated frequently by users. The developed instance speeds up the reconstruction of the utility matrix with increased accuracy. Additionally, it can switch between one of its sub-recommenders according to the context of their use. Our findings reveal that EnsVAE instances retain as much information as possible during the reconstruction of the utility matrix. Furthermore, the trained VAE's generative trait tackles the cold-start problem by accurately estimating the interest probabilities of newly-introduced users and resources. The empirical study on real-world datasets proves that EnsVAE significantly outperforms the state-of-the-art methods in terms of recommendation performances.

INDEX TERMS Hybrid recommender systems, neural recommender models, collaborative filtering, content-based filtering, variational autoencoders.

I. INTRODUCTION

Recommender systems are software tools conceived to assist users in reaching the most relevant elements from a gigantic collection of resources. They are widely used by websites and content-intensive systems to improve their user experience. Given the different purposes of recommendations, a plethora of recommender systems have been developed, with the earliest solutions being based on Collaborative Filtering (CF) models [1]–[5] and Content-based Filtering models (CBF) [6], [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Luca Bernardi.

Collaborative Filtering (CF) recommender systems extract the list of recommendations based on items' or users' similarities. It is drawn from the idea that if two users rate a set of items similarly, it is most likely that they share the same preferences. As for two resources that are rated similarly by several users, they would probably be rated likewise by the rest of the users. Hence, collaborative filtering models attempt to learn similarities between users to impute the missing values from the utility matrix accordingly. Collaborative filtering algorithms are divided into two groups: Memory-Based Collaborative Filtering approaches predict ratings based on their neighborhood and extract users/items with a similar history of ratings (similar users to a target user

or similar items to a target item). For example, the distances between users are computed. Then, the similarity is calculated as that the lower the distance, the higher the chance they are akin. An overabundance of similarity techniques has been proposed in the literature [8], [9]. On the other hand, Model-based CF algorithms [10], [11] use different machine learning methods on the training set that find patterns in the data and learn a model for predicting new ratings.

Content-based recommender systems use the descriptive attributes of items — which are labeled with ratings — as training data to create a user-specific classification or regression modeling problem, representing his profile. Mak *et al.* [12] designed a web-based movie recommender that makes suggestions based on text categorization of movie synopses. Semeraro *et al.* [13], [14] proposed a system that provides items recommendations in specific domains. However, the increase of the multi-modal data requires analytical cross-modal approaches that are vital for undertaking appropriate recommendations. Consequently, traditional recommender systems cannot take advantage of joint representation learning.

With the emergence of deep learning, many solutions were proposed in the literature for enhancing recommendation models. Neural-based recommender systems have the potential to learn better representations, and consequently produce accurate predictions than their counterparts, especially when presented with a considerable amount of learning data. He *et al.* [15] proposed a neural architecture to overcomes the cold-start problem by coupling matrix factorization with Multi-Layer Perceptron (MLP) to learn the user-item interaction function.

Numerous studies applied neural networks to hybrid recommender systems. Burke [16] proposed a taxonomy for classifying hybrid recommender systems. It divided them into seven classes: *weighted*, *switching*, *cascade*, *feature augmentation*, *feature combination*,*meta-level*, and *mixed hybrids*. The combination of recommendation techniques proved to be the most effective design pattern. For instance, winning entries in the Netflix Prize contest were mostly ensemble systems (''Bellkor's Pragmatic Chaos'' [17]).

Several challenges plague the creation of recommender systems such as cold start, gray sheep,.. etc. Our main focus in this article is to develop a novel framework that overcomes the difficulties posed by cold start. This problem occurs when a new user is introduced to the system where the recommender would has little to no information about him. A new item, as well, suffers from a cold start issue in a similar fashion. Indeed, the recommender system does not possess any prior history of interactions with the said item. Therefore, it is not able to precisely suggest it to people who might like it. Moreover, in a system that lacks novelty and serendipity, this item may never be recommended at all [18], [20], [44].

The flexibility of deep neural networks makes it possible to combine several neural building blocks to complement one another and form a more powerful hybrid model. The Variational Autoencoder (VAE) has proven to be an effective model that gives significant control over modeling the latent distribution [21], [22]. In a previous work, Liang *et al.* [23] developed a variant of VAE for collaborative filtering that achieved a competitive performance. They model the collaborative information in the form of a multinomial distribution to sample the likelihood of presenting certain items to certain users. However, the main drawback of their proposal is that it does not allow to model a rich semantic representation of data. Therefore, it cannot incorporate additional side information such as content, context, etc. Building on this work, we propose a novel probabilistic framework for hybridizing several building blocks: the Ensemblist Variational Autoencoder.

EnsVAE combines predictions from several sub-recommenders to leverage precise recommendations. Moreover, it transforms the their outputs to interest probabilities, in order to be used as input for a variational autoencoder that learns the distribution of preferences per user-item pair and outputs the interest probabilty predictions per said pair. The advantages of the EnsVAE framework are: a-) Aggregating and transforming the sub-recommenders outputs in a distribution. When sampled, this distribution produces accurate interest probabilities due to the generative abilities of the vartiational autoencoder. b-) EnsVAE framework retains as much information as possible from the distribution of preferences per user-item pair depending on the sub-recommenders. c-) The probabilistic nature of sub-recommenders' outputs alleviates many recommendation problems, including user biases and the cold start problem. d-) EnsVAE ensures an efficient prediction time because once the VAE model is trained, it is used directly for online predictions. Furthermore, it manages the introduction of new users/items without the need to resort to the sub-recommenders.

Inspired by the stacking ensemble model from the Machine Learning literature, instances of the EnsVAE framework follow the same best-practices in creating efficient and accurate hybrids. For example, in a merchant website, a content-based sub-recommender would be used to suggest items in the ''Similar Items'' section of the web-store. Whereas, a collaborative filtering sub-recommender is used to recommend items in the ''Users Like You Bought'' section. Subsequently, a ''Just for You'' is filled with the combined output of both sub-recommenders for a more personalized user experience.

To evaluate a first instance of the EnsVAE framework, two sub-recommender systems are developed: 1) Gated Recurrent Unit based matrix factorisation (GRU-MF) and 2) stacking Global Vectors based content filtering (GloVe-CBF).

The GRU-MF is a variant of Neural Collaborative Filtering approach (NCF) [15]. It uses deep learning to perform matrix factorization over the sparse, ground-truth rating matrix. It models the pairwise correlations between the dimensions of the embedding space for user-item interactions. The embedding layers are chained with a GRU layer, helping these embedded representations with memory-based reiterations thanks to update and reset gates [24]. Basically, the update gate determines how much of the past information needs to

be passed along to the future. The reset gate, on the other hand, controls thd amount of previous information that needs to be dropped. Therefore, adding the GRU layers within the GRU-MF recommender allows the retention of important information indicating similar behavior between users/items.

The stacking content filtering system — based on Global Vectors for Word Representation (GloVe) — combines different machine learning models in a manner that they collectively cooperate to leverage an accurate prediction. GloVe is an unsupervised algorithm aimed at getting vector representations for words [25]. This recommender is composed of a stack of ML models. Combining the predictive power of the constituent models allows it to gain an accuracy edge. Unlike previous works with stacked recommenders [26]–[28] the proposed stacking GloVe-CBF considers the content for recommendation to create a profile model for each user. Its main advantage is the ability of the embedding representation to integrate the side information in the EnsVAE framework.

Our contributions are summarized as follows:

- We developed the Ensemble Variational Autoencoder (EnsVAE), an architecture for building probabilistic framework. The EnsVAE framework is built based on mapping the original data to higher-order features interactions. This architecture is aimed at reducing the user-item interactions bias and improving the effectiveness of the collaborative systems. Furthermore, the EnsVAE framework falls in both mixing and switching classes of Burke's taxonomy, as it can switch between one of its underlying sub-recommenders or use a combination of them, all according to the context of their use.
- The proposed EnsVAE framework is used to learn the interest probability distribution on a per-item basis. Therefore, allowing the variational autoencoder to correctly learn and represent the distribution pattern. Moreover, sampling from the distribution generates the complete rows of new users and the columns of new resources even for minimal interaction.
- EnsVAE provides a flexible architecture that allows the creation of various instances for different use cases. However, the choice of recommender systems plays a major part in the efficiency of the final recommender systems both performance-wise and precision-wise.
- To evaluate EnsVAE, an instance called the ''Ensemblist GRU/GLOVE Model'' (EnsGG) is developed. It is composed with two innovative sub-recommenders: Glove-CBF and GRU-MF.
- The GRU-MF recommender models a high level of non-linearities and exhibits interactions between users and items in latent embeddings. Its Gated Recurrent Units adjusts embeddings for better collaborative filtering. Also, it reduces user biases towards items that are rated frequently by users.
- The GloVe-CBF exploits the strengths of embedding representations and the stacking ensemble learning for modeling the content. Our motivation while suggesting this recommender is to extract effective features from the item- based side information, which would ameliorate the analysis of user-item interactions and thus resulting in more accurate predictions.
- EnsVAE framework is a very powerful design and can be considered as a successful guideline for hybridizing various sub-recommenders. The empirical evaluation demonstrates that the EnsGG instance significantly outperforms state-of-the-art baselines on several real-world datasets, including two recently proposed neural-network approaches. It is worth mentioning that GloVe-CBF on its own managed outperforms the state-of-the-art recommendation methods, according to our evaluations.

The rest of this article is structured as follows: Section 2 reviews related works. Section 3 introduces the proposed EnsVAE framework for designing and building high-performance hybrid recommender systems. We create an experimental EnsVAE-compliant hybrid based on a two innovative sub-recommenders: collaborative GRU-MF model and an item-based content-based model. Section 4 is devoted to data and method presentation. Section 5 reports the results of the experimentations with EnsGG over real-world datasets. Finally, conclusions are provided in Section 6.

