Neural Processing Letters (2021) 53:1865–1888
https://doi.org/10.1007/s11063-021-10475-0

Enhancing Top-N Recommendation Using Stacked Autoencoder in Context-Aware Recommender System

S. Abinaya[1] · **M. K. Kavitha Devi**[1]

Accepted: 26 February 2021 / Published online: 15 March 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Context-aware recommender systems (CARS) are a vital module of many corporate, especially within the online commerce domain, where consumers are provided with recommendations about products potentially relevant for them. A traditional CARS, which utilizes deep learning models considers that user's preferences can be predicted by ratings, reviews, demographics, etc. However, the feedback given by the users is often conflicting when comparing the rating score and the sentiment behind the reviews. Therefore, a model that utilizes either ratings or reviews for predicting items for top-N recommendation may generate unsatisfactory recommendations in many cases. In order to address this problem, this paper proposes an effective context-specific sentiment based stacked autoencoder (CSSAE) to learn the concrete preference of the user by merging the rating and reviews for a context-specific item into a stacked autoencoder. Hence, the user's preferences are consistently predicted to enhance the Top-N recommendation quality, by adapting the recommended list to the exact context where an active user is operating. Experiments on four Amazon (5-core) datasets demonstrated that the proposed CSSAE model persistently outperforms state-of-the-art top-N recommendation methods on various effectiveness metrics.

Keywords Context-aware recommender systems · Item splitting · Sentiment analysis · Stacked autoencoder

1 Introduction

The enormity of commodities will increase apace within the e-market and to a larger extent, it becomes troublesome for the shoppers to search out their actual preferences. Different types of recommender systems have been proposed in [1, 2]. Context-aware recommender systems (CARS) extends the ancient recommendation approach by considering the particular context during which the user purchases and rates the items because users'

✉ S. Abinaya
abinayavengatesh@gmail.com

M. K. Kavitha Devi
mkkdit@tce.edu

[1] Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, Tamilnadu, India

preference over an identical item might vary because of the context changes and it has been widely used in several application areas [3, 4]. The approaches to build context-aware recommendations are post-filtering, contextual modeling, and pre-filtering [5]. Data sparsity is the major complexity when dealing with contextual modeling and pre-filtering approaches. The item splitting method [6] creates fictitious items by splitting the original item depending upon its contextual factors that make a distinct rating. Similarly, user splitting can also be done depending on the context via items.

Collaborative filtering based recommender systems generate a recommendation based on similarities between customer preference ratings. It usually considers that the users' actual preferences and items' genuine traits can be estimated only by the ratings. Still, in many real-world circumstances, this hypothesis does not accord because certain users always tend to offer the worst rating, while some users always tend to offer a good rating, by which the standard of the product or any resource cannot be revealed [7] and also users with different levels of fulfillment may give the same rating [8]. Additionally, an incorrect click and rating noise are introduced by paid workers to offer false likes or a low rating on products [9] also has an inherent influence on the untrustworthiness of ratings. E-commerce websites, like Yelp and Amazon, permit their customers to provide reviews of their product purchasing experience and offer judgment like down or up to estimate the effectiveness of the feedback [10]. Nowadays, reviews are utilized to enhance the intelligibility of recommender systems [11]. However, the rating and the reviews together provided by the users are often conflicting with each other as shown in Table 1. Therefore, only by considering both the rating and review of the user given for an item under certain contextual situation into the prediction process, the users' true preferences can be estimated.

Abundant sentiment analysis models [12, 13] employ review comments to promote the task of recommendation. For example, [14] uses sentiment analysis in phrase-level reviews by extracting explicit item features and users' attitudes to produce explainable recommendations. Sentiment analysis [15] is conducted on textual reviews using canonical correlation analysis to infer ratings. Rating predictions are improved by considering textual information to acquire review scores [16, 17]. The issue is that the above mentioned conventional straightforward linear transformations can't critically investigate profound semantic associations.

Due to the development of deep learning, Recommender systems utilize several deep learning approaches [18–21] which is proved to be better than that of traditional algorithms and demonstrates to have enhanced accuracy in the prediction process because of its desirable ability to learn the feature representations from scratch. Among different

Table 1 Conflicting reviews and ratings on amazon datasets

Review	Rating
I bought two of these for a midi controller that I use for looping audio samples, and I feel I must pass on is that they don't have much travel, so they would not be a great option as pedals for a keyboard (they would work, but not well)	5
Not very flexible, wouldn't buy again. More like a network cable than an instrument cord. Gets the job done but Meh	4
Good light scent. Have received compliments from people that I smell nice after using it	3
This product is a great deal. They don't sell this in the salon anymore, you can only find it online. It is a great product if you are looking for a shine	2

models, autoencoder [22, 23] an unsupervised method was popularly used for its effective operation in obtaining the latent representation of item attributes or interactions, feature extraction, and rating prediction. In the collaborative deep learning (CDL) model [24] Stacked denoising autoencoder [25] was utilized where the deep embedding of item attributes autoencoder is combined to train jointly with interaction matrix factorization. Hybrid collaborative filtering with the DAE model [26] was introduced which integrates the sparse rating matrix and side information as user/item attributes into a single network. A trust-based approach [27] using DAE was developed for the recommendation in social networks with enhanced rating prediction accuracy. SDDRS [28] utilizes a stacked autoencoder to combine the side information with explicit and implicit rating information, where the representative features are effectively learned. In [29] context-trained embeddings are built and integrated into a deep neural network for polarity detection task on reviews for sentiment analysis. To summarize, none of the approaches recognize the user-specified conflicts between ratings and reviews under a certain context and takes into account the ratings and reviews i.e., multiple feedback together to promote context-aware recommendations by significantly exploring deep semantic associations using neural networks.

Considering all the above motivations, this paper proposes Context-specific Sentiment based Stacked Autoencoder (CSSAE) which improves the context-aware Top N recommendation quality. Specifically, the CSSAE approach employs the item splitting method to model context, which creates fictitious items by splitting up actual item ratings based on all the combinations of contextual conditions [30]. Therefore stacked autoencoder can learn the nonlinear relationship from both the rating and sentiment data which is given exactly to a context-based item and can predict the concrete preference given by the user for an item under a certain contextual situation. The proposed CSSAE approach comprises four main components: (1) Context Generation which aims to capture the dynamic behavior of the user. (2) Item-Splitting for creating fictitious context-based items. (3) Sentiment Analysis determines the sentiment scores of the reviews. (4) Users' consistent preference prediction using CSSAE and top-N recommendation is accomplished depending upon the present context circumstances of the user.

Our proposed approach using CSSAE differs in several aspects from the aforementioned works. First, several approaches [12, 29] have been built in a recommender framework to use the review information. Some method uses review information as an aspect to help the prediction [14]. Different from this work, to obtain a numerical sentiment score, we performed sentiment analysis on reviews. Second, most approaches conducting sentiment analysis on reviews consider whether to apply the impact of sentiment to ratings. [13] Derived both negative and positive labels from feedback and added a positive and negative impact to ratings afterward. [15] Used an analysis of canonical correlations to map text and numerical ratings. Our algorithm, by contrast, fuses both the rating and sentiment information into the stacked autoencoder. Therefore, the latent feature vector (code) in the stacked autoencoder fits the ratings and reviews simultaneously. Third, only a few known works [16, 29] believed that the user's review was a stronger sentiment measure than the rating and none of the autoencoder based approaches[22–28] takes multiple feedback as input for context-aware recommendation generation. In our work, we considered the conflicting issues between rating and review and used a stacked autoencoder to leverage the review of the user by integrating different types of response (i.e.). Combining review with the rating to find the concrete preference of the user in a context-based environment.