II. RELATED WORKS

A. COLLABORATIVE FILTERING APPROACHES

In [29], the authors provided a recent comprehensive survey of collaborative filtering recommendation algorithms and compared their performance in terms of different evaluation metrics. Latent factor methods leverage pinpoint dimensionality reduction techniques that reduce, rotate, and impute the missing values of an incomplete data matrix [30]. For example, Matrix Factorization (MF) methods provide neat mathematical approaches that estimate the entire data matrix in one shot with high precision [31]–[33]. In [34], the authors proposed a collaborative filtering approach based on the users' and locations' points of view. They incorporate a geographical model into the matrix factorization approach. He *et al.* [35] used an outer product above the embedding layer results to explicitly model the pairwise correlations between the dimensions of the embedding space for user-item interaction. Several works presented recommender systems based on Recurrent Neural Networks (RNNs) to model the temporal dynamics and sequential evolution of content information [36]–[38]. Tang and Wang [39] proposed a sequential recommendation that incorporates the Convolution Neural Network (CNNs) to learn sequential features, and Latent Factor Model (LFM) to learn user-specific features. A Graph Neural Networks (GNNs) recommender has been proposed to learn meaningful representations for graph data. The main idea is how to iteratively aggregate feature information from local graph neighborhoods using neural networks [40]. In addition to exploiting neural networks such as recurrent and convolutions models, autoencoders have also been incorporated in recommender systems. Sedhain *et al.* [41]

presented an autoencoder model for collaborative filtering. The proposed model takes user vectors or item vectors as input and reconstructs them in the output layer to enhance recommendation performance. In [42], the authors proposed a novel measurement, considering both similarity and trust information to show the prediction reliability in the collaborative filtering approach. In [43], the authors proposed a time-aware recommendation algorithm that is based on an overlapping community structure between users. The algorithm uses a time-weighted association rule to design efficient recommendations. Ahmadian *et al.* [44] developed a social recommender algorithm based on similarity values and trust relations between the users. They propose a temporal clustering approach exploiting the temporal information of ratings provided by users on items and also the social information among users.

B. AUTOENCODERS FOR HYBRID COLLABORATIVE FILTERING

Autoencoders proved their effectiveness in representing user-item interactions for easy and accurate collaborative filtering. They harvested many research successes: Strub *et al.* [45] proposed a collaborative filtering approach using *stacked denoising autoencoder*, which injects side-information to every layer input to mitigate both sparsity and cold start influences. Wu *et al.* [46] designed a similar collaborative model based on denoising autoencoders that learn latent representations of corrupted user-item preferences. It allows the reconstruction of the rating matrix with a smaller error margin. According to the reported experiments, the proposed method outperformed standard baselines in terms of *the mean average precision*. Liang *et al.* [23] also proposed Variational Autoencoders for collaborative filtering (MultVAE). The authors define a generative model with multinomial likelihood representation and use Bayesian inference for parameter estimation. Lee *et al.* [47] suggested a hybrid recommender system based on auto-encoders, which learns a content augmented rating matrix and recovers each row of the rating matrix by a linear combination of the learned basis. And the framework's performance gain is significant in the cold-start. Liu *et al.* [48] combined a stacked denoising auto-encoders with neural collaborative filtering for implicit feedback recommendations. The auto-encoder and its variants, like the VAE, can effectively explore high-level user-item relations and extract contingencies from input data. However, there is little work applying VAE to hybrid recommender systems. Nonetheless, VAE managed to prove itself as a promising approach for recommendation systems.

C. HYBRID RECOMMENDATION MODELS

The recommendation problem literature suffers from no shortage of hybrid approaches, all providing unique advantages over individual algorithms. Zhao *et al.* [49] proposed a cascading hybrid model that uses topic modeling in the first step and matrix factorization in the second to compute the expectations of users' ratings on items. Romov *et al.* [50] won the RecSys 2015 Challenge award with an ensemble learning framework that deals with categorical features over implicit feedback. This model goes through two phases: the first one being a purchase detection predictor that infers whether a given user would buy an item or not. The second phase, on the other hand, consists of a purchased item detection classifier, built upon the result of the preceding predictor and the sessions with bought items to approximate items that are most likely to be acquired next. Therefore, the architecture is a *feature augmentation hybrid* built from two successive learners. Frolov and Oseledets [51] proposed the renowned HybridSVD that incorporates both user and item features within the formula of the *Singular Value Decomposition* (SVD) to leverage a more accurate recommender system. The algorithm uses a different similarity metrics that consider all pairwise distances while allowing the insertion of side information. Otunba *et al.* [26] stacked an ensemble of a Generalized Matrix Factorization (GMF) and MLP that propagates the prediction from constituent models through other constituent models to final output. The goal of *stacked ensemble recommenders* is to combine the predictions of several base estimators built with a given learning algorithm to maximize the predictive performance.

III. THE ENSEMBLE AUTOENCODER FRAMEWORK

EnsVAE is an architectural framework for building probabilistic, hybrid recommender systems that fall in both mixing and switching classes of Burke's taxonomy. It describes a simple, yet effective way to regroup several recommender systems in one entity that provides personalized recommendations. Fig. 1 shows the general architecture of an EnsVAE-compliant recommender system. The hybrid is composed of a set of recommender systems (sub-recommenders), with their rating matrices values adjusted to output *interest probabilities*. Afterward, these matrices are combined using a probabilistic aggregation function and passed to the variational autoencoder that learns the statistical distribution of interests across possible user-item interaction patterns. Moreover, the trained variational autoencoder is not affected by the few interactions of newly-introduced users or resources. Therefore, it manages to collaboratively contrast their interactions to the latent distribution and samples accurate interest probabilities per each item for a given user.

Two simple guidelines should be followed for building an EnsVAE-compliant hybrid recommender system.

1) **Adjust the rating matrix of the sub-recommenders to output interest probabilities:** EnsVAE does not impose a fixed number of sub-recommenders to use. It implies, however, that their outputs are adjusted to output continuous values that indicate the degree in which a user is interested in a certain item. Such scale transformation can be achieved using *normalization* (*Min/Max scaling*), where the rating matrix is rescaled to continuous values between 0 and 1, depicting the *interest probabilities* of a user over

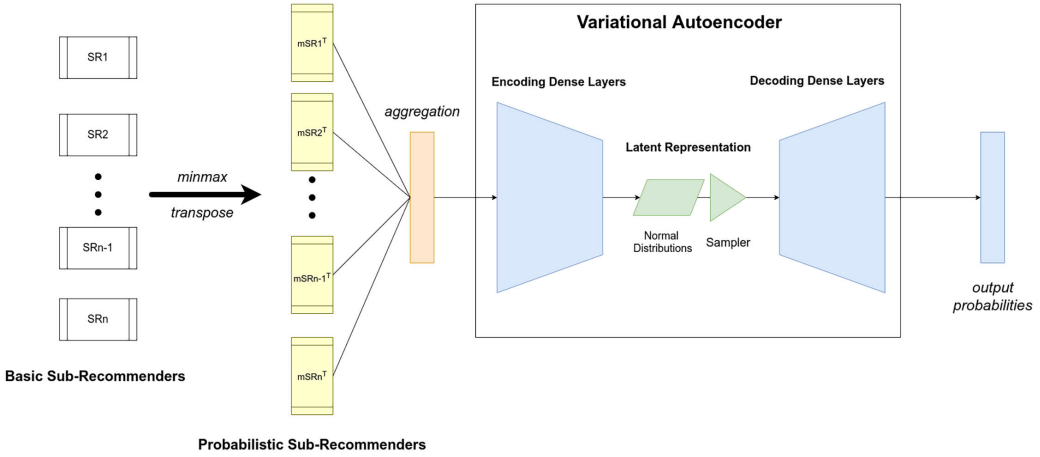


FIGURE 1. Architecture of Ensemblist Variational Autoencoder (EnsVAE).

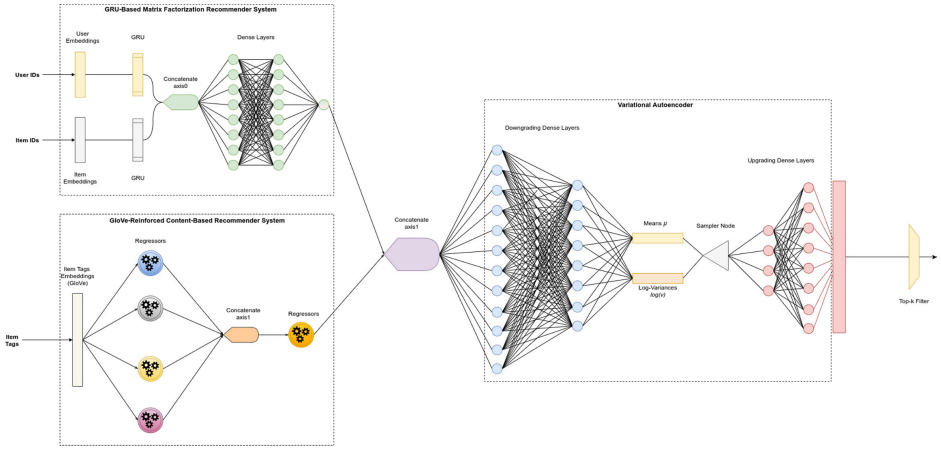


FIGURE 2. EnsGG: A hybrid recommender system based on EnsVAE.

different items. Alongside simplifying aggregation and distribution learning, this transformation alleviates many recommendation problems, including user bias, shilling attacks, and gray-sheep users.

2) **Combine the normalized outputs of sub-recommenders using a probabilistic aggregation function:** After gathering, training, and normalizing the sub-recommenders needed for the task at hand, an aggregation function is applied to merge their outputs into one. Choosing the right aggregation function is very crucial for the accuracy of the final hybrid. EnsVAE does not assume nor impose any method. However, given the probabilistic nature of sub-recommenders' outputs, it is strongly advisable to use aggregation functions of probabilistic events, such as the (weighted) average, intersection of events, or the Bayesian aggregation [52].

To evaluate the framework, an experimental instance based on the EnsVAE architecture is proposed. It consists of two sub-recommenders that focus on an explicit rating scenario: A GRU-based collaborative filtering recommendation system, and a stacking GloVe-based content filtering one. In the following, we refer to this instance as the *Ensemblist GRU/GloVe*, abbreviated *"EnsGG"*.

The proposed model merges a *GRU-based Matrix Factorization Recommender System*, a neural matrix factorization algorithm reinforced with embeddings adjusted using *Gated Recurrent Units*. It adjusts embeddings for more accuracy reconstruction, with a stacking *GloVe- Content-Based Filtring Recommender system*, which uses item descriptions and pre-trained GloVe vectors [25] to learn embeddings for each item. Afterwards, the *variational autoencoder component* learns a high-order features interaction to provide robust rating predictions. *Ensemblist GRU/GloVe* architecture is depicted in Fig. 2.

A. NOTATIONS

The recommendation task is formulated as a prediction problem. The interaction between users and items are represented in the form of a utility matrix R. For each user $u \in U$, the recommender system attempts to predict all of the

TABLE 1. Notations.

Symbols	Definitions and descriptions
n	the total number of users.
m	the total number of items.
$U = u_1, u_2, ..., u_n$	the set of users.
$I = i_1, i_2, ..., i_n$	the set of items.
R	the utility matrix of (n, m) dimensions.
$\hat{R}$	the predicted utility matrix.
$r^s(u)$	the set of ratings explicitly specified by user u.
$r_u = r^s(u) \cup r^u(u)$	the set of all possible ratings of user u. Equivalent to the row representing user u in the utility matrix R.
r_{ui}	a scalar referring to the rating of an item i as specified by user u.
$\hat{r}_{ui}$	a scalar referring to the predicted rating of item i according to user u.
$P_{(n\times k)} \in \mathbb{R}^{N\times K}$	the latent factors for user u.
$Q_{(m\times k)} \in \mathbb{R}^{M\times K}$	the latent factors for item i.
$S_u = arg(r^s(u))$	the set of all items that the user u has explicitly interacted with.
$S_u = arg(r^u(u))$	the complementary set that refers to all the items that are not interacted with by user u.
e_u	the embedding of user u.
e_i	the embedding of item i.
C	the number of observed ratings..

unspecified ratings $\hat{r}_u$. This formulation is more computationally intensive than the ranking problem formulation, but it is more accurate and diverse. Notwithstanding, this computation overload can be alleviated by, for instance, fulfilling the learning phase offline in a cloud-based environment, and conducting predictions on-demand whenever the user requests them. In the rest of this article, we use the notations reported in Table 1.

B. NORMALIZING SUB-RECOMMENDERS

Normalizating the output of sub-recommenders is a crucial step for building EnsVAE-compliant hybrids. For a given predicted rating matrix $\hat{R}$, normalization is done on a user-basis: For each user $u \in U$, the minimal rating $min = \min(r^s(u))$, and the maximal rating $max = \max(r^s(u))$ are extracted. Then, the min/max scaling function is applied to $\hat{R}$ as follows:

$$minmax(x) = \frac{x - min}{max - min} \quad \forall x \in \hat{r}(u) \tag{1}$$

The normalization step can be performed before, during, or after the generation of the rating matrix. Moreover, neural-based sub-recommenders are adjusted easily for this purpose by using *sigmoid* as the activation function for the output layer, ultimately skipping the use of the Min/Max scaler.

C. GRU-BASED MATRIX FACTORIZATION RECOMMENDER SYSTEM

The matrix factorization (MF) (Latent factor model) characterizes items and users using vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. The MF performs remarkably well on the dyadic data prediction task. However, it fails to capture the heterogeneous nature of objects and their interactions. It also falls short in modeling the context in which a rating is given. In fact, the latent factor models are inherently linear, which limits their modeling capacity for recommendation [31], [53], [54]. Recently, a growing body of work adding crafted non-linear features into the linear latent factor models have been introduced. Powered by neural networks, these models aim at boosting recommendation performance [41], [55]. Inspired by these works, a collaborative filtering recommender "GRU-MF" introducing context dependence for rating prediction is proposed.

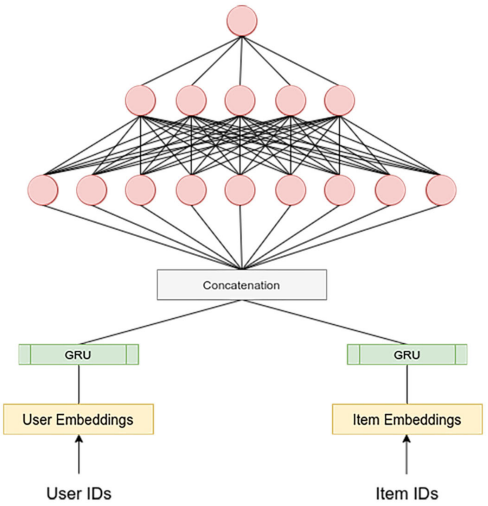


FIGURE 3. Architecture of the GRU-based Matrix Factorization.

GRU-MF is a Collaborative Matrix Factorization (CMF) recommender that model a high level of non-linearities and exhibit interactions between users and items in latent embeddings. Fig. 3 illustrates its architecture.

The matrix factorization algorithm decomposes a matrix $M_{(n\times m)}$ into two matrices $P_{(n\times k)} \in \mathbb{R}^{N\times K}$ and $Q_{(m\times k)} \in \mathbb{R}^{M\times K}$. A user's interaction on an item is modelled as the inner product (interaction function) of their latent vectors. Let R_{ui} be the ground truth rating assigned by the user u on the item i. The estimate interaction is defined as:

$$\hat{R}_{iu} = P^T Q = \sum_{k=1}^{K} p_{uk} q_{ki} \tag{2}$$

where K denotes the dimension of the latent space.

The list of users U and the list of items I are fed to two different embedding layers: e_u and e_i respectively. Each of these embedding layers is chained with a Gated Recurrent Unit layer (GRU). GRUs are used to enhance these embedded representations using their Update gate Γ_u and the Reset gate Γ_r. The outputs of these two GRUs are concatenated

and fed into a fully connected multi-layer perceptron. The predicted score of interaction is $\hat{R_{ui}}$. The predictive model can be formulated as follows:

$$\hat{R_{ui}} = f(P^T e_u^U, Q^T e_i^I | P, Q, \Theta_f) \quad (3)$$

Θ denotes the model parameters, while f denotes the function that maps the model parameters to the predicted score. Note that the output layer is a single neuron with the activation function *sigmoid*. During the training phase, a grid-search method is used to learn the model's hyperparameters. The cost function employed is the *mean absolute error* (MAE) as defined below:

$$L(R_{ui}, \hat{R}_{ui}) = \frac{1}{|C|} \sum_{(u,i)\in C} (R_{ui} - \hat{R_{ui}}) \quad (4)$$

The model predicts interest probabilities for each possible combination of the prediction set $\wp$. The GRU-MF training procedure is summarized in Algorithm 1.

Algorithm 1: GRU-MF Recommender System

Input : U: list of user ids: size n
I: list of item ids: size m
R: list of groundtruth ratings per couple (u, i): size $s \leq n \times m$
Θ_f: the model parameters of the interaction function.
Output: $\hat{R}$: predicted utility matrix: size $(n \times m)$
begin
`// Preparing data to be passed to the network`
foreach $u \in U$ **do** $R_i = \text{minmax}(R_i)$;
$C = U \times I$; `/* cartesian product */`
$D = \emptyset$; `/* training set */`
foreach $(u, i) \in D$ **do** $D \leftarrow (u, i, R_{ui})$ **if** $R_{ui} \neq null$ **else** $(u, i, 0)$;
grumf = `BuildModel(`*usersize*=$|U|$, *itemsize*=$|I|$, Θ`)` ;
grumf`.trainModel(`D`)` ; `// grumf is saved`
return *grumf*`.predict(`$\wp$`)`
end

D. GloVe CONTENT-BASED FILTERING RECOMMENDER SYSTEM

The second sub-recommender in EnsGG orients the hybrid to learn the personal preferences of each user concerning resources' traits, including but not limited to their type, genre, and content. First, the recommender extracts *"tags"* from items' descriptions or reviews and uses them to calculate each item's embedding vector with the help of Stanford's pre-trained GloVe vectors [25]. Afterward, for each user, the recommender employs the stacking ensemble learning technique to learn his profile [56]. The goal of ensemble learning is to gain a predictive accuracy edge by combining the power of the constituent models.

Let $L = \{L_1, L_2, \ldots, L_l\}$ be different regression models, and x_{train} be the training dataset. U and *emb* are independent variables, R is the dependent variable. Our base regression models' hyperparameters are $\forall l \in L \quad \theta_l$. The number and the type of regression algorithms is tunable.

Each $l \in L$ is trained separately with the same training dataset. Each model provides predictions for the outcomes (R) which are then cast into a meta-learner (blender). In other words, the L predictions of each regressor become features for the blender. The latter can be any model such as linear regression, SVR, Decision Tree, etc..., as expressed in (5).

$$f_{features}(x) = f_{STK}(L_1(x), L_2(x), \ldots, L_l(x)) \quad (5)$$

A blender model can then be defined and tuned with its hyperparameters $\theta_{blender}$. It is then trained on the outputs of the stack L. It learns the mapping between the outcome of the stacked predictors and the final ground-truth ratings. The final prediction is expressed as follows:

$$\hat{R_{BL}} = \phi(f_{blender}(x), f_{STK}(L_1(x), L_2(x), \ldots, L_l(x))) \quad (6)$$

A user profile in an ML content-based recommender system is a trained model. Thus, GloVe-CB generates a *user profile* for each user u in the form of a *learned model*. The training procedure of this second sub-recommender is summarized in Algorithm 2.