In summing up, this paper has the major contribution as listed as follows:

- To the best of our knowledge, Context-specific Sentiment based Stacked Autoencoder (CSSAE) is the first model that computes concrete preference of the user by using stacked autoencoder which expands the first layer of the model for the effective integration of rating and review together given for a particular context-based item which is not possible in traditional autoencoder. Thereby the concrete context-based preference of the user is determined by integrating multiple feedbacks which enables the proposed approach to generate the most relevant item as a recommendation.
- By considering the proposed methodology in the process of generating recommendation shows a significant improvement in its top-N items prediction accuracy when comparing with the methodology which considers that the user preference can be predicted by ratings only.
- Extensive experiments on four real-world Amazon (5 core) datasets confirm that CSSAE outperforms the traditional recommender systems.

2 Preliminaries

2.1 Sentiment Analysis

Sentiment analysis is an efficient process that is widely applied in revealing users' satisfaction and mind-set using resources in the form of reviews. The process of sentiment analysis is done by machine learning technique (corpus-based [31]), a word dictionary (lexicon-based [32]), and a hybrid method (a combination of corpus-lexicon [33]). It has to be mentioned that while using machine learning technique enormous data is required to acquire consistent sentiment outcome because sentiment computation will be interrupted by a noise that occurs due to abnormal sentences.

A sentiment lexicon could be a set of lexical options that are typically labeled consistent with their linguistics orientation as either negative or positive [34]. The creation of a lexicon is the most challenging task which depends on physically created lexicons that pre-exist. In this paper, the most commonly used sentiment lexicon *TextBlob* [35] is used to perform sentiment analysis on user review to compute a numeric sentiment score. TextBlob makes use of a sentiment lexicon and also an engine pattern.en for sentiment analysis. Pattern.en influences WordNet to get sentiment in keeping with adjectives employed in the text. Once TextBlob executes sentiment analysis on text, it returns a tuple of the shape (polarity, subjectivity), wherever the polarity value could be a float value between $[-1.0, 1.0]$. A float number in the range of [0.0, 1.0] is the subjectivity value. For example consider the when the TextBlob runs on the following user review for a product as "*This product was very useful, great value for money*", it returns the output as (polarity = 0.39166666666666666, subjectivity = 0.435714 2857142857). In this paper, a polarity score within the range of $[-1.0, 1.0]$ is taken into account wherever 1 indicates a positive statement and -1 indicates a negative statement.

2.2 Stacked Denoising Autoencoder

Autoencoders [22] was the widely used deep learning methodology for recommender systems which is an unsupervised learning method that learns the robust latent features of the partly corrupted [36] input that is being reconstructed as the output from the learned hidden features. The autoencoder operates with an encoder and a decoder function.

The encoder uses an activation function to encode the input to a hidden representation as Eq. (1):

$$g(r) = \rho\left(V_1 r + c_1\right) \tag{1}$$

The decoder decodes the latent representation to the output using an activation function which trains the network to equate the input as in Eq. (2)

$$h(g(r)) = \rho\left(V_2 g(r) + c_2\right) \tag{2}$$

where $\rho(.)$ is the activation function, r is the input data, $\left(V_1, V_2\right)$ and $\left(c_1, c_2\right)$ are the weights and biases.

Stacked Denoising AutoEncoder [25] is a feed-forward neural network in which for learning latent representations multiple encoding layers are linked that have consecutive decoding layers. Here the latent representation of the entire encoder network resides in the last encoding layer. By merging multiple encoding layers, the performance and the prediction accuracy of the entire network gets increased. According to the functionality, the training is performed separately for the parameters in each layer, whereas the parameters stay unchanged in the rest of the network.

3 Proposed Method

In this section, the proposed CSSAE (Context-specific Sentiment based Stacked AutoEncoder) model was introduced, which learns compact and robust representations from both rating and review data for a context-based item. The framework of the proposed system is given in Fig. 1 and Algorithm (1) summarizes the procedure of the proposed system using CSSAE.