E. VARIATONAL AUTOENCODER AND SUB-RECOMMENDERS

In the final step, the output of the sub-recommenders is aggregated into a single utility matrix. EnsGG uses the simple unweighted average aggregation function. VAE's architecture is similar to other autoencoder categories, as it is composed of two MLPs: the encoding and decoding layer, with a layer in the middle that represents the data in another form [21]. The user input x_u is encoded to learn the *mean* μ, and the standard deviation σ of the K-dimensional latent representation through the encoder function $g_\phi(x)$ (Eq 7). The latent vector for each user, z_u, is sampled randomly from a Gaussian distribution with μ and σ. The decoder function d_θ (Eq 8) is then used to decode the latent vector from K-dimensions to a probability distribution π_u in the original N-dimension. The variational autoencoder replaces individual variational parameters with an inference model parameterized by ϕ. The output is a probability distribution over the K items.

$$g_\phi(x_u) = \mu_u, \sigma_u \quad z_u \sim \mathcal{N}(0, I_k) \quad (7)$$

$$d_\theta(z_u) = \pi_u \quad (8)$$

Variational autoencoders employ two *loss functions*. The first is the usual reconstruction loss that pushes the autoencoder to reproduce its input, such as the cross-entropy or the mean squared error. The second is the latent KL (Kullback-Leibler) divergence between the target distribution and the actual distribution of the coding. It pushes the autoencoder

Algorithm 2: GloVe-CBF Recommender System

Input : u: current user
I: list of item descriptions: size m
R: list of groundtruth ratings per couple (u, i): size $s \leq n \times m$
θ_{reg}: regressors hyperparameters
$\theta_{blender}$: blender hyperparameters
Output: $\hat{R_{BL}}$: predicted utility matrix: size $(n \times m)$

```
begin
    // Preparing data to be passed to the stack
    foreach u ∈ U do R_i = minmax(R_i) ;
    // Calculating embeddings for each item
    foreach i ∈ I do
        tags(i) ← nlp.extractKeywords(i) ;
        emb(i) ← glove(tags(i))
    end
    // Generating user profiles
    profiles = ∅ ;
    foreach u ∈ U do
        regressors = initRegressors(θ_reg) ;
        blender = initBlender(θ_blender)
        // Training the stack
        features = ∅
        foreach r ∈ regressors do features ← r.trainAndPredict(U, emb, R) ;
        blender.train(U, features, R) ;
        user_profile = profile(regressors, blender) ;
        profiles ← user_profile ;
        /* Profiles can be persisted in hard memory and loaded whenever needed */
    end
    // Constructing predicted utility matrix
    R̂_BL = [] ;
    foreach profile ∈ profiles do R̂_BL ← profile.predict() ;
    return R̂_BL ;
end
```

to have coding that look as though they were sampled from a simple Gaussian noise. The formula for calculating the KL divergence is as follows [22]:

$$L = -\frac{1}{2}\sum_{i=1}^{K} 1 + \log(\sigma_i^2) - \sigma_i^2 - \mu_i^2 \tag{9}$$

In practice, VAE's architecture is slightly tweaked by making the encoder output the log of the variance γ rather than the standard deviation σ, i.e. $\gamma = \log(\sigma^2)$. Consequently, Eq 9 is reformulated as shown in Eq 10. This approach is numerically more stable, which speeds up training [57].

$$L = -\frac{1}{2}\sum_{i=1}^{K} 1 + \gamma_i - \exp(\gamma_i) - \mu_i^2 \tag{10}$$

The variational autoencoder takes the aggregated utility matrix as input and attempts to learn the latent representation of user-item interests as a statistical distribution. *Algorithm 3* defines the step to aggregate outputs, and pass them to the variational autoencoder for learning the representation. Once the VAE model is trained, it is used directly for online predictions (with the introduction of new users/items) without the need to resort to the sub-recommenders.

Algorithm 3: The Variational Autoencoder

Input : *CFR*: GRU-MF predicted Utility Matrix: size $(n \times m)$
CBR: GloVe-CB predicted Utility Matrix: size $(n \times m)$
R: groudtruth ratings: size $s \leq n \times m$
b: is the amount of batches
d_Θ:non-linear function that produce a probability distribution over I items
ϕ, θ: Learned parameters
Output: $\hat{R}$: predicted utility matrix: size $(n \times m)$

```
begin
    // Aggregating the utility matrices of sub-recommenders
    X = aggregation(CFR, CBR);
    while not converged do
        for i in 0 to b do
            Estimating μ, σ ;
            Compute d_θ(z_u) = π_u ( where z_u ~ N(0, I_k)) ;
            Compute the objective function.
        end
        Update φ and θ
    end
    R̂ = vae.predict();
    return R̂
end
```

In practice, generating the final predicted utility matrix is optional. The system can generate a single vector of interest probabilities per user. Furthermore, the user is only interested in the top-k items he might like. Hence, the vector can be sorted, potentially cached server-side, and the top-k elements are served to the user on-demand. This reduces the computational overload from the server and enables fast delivery of recommendations.

IV. EXPERIMENTAL SETTINGS

A. DATASETS

Two datasets from real-world applications are used to evaluate EnsGG: MovieLens and Amazon Review.

MovieLens are user-movie ratings collected from a movie recommendation service [58]. It is the most popular MovieLens Dataset version that is widely used for benchmarking recommender systems. It contains 1 million ratings from 6000 users over 4000 movies.

Amazon Review is a set of interactions and reviews collected from millions of items provided by Amazon. It contains real information about interactions between users and a colossal number of products of different types and categories. Initially mined by Jianmo Ni *et al.* in 2014, it is updated constantly [59]. The last version was released in 2018. The whole Amazon review dataset includes around 233.1 million ratings/reviews. Therefore, using the entire dataset for research is virtually impossible. Consequently, smaller versions of the dataset (divided by the category of items) are made available for researchers and developers. The *electronics* split, alongside their metadata, is used in the experiments. Table 2 summarizes the basic statistics of the two datasets.

TABLE 2. Datasets statistics.

Specifications	MovieLen1M	Amazon dataset
# of Users	6040	4201696
# of Items	3883	476002
# of Ratings	1000209	20994353
Sparsity %	95.5%	99%
Item Description	Genomic Tags	Attributes
User Description	Demographics	N/A

B. VALIDATION PROTOCOL

First, EnsGG is composed of two sub-recommenders: A a GRU-Based Matrix Factorization recommender system, which is a deep neural network, and a GloVe-content based filtering recommender system (GloVe-CBF), that is essentially based on a stacking ensemble of traditional machine learning models (namely a linear regression, an RBF-kernel SVM, a poly-kernel SVM, and an ARD regression as meta learner). The (aggregated) output of these two recommenders is passed to a variational autoencoder, which is also a deep neural network. The dataset is subdivided into two subsets: 75% for training, and 25% for testing in a stratified manner (i.e. the same proportion of appearance of each user is kept the same in training and test sets as they are in the whole dataset). Let us note the training set and the test set as Tr, Ts respectively. Therefore, GloVe-CBF is trained using Tr, and its performance is evaluated using the test set Ts that it has never seen. On the other hand, GRU-MF and the VAE are both trained with early-stopping, which is a technique that stops training the neural network when the objective function's score is not getting any better after a patience threshold. This ensures obtaining the best model possible with significantly shorter learning time. Therefore, to implement this strategy while keeping the test set *Ts* unseen (hence ensuring the integrity of evaluations). The training set Tr is further subdivided into 80% pure training set *PTr*, and 20% validation set *V*. The former is used for training the neural networks, whereas the latter is used for evaluating the current performance of the network per epoch. It is also to note that this division of *Tr* is done randomly and per epoch, meaning that *Ptr* and *V* are different during each epoch. This trades some bias in favor of the variance. Fig.4 illustrates this division.

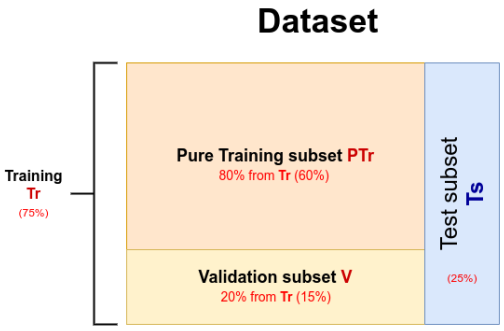


FIGURE 4. Dataset Splitting for training and validation. GloVe-CBF is trained using the Training set Tr, and its performance is evaluated using the test set Ts. Tr is further subdivided into 80% pure training set PTr, and 20% validation set V. The former is used for training the neural networks, whereas the latter is used for evaluating the current performance of the network per epoch.

C. METRICS

Two widely-used metrics for recommender systems evaluation are used to assess the effectiveness of EnsGG (The Mean Average Precision and the Normalized Discounted Cumulative Gain).

1) MEAN AVERAGE PRECISION

The mean average precision (*MAP*) is a popular metric used to measure the accuracy of information retrieval and object detection systems [60]. For a set of queries Q, MAP calculates the mean of the average precision scores for each query $q \in Q$ as follows:

$$\mathrm{Pr} = \frac{TP}{TP + FP}$$

$$MAP = \frac{\sum_{q=1}^{Q} \mathrm{avg}(\mathrm{Pr}_q)}{Q}$$

where *TP* stands for True Positives: positive items that are detected by the system as positive. *FP* denote False Positives: negative items that are detected by the system as positive. Pr_q: the precision value considering only q-first items.

The result is always a value between 0 and 1. The bigger the score, the better the accuracy. In recommender systems, the results are usually truncated to return the top-k elements, where $1 < k \leq q$, and it is variable depending on use: a system may show the top 3 trending items or the best

10 items that match the taste of the current user. Therefore, a more truncated variant of *MAP* is used:

$$MAP@k = \frac{\sum_{q=1}^{k} \text{avg}(\text{Pr}_q)}{Q}$$

2) NORMALIZED DISCOUNTED CUMULATIVE GAIN

The rank in which recommendations are shown is an important factor in determining recommender systems' performance. The normalized discounted cumulative gain (*NDCG*) is a ranking quality evaluation metric that is also used for information retrieval systems [61]. NDCG measures the *normalized* usefulness of items based on their positions in the resulting list by calculation the ratio between discounted cumulative gain (DCG) of the recommended items, over the DCG of their ideal ranking:

$$nDCG = \frac{DCG}{iDCG}$$

$$DCG = \sum_{q=1}^{Q} \frac{2^{\text{Pos}(q)-1}}{\log_2(i+1)}$$

$$iDCG = \sum_{q=1}^{Q} \frac{2^{\text{iPos}(q)-1}}{\log_2(i+1)}$$

where *DCG* is the discounted cumulative gain of the predicted item set, as ranked by the recommender system. *iDCG* is the discounted cumulative gain of the ideal ranking of predicted items. Pos(q) is the position of q, as predicted by the recommender system. iPos(q) is the ideal position of q.

The result $\in [0-1]$ indicates the gain of the recommender. The higher this value, the better.

Similarly to the mean average precision, $NDCG@k$ is used to evaluate the ranking quality of the top-k recommended items. The calculation is the same with the set of items limited to the first k.

D. BASELINES

To ensure the integrity of comparisons, baselines are executed in the same evaluation environment as EnsGG. Furthermore, the *NeuRec* library, which implements 33 neural recommender systems using TensorFlow is used. It is created by a research team led by Bin Wu and released on Github as open-source software under MIT licensing [62]. EnsGG is compared with the following recommendation methods:

- **GRU-based Matrix Factorization:** It is the first component of EnsGG hybrid (*section. III-C*). It is a novel, state of the art collaborative filtering recommender system, based on the neural matrix factorization technique. It takes its power from the GRU units that adjust the embeddings while learning user and item biases thanks to their Update and Reset gates.
- **GloVe-content-based filtering Recommender:** It is the second component of EnsGG hybrid (*section. III-D*). This content-based recommender system uses the pre-trained GloVe vectors and item descriptions to represent the latter in the form of embeddings, which are used afterward in a stacking ensemble model to create a profile model for each user.
- **Neural Collaborative Filtering (NCF): [15]** This recommender system applies the multi-layer perceptron to learn the user-item interaction function. The number of parameters for NCF grows linearly with the number of users and items, and thus training it on a large dataset would be problematic.
- **Neural Attentive Item Similarity Model (NAIS) [63]:** Developed by the same research team that created NCF, this item-based CF method implements an attention network to distinguish which historical items in a user profile are more important for a prediction.
- **Variational Autoencoders for Collaborative Filtering: [23]** This technique extends variational autoencoders (VAEs) to collaborative filtering for implicit feedback. The idea is to model the collaborative information in the form of a multinomial distribution to sample prediction for items that are not seen.

As the proposed method aims to model the relationship between users and items, comparison focuses mainly on user–item models. More particularly with MultiVAE which is a valid related work as it employs the VAE in recommendations but for a different end (collaborative filtering).

E. IMPLEMENTATION DETAILS OF EnsGG

The proposed method implementation is based on Keras [64]. Additionally, the latter runs over Tensorflow [65], which ensures the ability to reproduce the same results by replicating the same developing and evaluating environment. The machine used throughout the whole development and evaluation phases is a custom Lenovo Z40, 2015 edition. It has an Intel® Core™ i7-4500U CPU, with 4 cores running at a clock speed of 1.8GHz each.It is overclockable to up to 2.6 GHz during heavy calculations. It has 8Gb of DDR4 RAM with a transfer rate of 3415MT/s. Although Lenovo Z40 has an an NVIDIA® GeForce™ 820M GPU, it has a restricted CUDA® support.

V. RESULTS AND DISCUSSION

Results of the experiments are performed to answer the following research questions:

- **RQ1:** What are the key hyper-parameters for EnsGG and how do they impact the performance of EnsGG?
- **RQ2:** How do EnsGG performs compared with state-of-the-art recommendation methods?
- **RQ3:** Is the proposed Ensemblist architecture helpful for providing more accurate recommendations?

A. HYPER-PARAMETER STUDY (RQ1)

This section depicts the hyperparameters analysis step to answer (RQ1). It is performed separately on each sub-recommender and the assembling variation autoencoder. Results are evaluated using $MAP@k$ and $NDCG@k$.

TABLE 3. Top 5 GloVe-CBF stacks.

	Mean Average Precision			Normalized DCG		
Stack	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
Stack 1	0.75	0.72	0.7	0.62	0.71	0.74
Stack 2	0.79	0.75	0.73	0.65	0.72	0.76
Stack 3	0.75	0.73	0.72	0.62	0.71	0.75
Stack 4	0.77	0.75	0.73	0.64	0.72	0.75
Stack 5	0.8	0.76	0.73	0.66	0.72	0.75

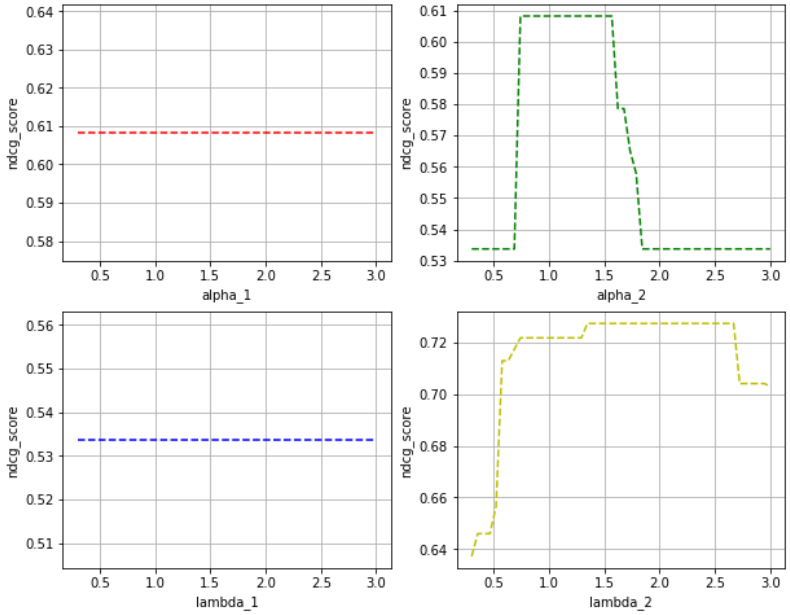


FIGURE 5. Hyperparameter searching GloVe- Content-Based Filtering Recommendation System.

The GloVe-CBF recommender is a stack-based ensemble learner, composed of a set of regression models and a meta-learner. The idea behind stacking is to enhance *weak learners* with a *strong learner*. Therefore, tweaking the meta-learner's hyperparameters would be sufficient [57]. The analysis is performed on 3-1 stacks. It focuses on the learning algorithms used by each learner alongside the meta-learner's hyperparameters.

The first step is to find the combination of regression models that compose the best stack. A grid search is performed over five algorithms (*polynomial-kernel support vector machines* (SVM_{poly}), *RBF-kernel support vector machines* (SVM_{rbf}), *decision trees* (DT), *automatic relevance detection regression* (ARD) and *linear regression* (LR)). Table 3 reports the results of the top 5 stacks for $k \in 10, 30, 50$ averaged over MovieLens and Amazon datasets.

The structure of each stack is as follows

- *Stack 1:* DT - LR - SVM_{rbf} | ARD
- *Stack 2:* LR - DT - ARD | SVM_{poly}
- *Stack 3:* SVR_{poly} - DT - DT | ARD
- *Stack 4:* DT - ARD - SVR_{poly} | SVR_{rbf}
- *Stack 5:* LR - SVR_{poly} - SVR_{rbf} | ARD

Stack 5 exhibits a higher score as compared to other stacks. Thus, it is selected for further tweaking.

The second step is to fine-tune the stack's learners and the meta-learner's hyperparameters. The meta-learner is an ARD regression model. Therefore, grid search over its hyperparameters, namely α_1, α_2, λ_1 and λ_2, is performed. The evaluation metric used is the normalized discounted cumulative gain (NDCG) at $k = 20$. Fig. 5 shows the changes in NDCG scores while tweaking these hyperparameters. For the GloVe-CBF recommendation system benchmark. The best nDGC scores are NDCG@30=0.75 and MAP@30=0.77. α_1 and λ_1 does not affect the accuracy of the recommendation. On the other hand, changes in α_2 and λ_2 impacts its performance. The final hyperparameters for GloVe-CBF are as follows:

- $\alpha_1 = 0.5$ (arbitrary)
- $\alpha_2 = 0.74$
- $\lambda_1 = 0.5$ (arbitrary)
- $\lambda_2 = 1.4$

GRU-MF is an alternative version of neural matrix factorization. It that drops the need for separate layers for users/items biases thanks to GRUs (see Fig. *3*). Analyzing GRU-MF's hyperparameters leads to tweaking the:

- Dimensions of the embedding α.
- Number of units per GRU layer θ.
- Number of dense layers after the concatenation τ

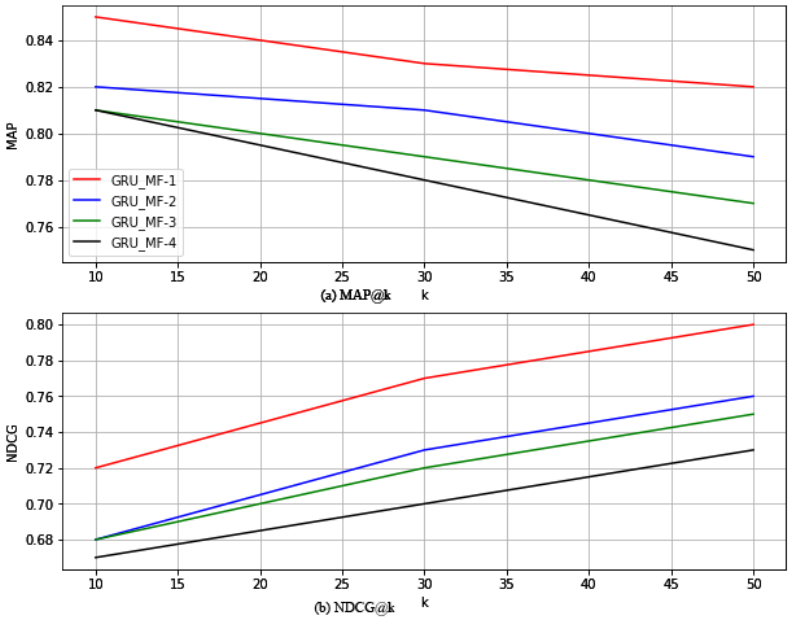


FIGURE 6. Top-4 Hyperparameters for GRU-MF.

- Number of neurons per dense layer ϕ.
- Activation function used in the dense layers σ.
- Optimizer γ.