3.1 Problem Formulation

Given a transaction database with a set of users $U = \{1, 2, ..., k\}$, set of items $I = \{1, 2, ..., m\}$, and a record of the users' past multiple preferences of items $O = \left(u, i, rt_{ui}, rv_{ui}, d_{ui}, u_s, i_s\right)$ where rt_{ui}, rv_{ui} indicates the rating and review given for the item i by a user u on a particular date d_{ui} together with the user and item features u_s, i_s respectively, the primary goal is to recommend to each active user u_a a list of N items according to their current context situation c that will maximize his/her satisfaction. In many cases, the record contains missing, unobserved preferences which are denoted as $\tilde{O}$. Let O_u indicate the set of item multiple preferences in the training set for a specific user u, and $\tilde{O}_u$ the unobserved preferences of the user u. Items in $\tilde{O}_u$ are the candidates to be recommended to a user u. The goal of the recommender is to pick for each user u a subset of context-based items that have the highest predicted concrete score from the candidate set.

Algorithm 1: *Proposed System using CSSAE.*

Input: Transaction database *(user_id, item_id, rating, date, item_review, user_feature informations),* Active user u_a.
Output: List of Top –N recommendation Item depending upon the current context situation of the active user.

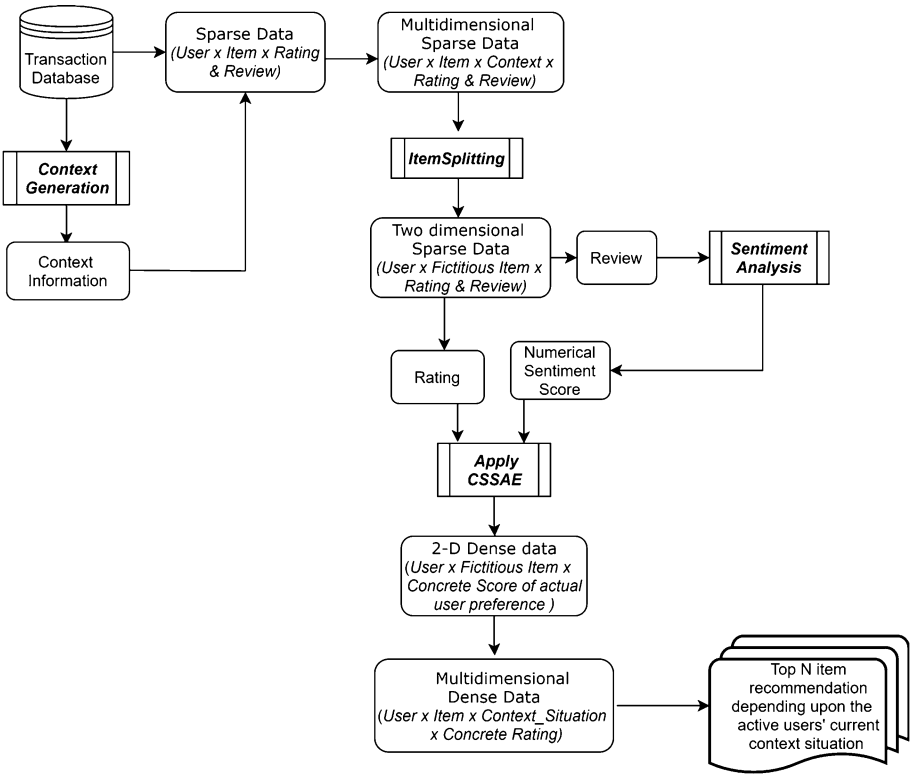


Fig. 1 The framework of the proposed system using CSSAE

Method:

Step 1. Transform the Transaction database into a Multidimensional matrix.
Perform "context generation" procedure as mentioned in the algorithm(2).
Step 2. Convert Multidimensional matrix Q into Sparse 2-D matrix Q′.
Perform the "item splitting" procedure as mentioned in the algorithm(3).
Step 3. Predict numerical sentiment score from review.
Perform the "Sentiment Analysis" procedure as mentioned in the algorithm(4).
Step 4. Sparse 2-D matrix $\hat{Q}$ (*user_id* x *fictitious_item_id*) contains < *rating, Sentiment_Score* > is processed by CSSAE model as described in Sect. 3.5 to a completely filled 2-D matrix $\hat{Q}$ (*user_id* x *fictitious_item_id*) contains < *concrete score* > which is mapped to multidimensional matrix Q(*user_id* x *item_id x contextual situations C)* contains < *concrete score* > .
Step 5. Recommend Top –N Item according to the current context situation for the active user u_a.
Perform "Recommendation Process" as mentioned in the algorithm(5).

3.2 Context Generation

The transactional database contains purchasing information like user_id, item_id, rating, date, review, user_feature information. The database contains different rating values and reviews comments for the same item by the user. Besides the remaining fields, these records have the unique *date* and *rating value. T*he *date* field helps to capture the dynamic behavior of the user. So, the context information is generated from the *date* field. The process for context generation is described in the algorithm (2).

Algorithm 2: *Context Generation.*

Input: Transaction database *(user_id, item_id, rating, date, item_review, user_feature informations).*
Output: Multidimensional matrix $Q < user\_id$ x $item\_id$ x $context >$ contains $< rating, item\_review >$

Method:

1. Three context information are generated from the *date* field.

   a. *Time (Weekend, Weekday)*
   b. *Season (Autumn, Summer, Winter, Spring)*
   c. *Day_info (Festival, Normal)*

The process of context generation is clearly explained with an example. Consider transaction table (Table 2), which contains user id, item id, rating, review, and date. Say, user_id *280* purchased item_id I_1 on different *dates* with different ratings and reviews. So, from the date field, generate three context information (time, season, day_info) as given in Table 3.

3.3 Item splitting

Item Splitting is the process of creating fictitious items by transforming the multidimensional matrix over various context factors into a two-dimensional matrix, where every fictitious item is a merger of contextual conditions and an original item [6]. Generate the fictitious items using the algorithm (3) [30]:

Algorithm 3: *Item Splitting.*

Table 2 Transactional table

User Id	Item Id	Rating	Item_review	Date
280	I_1	5	If you know the scent of.	16-05-2010
861	I_{13}	5	This product is seasonal but i…	04-07-2013
364	I_{25}	3	this gel uses what feels like…	04-12-2013
979	I_1	5	it dries my hair doesnt help to reduce….	04-03-2012
280	I_1	3	Smells great!! Thanks for the fast…	03-11-2017

Table 3 Transactional table with context information

User Id	Item Id	Rating	Item_review	Date	Generated context		
					Time	Season	Day_ info
280	I_1	5	If you know the scent of.	16-05-2010	Weekend	Summer	Festival
861	I_{13}	5	This product is seasonal but i…	04-07-2013	Weekday	Spring	Normal
364	I_{25}	3	this gel uses what feels like…	04-12-2013	Weekday	Winter	Festival
979	I_1	5	it dries my hair doesnt help to reduce….	04-03-2012	Weekend	Summer	Festival
280	I_1	3	Smells great!! Thanks for the fast…	03-11-2017	Weekday	Autumn	Normal

Input: Multidimensional matrix Q(*user_id* x *item_id* x *context_situation*) contains < *rating, item_review* > for users U of size m and items I of size y.
Output: Sparse 2-D matrix Q'(*user_id* x *fictitious_item_id*) contains < *rating, item_review* >
for users U of size m and fictitious items F of size $n = y \times X_C$.