First, the best hyperparameters are searched and extracted. Furthermore, using the bandit-based Hyperband approach [66], a range of values is explored for each hyperparameter. They are: $\alpha \in [20, 160]$, $\theta \in [20, 200]$, $\tau \in [2, 16]$, $\sigma \in \{selu, elu, relu\}$, and $\lambda \in \{sgd, rmsprop, adam, adagrad\}$

Figure 6 shows the performance of the top 4 GRU-MF networks, based on 4 different hyperparameter settings, as extracted by the optimization approach. The four different hyperparameter settings are defined in Table 4.

TABLE 4. The top hyperparameter settings using the bandit-based Hyperband approach.

	α	θ	τ	σ	λ
1	75	80	8	*elu*	*Adam*
2	75	120	8	*selu*	*Adam*
3	25	80	6	*elu*	*Adam*
4	50	100	3	*selu*	*Adam*

Notably, *Adam* is consistently recorded as the best neural network optimization function along with the top 4 hyperparameter settings. Hence, it is safe to assume that the latter marginally outperforms its counterparts. However, slight variations in the other hyperparameters cause clear shifts in the overall performance of GRU-MF as shown in Figure 6. Therefore, a further test is conducted to determine the degree of impact of each hyperparameter. This is done by a systematic *change-one-fix-rest* approach; First, a neural network is build based on the best hyperparameter setting, as obtained by the Hyperband approach. The value of one hyperparameter is modified while fixing others to their best values. Then, the resulting neural network is trained and evaluated. The process is repeated for each hyperparameter.

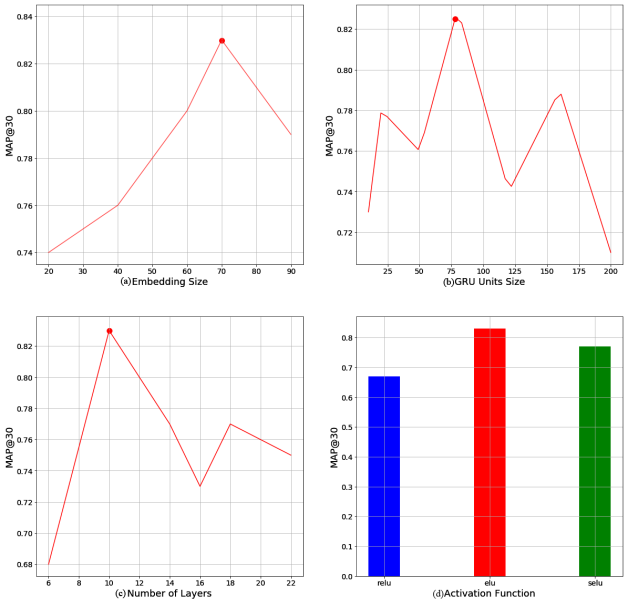


FIGURE 7. Hyperparameter Searching GRU-based Matrix Factorization Recsys.

The resulting neural networks are trained and tested over both MovieLens and Amazon Review Data. Evaluation is performed using the average MAP obtained from both datasets. As shown in Fig. 7, the embedding size α, GRU size θ, and the number of hidden layers after concatenation τ all play

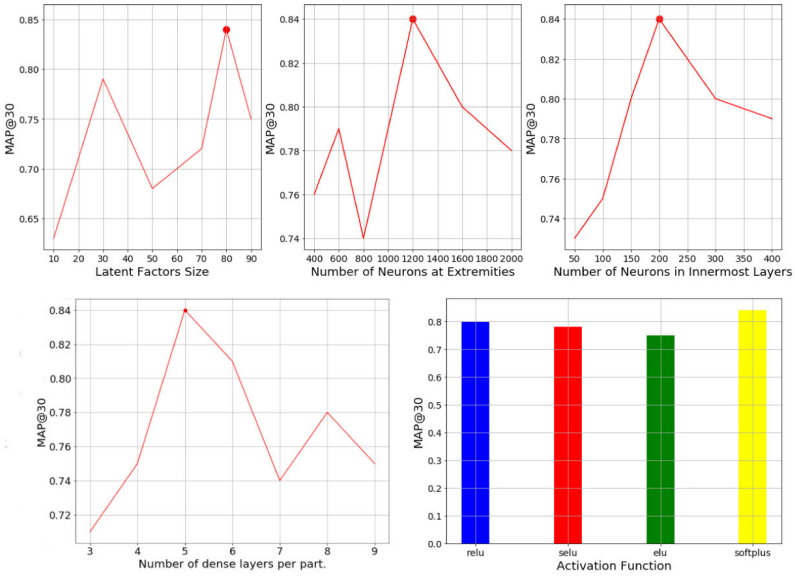


FIGURE 8. Hyperparameter Searching the Variational Autoencoder.

their part in tuning the performance of the recommender system. Also, the *relu* activation function dramatically decreases the performance. Therefore, it is not recorded in any on of the top-4 hyperparameter settings (see Fig. *7.d*). The influence of the embedding sizes and the hidden unit size in GRU is shown in Fig. *7*.a and b. The performance of the GRU-MF improves at the beginning and reaches a maximum at $\alpha = 75$, GRU size $\theta = 80$ with an increase of α and θ. Then the performance starts to deteriorate. In terms of the dimensionality of the latent factors, it indicates that low dimension vectors have a limitation of modeling complex interactions while the high dimension vector may affect the generalization of the recommender and increase the number of parameters. Besides, as presented in Fig. *7.c*, increasing the number of hidden layers improves the performance of the models at the beginning. However, their performance decreases if the number of hidden layers keeps increasing, because of overfitting. Hence, the best performance values of the GRU-Based Matrix Factorization recommender system are $MAP = 0.78$ and $NDCG = 0.73$. It is obtained for the set of parameters: $k = 30$, embedding size $\alpha = 75$, GRU size $\theta = 80$, and $\tau = 8$.

The final component that needs analysis is the variational autoencoder that hybridizes the previous recommenders and represents them in a latent distribution: The output of this VAE is the final output of EnsGG. Since it is initially a neural network, the same analysis method used for GRU-MF is also applied here. Hence, analyzing the VAE leads to tweaking the following hyperparameters in the following ranges:

- Dimensions of latent factors (mean layer/ log of variance layer) $\alpha \in [30 - 100]$.
- Number of dense layers per autoencoder component (encoding/decoding) $\tau \in [3 - 9]$.
- Number of neurons at the extreme layer of each autoencoder component $\Phi \in [400 - 3600]$.
- Number of neurons at the innermost layer of each autoencoder component $\phi \in [50 - 350]$.
- Activation function $\sigma \in \{relu, selu, elu, softplus\}$.
- Optimizer $\gamma \in \{sgd, Nadam, rmsprop\}$.

Fig. *8* shows the impact of each hyperparameter on the performance of the final recommender system. Results are evaluated based on MAP and NDCG metrics. It is clear that α, alongside the size of both the mean (μ_u) and the log of variance $(\log(\sigma^2))$, layers play a major role in the final performance of the hybrid, as seen in the upper leftmost plot. For slight changes in α, *MAP* oscillates in wide ranges, changing from 0.62 at 10 neurons to up to 0.84 at 80 neurons. On the contrary, the activation function has a small impact on *MAP*, as shown at the bottom rightmost plot.

Subsequently, the top hyperparameter setting is extracted by Hyperband approach. Their values are as follows: $\alpha = 80$, $\tau = 5$, $\Phi = 1600$, $\phi = 200$, $\sigma = softplus$, $\lambda = Nadam$. The best performance values associated are $MAP@30 = 0.82$, and $NDCG@30 = 0.70$.

B. PERFORMANCE COMPARISON WITH THE BASELINES (RQ2)

In this section, results of extensive comparisons are reported between various baselines and the EnsGG, GRU-MF, and Glove-CBF methods. Table 5 and Table 6 present the recommendation performance of all methods on the MovieLens and Amazon Electronics datasets (**RQ2**). EnsGG outperforms all baselines on both datasets according to the mean average precision. The most likely reason is that the ensemblist model is based on learning biased and unbiased rating patterns of each user. A list of recommended items is obtained for $k = 10$, 30 and 50 on both datasets.

Fig. 9.a) and Fig. 10.a) present the MAP@k performance versus k-top items. It appears that EnsGG can recall the relevant items for the user better than the other models

TABLE 5. Recommendation accuracy scores (%) of compared methods conducted on MovieLens 1M dataset. We generate Top-10, 30 and 50 items for each user. The best performance of MAP@k and NDCG@k are highlighted as bold font.

	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
NCF	0.74	0.68	0.65	0.68	0.73	0.76
NAIS	0.79	0.76	0.71	0.70	**0.75**	0.77
MultiVAE	0.62	0.58	0.54	0.57	0.62	0.65
Glove-CBF	0.82	0.80	0.78	0.69	0.75	0.77
GRU-MF	0.84	0.82	0.79	**0.72**	**0.75**	**0.78**
EnsGG	**0.87**	**0.84**	**0.82**	0.70	**0.75**	0.77

TABLE 6. Recommendation accuracy scores (%) of compared methods conducted on Amazon Electronics dataset. Top-10, 30 and 50 items are generated for each user. The best performance of MAP@k and NDCG@k are highlighted using a bold font.

	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
NCF	0.70	0.64	0.61	0.62	0.69	0.74
NAIS	0.75	0.72	0.67	0.66	0.70	0.75
MultiVAE	0.62	0.56	0.51	0.55	0.60	0.63
Glove-CBF	0.80	0.76	0.74	**0.69**	**0.77**	**0.8**
GRU-MF	0.78	0.74	0.73	0.67	0.72	0.75
EnsGG	**0.82**	**0.80**	**0.79**	0.67	0.70	0.74

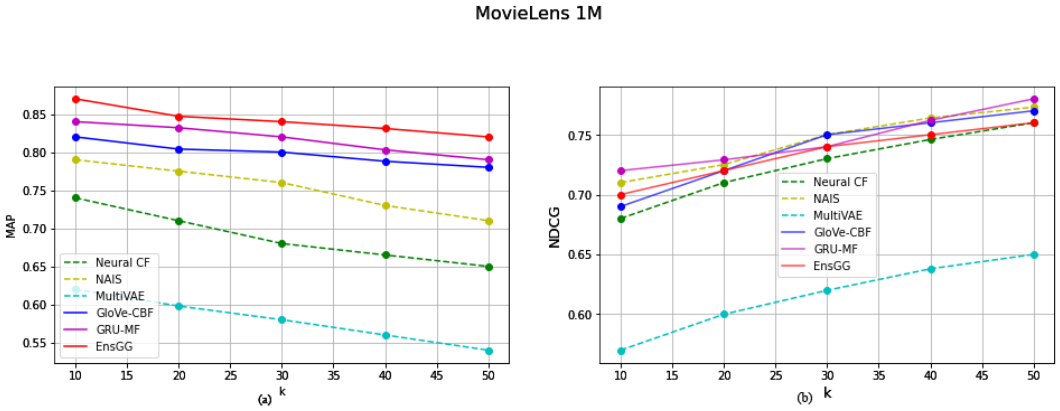


FIGURE 9. Results of the comparison on MovieLens dataset. Evaluation of the performance of Top-K recommended lists, in terms of MAP and NDCG. The ranking position K ranges from 1 to 50.

with a significant margin. Furthermore, both Glove-CBF and GRU-MF produce competitive empirical results on both datasets. For example, on the MovieLens dataset, GloVe-CBF and GRU-MF achieve respectively MAP@10=0.80 and MAP@10=0.78 which is higher than the baselines.

The Glove-CBF recommender acquires the item similarities and creates a user personalized task recommendation justifying its high recorded scores. The GRU-MF recommender as a variant of neural matrix factorization discovers the user-item interactions with a non-linear function and the gated recurrent units produce a noticeable improvement.

Based on the results shown in Fig. 9.b) and Fig. 10.b), one can observe that the GloVe-CBF recommender obtains a high NDCG score on Amazon Electronics dataset and GRU-MF records a high NDCG score on the MovieLens dataset. It means that the order in which an item appears in the top-k recommendation list is close to its order in the ground-truth list. Also, the EnsGG recommender and NAIS method still exhibit good NDCG scores. Similar to the GloVe-CBF recommender, NAIS method characterizes a user with his historically interacted items and recommends items similar to the user's profile. The sparsity rate difference between the two datasets produces slight differences with the score of GRU-MF and the score of GloVe-CBF. Indeed, the prediction of the GloVe-CBF recommender depends on the similarities of items the user has interacted with in the past.

Although EnsGG scored high in normalized discounted cumulative gain (*NDCG*), its performance is a bit weaker compared to its sub-recommenders. This is mainly due to the use of average aggregation on sub-recommenders, which reduces the interest probabilities of some items. Given that *NDCG* is a ranking metric, this reduction may cause items to be misplaced on the query, reducing the *NDCG* score. Nevertheless, the high values of *MAP* indicate that the top-k items are still relevant to the user, albeit misplaced.

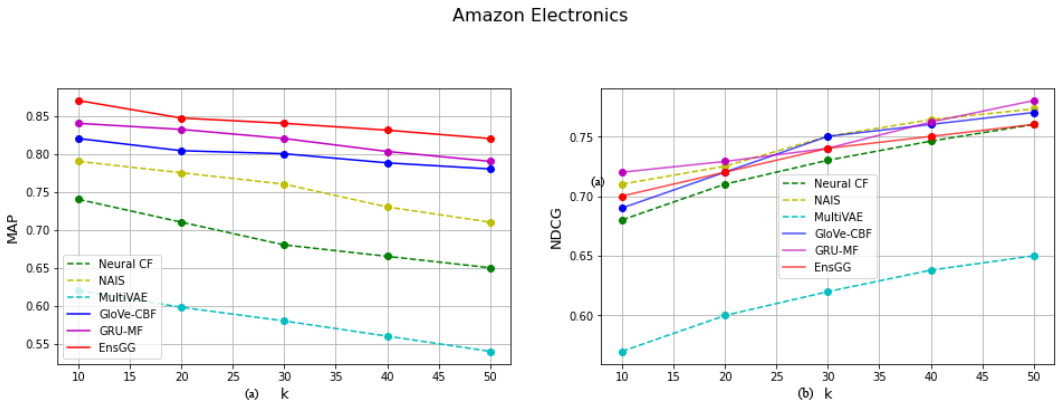


FIGURE 10. Results of the comparison on Amazon Electronics dataset. Evaluation of the performance of Top-K recommended lists, in terms of MAP and NDCG. The ranking position K ranges from 1 to 50.

C. IS ENSEMBLIST GRU/GLOVE FRAMEWORK HELPFUL (RQ3)?

We further evaluate the performance of Top-K recommended lists where the ranking position K ranges from 10 to 50, for the EnsGG and its two sub-recommenders to see whether using a hybrid architecture is beneficial to the recommendation task (**RQ3**). Towards this end, Fig. 9 and Fig. 10 report detailed evaluation results for each recommender system using the MAP and NDCG metric, respectively, over both datasets. EnsGG records the best performance according to the mean average precision metric (*MAP*). As discussed in section B, this is because EnsGG tears its recommendations from combining the GRU-MF and GloVe-CBF. The GRU-MF recommender discovers and exploits the similarity patterns across users and items. The GloVe-CBF recommender learns each user profile as a model.

Overall, the proposed Glove-CBF recommender is effective to obtain the user's overall interest built by the stack's learners. This sub-recommender (Glove-CB) captures the side-information to create a profile model for each user while optimizing the stack's learners' objective function. Moreover, the GRU-MF recommender greatly enhances the modeling of user-item interaction due to GRU units that adjust the user-item biases. Indeed, leveraging a combination of recommendation techniques, namely Glove-CBF and GRU-MF, significantly boosts the recommendation system performance. Hence, the EnsGG framework outperforms other baselines with a significant margin in terms of the mean average precision. This proves that the variational autoencoder successfully represents the combination of sub-recommenders in a distribution that, when sampled, produces accurate interest probabilities.

The GloVe-CBF recommender deal with the cold start by incorporating the item- based side information into the GRU-MF collaborative filtering recommender. Besides, using neural network approaches to incorporate content information into collaborative filtering methods have shown success in many specific-domain recommendation [67], [68]. The aggregated outputs $(\hat{R}, \hat{R}_{BL})$ reduces the user biases towards items that are rated frequently by users. Moreover, the EnsVAE architecture transforms the sub-recommender outputs to interest probabilities, to use them as input for a variational autoencoder. The variational autoencoder learns the interest probabilities and takes random samples from the latent representation of user-item interests as a statistical distribution. These random samples can then be decoded using the decoder network to generate features that alleviate the cold start problem item-wise. An interesting observation that can be made through experiments is that the VAEs can generate the complete rows of new users and the columns of new resources even with minimal interactions. This is due to the fact that the VAE is an effective model for control of the generation of new data.

VI. CONCLUSION

In this article, an *Ensemblist Variational Autoencoder framework* is developed for recommender systems (*EnsVAE*). The key idea is combining sub-recommenders outputs using VAE to ensure the transformation of user-item interactions to interest probabilities and gain predictive accuracy. This ensemblist framework boost the representation capacity, which strongly supports that the EnsVAE is more robust to all kinds of user-item interactions.

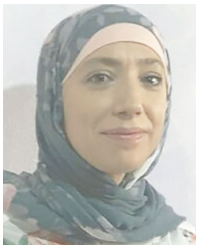
An instance called *EnsGG* following the EnsVAE framework is presented and evaluated. It is composed of two innovative sub-recommenders; the GRU-MF collaborative filtering recommender and a stacking GloVe-content based filtering one. *EnsGG* confirms the effectiveness of the Ensemblist VAE framework. Indeed, it scores very high on benchmarks compared to current, state-of-the-art methods. Furthermore, despite using a simple aggregation function, EnsGG manages to produce accurate predictions without major penalties to the recommender's overall performance. Empirical evaluations show that both EnsGG and GRU-MF/GloVe-CBF provide competitive performance with EnsGG significantly outperforming the state-of-the-art baselines on influential real-world datasets.

Future work aims to explore more deeply the aggregation function on sub-recommenders comparing various functions, such as the Bayesian approach, to gain further improvements on the recommendation accuracy. Technically speaking, another interesting direction worth exploring is to integrate context-awareness which generates items relevant to the users according to a specific context. Indeed, the proposed framework is a highly flexible approach allowing to incorporate high dimensional context information.