Method:

1. The Cartesian product of the entire context factors is computed (i.e.). A context dimension C along with X_C context conditions is newly constructed as $X_C = X_{C_1} \times X_{C_2} \times \cdots \times X_{C_j}$, whose components are a combination of contextual conditions as of the original context dimensions.
2. The fictitious item f_i was introduced by taking the Cartesian product among the set of items I and the context dimension C.
3. 2D user-item rating matrix along with user review comments is created by removing the generated contexts from the Multidimensional matrix and substituting the item set I with the fictitious item set F.

By the above-mentioned process, a new set of fictitious items of size n are created which is greater than the original items of size m, which therefore makes the matrix sparser. The proposed CSSAE method (Sec 3.5), is capable of predicting the user preference under all possible context situation of the user for all the items by effectively dealing with the sparsity issue. Table 4 illustrates the effect of employing algorithm (2) to the records illustrated in Table 3, for example, a new fictitious item f_1 is created by merging of item I_1 and the context situation "Weekend, Summer, Festival".

3.4 Sentiment Analysis

Sentiment analysis is a procedure used to find out the sentiment scores S_i on each review given to the fictitious item f_i in the dataset. Using Python Textblob, a package installed in Natural Language Toolkit [35] the reviews in the dataset are preprocessed for transferring text from human language to machine-readable format for further Sentiment analysis computational task. The process of sentiment analysis to predict sentiment scores from the review is described in the algorithm (4).

Algorithm 4: *Sentiment Analysis.*

Table 4 Item splitting results

User Id	Item Id	Rating	Item_review
280	f_1	5	If you know the scent of.
861	f_2	5	This product is seasonal but i…
364	f_3	3	this gel uses what feels like…
979	f_1	5	it dries my hair doesnt help to reduce….
280	f_4	3	Smells great!! Thanks for the fast…
861	f_1	φ	φ
.	.	.	.
.	.	.	.
.	.	.	.

Input: 2-D matrix Q' (*user_id* x *fictitious_item_id*) contains < *rating, item_review* >
Output: 2-D matrix $\hat{Q}$ (*user_id* x *fictitious_item_id*) contains < *rating, Sentiment_Score* >

Method:

1. *Item_review* for each fictitious item is preprocessed using the following normalization techniques:
   a. *All the letters in the review are converted to lower case.*
   b. *Punctuation symbols are removed.*
   c. *Spelling Correction is done to correct all the spelling mistakes in the review.*
   d. *Tokenization is performed which splits the review into smaller pieces called tokens.*
   e. *Stopwords that do not carry important meaning are removed from the review.*
   f. *The stemming process is performed to reduce words to their word stem, base, or root form.*
   g. *The lemmatization process is performed to reduce inflectional forms to a common base form.*
2. The *Sentiment score* S_i, which is a float value in the span of $[-1, 1]$ (where 1 indicates positive review and -1 indicates a negative review) was calculated for the preprocessed item_review.

Table 5 shows the results of predicted Sentiment Score by the algorithm (3) to the *Item_review* shown in Table 4, in which, for example, the item_review "*if you know the scent of diva you'll love this body cream everyone says who smells so good in here*" for the fictitious item f_1 is converted to a sentiment score of 0.6.

3.5 Context-Specific Sentiment Based Stacked AutoEncoder (CSSAE)

To deal with users' right preferences and items' genuine traits, a novel approach CSSAE model is proposed as shown in Fig. 2, which utilizes unsupervised stacked autoencoders [28] networks to give appropriate recommendation for the active user through predicting the user concrete preferences exactly from both rating and sentiment data. In the CSSAE model, the network output intends to recreate the original input. In traditional autoencoders, the nodes in the input layer are equivalent to the input attributes.

Table 5 Prediction of sentiment score on review

User Id	Item Id	Rating	Sentiment score
280	f_1	5	0.6
861	f_2	5	0.625
364	f_3	3	− 0.01429
979	f_1	5	0.016667
280	f_4	4	0.48
861	f_1	φ	φ
.	.	.	.
.	.	.	.