REFERENCES

[1] C. C. Aggarwal, Z. Sun, and P. S. Yu, "Online algorithms for finding profile association rules," in *Proc. 7th Int. Conf. Inf. Knowl. Manage.*, 1998, pp. 86–95.
[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, 1994, pp. 175–186.
[3] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, Jan. 2004.
[4] 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, 2003.
[5] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, p. 285.
[6] P. Lops, G. M. De, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender System Handbook*. Boston, MA, USA: Springer, 2001, pp. 73–105
[7] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowl.-Based Syst.*, vol. 157, pp. 1–9, Oct. 2018.
[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004.
[9] W. Zeng, M.-S. Shang, Q.-M. Zhang, L. Lü, and T. Zhou, "Can dissimilar users contribute to accuracy and diversity of personalized recommendation?" *Int. J. Modern Phys. C*, vol. 21, no. 10, pp. 1217–1227, Oct. 2010.
[10] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Soc. Ind. Math.*, vol. 5, pp. 471–480, Dec. 2005.
[11] H. Shen and J. Z. Huang, "Sparse principal component analysis via regularized low rank matrix approximation," *J. Multivariate Anal.*, vol. 99, no. 6, p. 1015–1034, 2008.
[12] H. Mak, I. Koprinska, and J. Poon, "INTIMATE: A Web-based movie recommender using text categorization," in *Proc. IEEE/WIC Int. Conf. Web Intell.*, Halifax, NS, Canada, 2003, pp. 602–605, doi: 10.1109/WI.2003.1241277.
[13] G. Semeraro, M. Degemmis, P. Lops, and P. Basile, "Combining Learning and Word Sense Disambiguation for Intelligent User Profiling," in *Proc. Int. Joint Conf. Artif. Intell.* Jan. 2007, pp. 2856–2861.
[14] G. Semeraro, P. Basile, M. Gemmis, and P. Lops, "User profiles for personalizing digital libraries," in *Handbook of Research on Digital Libraries: Design, Development, and Impact*. Hershey, PA, USA: IGI Global, 2009, pp. 149–158.
[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and C. TS, "Neural collaborative filtering," in *Proc. 26th Int. Conf. on World Wide Web*, Geneva, Switzerland, Apr. 2017, pp. 173–182.
[16] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Model. User-Adapted Interact.*, vol. 12, no. 4, pp. 331–370, 2002, doi: 10.1023/A:1021240730564.
[17] R. M. Bell, Y. Koren, and C. Volinsky, "The BellKor solution to the Netflix prize," AT&T Labs-Res., Netflix, Scotts Valley, CA, USA, Tech. Rep., Oct. 2007.
[18] S. Ahmadian, N. Joorabloo, M. Jalili, Y. Ren, M. Meghdadi, and M. Afsharchi, "A social recommender system based on reliable implicit relationships," *Knowl.-Based Syst.*, vol. 192, Mar. 2020, Art. no. 105371.
[19] S. Ahmadian, M. Meghdadi, and M. Afsharchi Incorporating reliable virtual ratings into social recommendation systems, *Appl. Intell.*, vol. 48, no. 11, pp. 4448–4469, 2018.
[20] S. Ahmadian, M. Afsharchi, and M. Meghdadi, "An effective social recommendation method based on user reputation model and rating profile enhancement," *J. Inf. Sci.*, vol. 45, no. 5, pp. 607–642, Oct. 2019.
[21] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2014, pp. 1–5.
[22] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019, doi: 10.1561/2200000056.
[23] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. World Wide Web Conf.*, 2018, pp. 689–698.
[24] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: http://arxiv.org/abs/1406.1078
[25] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, p. 1532.
[26] R. Otunba, R. A. Rufai, and J. Lin, "Deep stacked ensemble recommender," in *Proc. 31st Int. Conf. Sci. Stat. Database Manage.*, Jul. 2019, pp. 1–5.
[27] A. F. da Costa and M. G. Manzato, "Exploiting multimodal interactions in recommender systems with ensemble algorithms," *Inf. Syst.*, vol. 56, pp. 120–132, Mar. 2016.
[28] X. Bao, L. Bergman, and R. Thompson, "Stacking recommendation engines with additional meta-features," in *Proc. 3rd ACM Conf. Recommender Syst.*, 2009, p. 109.
[29] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating collaborative filtering recommender algorithms: A survey," *IEEE Access*, vol. 6, pp. 74003–74024, 2018.
[30] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, Jul. 2001.
[31] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput. Soc.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[32] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, pp. 1–24, Jan. 2010.
[33] L. Quijano-Sanchez, R.-G. JA, B. Diaz-Agudo, and G. Jimenez-Diaz, "Social factors in group recommender systems," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 1, pp. 1–30, Jan. 2013.
[34] H. A. Rahmani, M. Aliannejadi, S. Ahmadian, M. Baratchi, M. Afsharchi, and F. Crestani, "LGLMF: Local geographical based logistic matrix factorization model for POI recommendation," in *Proc. Asia Inf. Retr. Symp.* Cham, Switzerland: Springer, 2019, pp. 66–78.
[35] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *Proc. Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2227–2233.
[36] C. Musto, C. Greco, A. Suglia, and G. Semeraro, "Ask me any rating: A content-based recommender system based on recurrent neural networks," *Proc. IRR*, 2016, pp. 1–4.
[37] J. Tan, X. Wan, and J. Xiao, "A neural network approach to quote recommendation in writings," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2016, p. 65.
[38] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proc. ACM SIGKDD*, Aug. 2017, pp. 1933–1942.
[39] J. Tang and K. Wang, "Personalized Top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, p. 565.
[40] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, 2019, pp. 417–426.
[41] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, p. 111.
[42] S. Ahmadian, P. Moradi, and F. Akhlaghian, "An improved model of trust-aware recommender systems using reliability measurements," in *Proc. 6th Conf. Inf. Knowl. Technol. (IKT)*, May 2014, pp. 98–103.
[43] P. Moradi, F. Rezaimehr, S. Ahmadian, and M. Jalili, "A trust-aware recommender algorithm based on users overlapping community structure," in *Proc. 16th Int. Conf. Adv. ICT for Emerg. Regions (ICTer)*, Sep. 2016, pp. 162–167.

[44] S. Ahmadian, N. Joorabloo, M. Jalili, M. Meghdadi, M. Afsharchi, and Y. Ren, "A temporal clustering approach for social recommender systems," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2018, pp. 1139–1144.
[45] F. Strub, R. Gaudel, and J. Mary, "Hybrid recommender system based on autoencoders," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, p. 11.
[46] Y. Wu, C. DuBois, Z. AX, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. ACM Int. Conf. Web Search Data Mining,* Feb. 2016, pp. 153–162.
[47] K. Lee, H. Jo, H. Kim, and Y. H. Lee, "Basis learning autoencoders for hybrid collaborative filtering in cold start setting," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2019, pp. 1–6, doi: 10.1109/MLSP.2019.8918843.
[48] L. Yu, W. Shuai, and M. S. Khan, "A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering," *Big Data Mining Anal.*, vol. 1, no. 3, pp. 211–221, 2018.
[49] X. Zhao, Z. Niu, W. Chen, C. Shi, K. Niu, and D. Liu, "A hybrid approach of topic model and matrix factorization based on two-step recommendation framework," *J. Intell. Inf. Syst.*, vol. 44, no. 3, pp. 335–353, Jun. 2015.
[50] P. Romov and E. Sokolov, "RecSys challenge 2015: Ensemble learning with categorical features," in *Proc. Int. ACM Recommender Syst. Challenge*, 2015, pp. 1–4.
[51] E. Frolov and I. Oseledets, "HybridSVD: When collaborative information is not enough," in *Proc. 13th ACM Conf. Recommender Syst.*, Sep. 2019, p. 331.
[52] Y. Yao, "Bayesian aggregation," 2019, *arXiv:1912.11218*. [Online]. Available: http://arxiv.org/abs/1912.11218
[53] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.
[54] P. Gopalan, M. J. Hofman, and M. D. Blei, "Scalable recommendation with hierarchical Poisson factorization," in *Proc. Uncertainty Artif. Intell.*, 2015, pp. 326–335.
[55] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A Neural Autoregressive Approach to Collaborative Filtering," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 764–773.
[56] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, 1996.
[57] A. Gäron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Newton, MA, USA: O'Reilly Media, 2019.
[58] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Sys.*, vol. 5, no. 4, pp. 1–19, Dec. 2015, doi: 10.1145/2827872.
[59] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proc. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, p. 188, doi: 10.18653/v1/d19-1018.
[60] V. Labatut and H. Cherifi, "Accuracy measures for the comparison of classifiers," 2012, *arXiv:1207.3790*. [Online]. Available: http://arxiv.org/abs/1207.3790
[61] S. Balakrishnan and S. Chopra, "Collaborative ranking," in *Proc. Web Search Data Mining Conf.*, 2012, pp. 143–152.
[62] B. Wu, Z. Sun, X. He, X. Wang, and J. Staniforth, "NeuRec: Next RecSys Library," 2019. [Online]. Available:https://github.com/wubinzzu/NeuRec.
[63] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "NAIS: Neural attentive item similarity model for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2354–2366, Dec. 2018.
[64] F. Chollet. (2020). *About Keras*. [Online]. Available: https://keras.io/about/
[65] Tensorflow. (2020). *Why TensorFlow*. [Online]. Available: https://www.tensorflow.org/about
[66] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Res.*, vol. 18, pp. 1–52, Mar. 2018.
[67] D. Liang, M. Zhan, and D. Ellis, "Content-aware collaborative music recommendation using pre-trained neural networks," in *Proc. ISMIR*, 2015, pp. 295–301.
[68] S. Ahmadian, M. Afsharchi, and M. Meghdadi, "A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems," *Multimedia Tools Appl.*, vol. 78, no. 13, pp. 17763–17798, Jul. 2019.



AHLEM DRIF received the Ph.D. degree in computer science from Sétif 1 University. She is currently an Associate Professor with the Department of Computer Science, Faculty of Sciences, Setif, Algeria. She is also a Researcher with the LRSD–Laboratory of Networks and Distributed Systems, Setif. She has published many articles on community structure, link prediction, and information diffusion subjects. Her research interests include social computing, machine learning, intelligent systems, knowledge discovery, and data mining. She also serves as a Reviewer of many international journals and conferences.

HOUSSEM EDDINE ZERRAD received the master's degree in fundamental computer science and artificial intelligence from Sétif 1 University. He is currently a certified Data Engineer and a Professional Machine Learning/Deep Learning Practitioner. He also holds a position as a Data Scientist at FelixSmart, and contributed to many open-source projects, such as the Android Open Source Project and Magenta. His research interests include recommender systems, the IoT, the application of machine learning in industry, and dynamically controlled agents.

HOCINE CHERIFI received the Ph.D. degree from the National Polytechnic Institute, Grenoble, in 1984. He has been a Full Professor of computer science with the University of Burgundy, Dijon, France, since 1999. Prior to moving to Dijon, he held faculty positions at Rouen University and Jean Monnet University, France. He has held visiting positions at Yonsei, South Korea, The University of Western Australia, Australia, National Pintung University, Taiwan, and Galatasaray University, Turkey. He has published more than 200 scientific articles in international refereed journals and conference proceedings. His current research interests include computer vision and complex networks. He held leading positions in more than 15 international conference organizations (the General Chair and the Program Chair) and he served in more than 100 program committee. He is also the Founder of the International Conference on Complex Networks and Their Applications. He is also a member of the Editorial Board of *Computational Social Networks*, *PLOS One*, IEEE Access, the *Journal of Imaging*, *Complex Systems*, *Quality and Quantity*, and *Scientific Reports*. He is also the Founding Editor-in-Chief of the *Applied Network Science* journal.