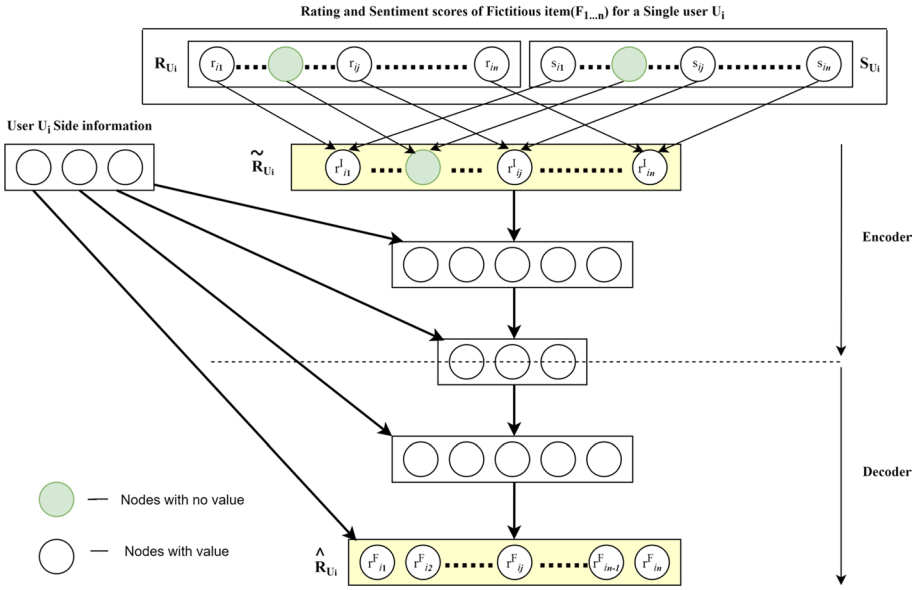


Fig. 2 High-level representation of context-specific sentiment based stacked autoencoder (CSSAE)

However, in-case of dual inputs such as rating and sentiment scores to be considered for concrete rating prediction, each item neuron has two inputs that are required as input to the network. Therefore, the stacked autoencoder was expanded by integrating an additional layer which holds two input layer for both the rating and sentiment data. Thus, the additional layer acts as the input layer in the context-specific sentiment based stacked autoencoder. The first layer acts as a dual input layer which is linked to the corresponding item nodes in the following intermediate layer which in turn is linked to consecutive M encoding layers that are employed to find out the hidden representation of the items. The final encoding layer is linked to M successive decoding layers that are used to decode the hidden factors acquired from corresponding encoders. The Final decoding layer acts as an output where the concrete ratings of the items are predicted.

Let r_{ij}^{J} be the combined rating value of an item j rated by user i which is obtained from both r_{ij}, the rating and s_{ij} the sentiment score obtained from the review specified by user i to item j respectively which is computed as Eq. (3)

$$r_{ij}^{J} = r_{ij} \times w_j^1 + s_{ij} \times w_j^2 \tag{3}$$

where w_j^1 w_j^2 are the corresponding weights of r_{ij} and s_{ij}. These combined rating values $\tilde{R}_{U_i}$ are fed to the succeeding layers of the network. However, due to the sparsity of the rating, using only the users' preferences towards an item is greatly restraining. Therefore, the performance of the recommendation can be improved if more profuse information can be acquired for the item or the user [23, 27]. In CSSAE, to overcome sparsity and to enhance the accuracy, users' side information was integrated [26] in each successive layer as exposed in Fig. 2.

A simple approach is to append the side information into each successive layer of CSSAE to train the model. When no preceding rating exists, it facilitates the CSSAE model to include it as an input to forecast pristine ratings. Hence, the users' side information is incorporated as pseudo-ratings which persist always for every user. The association between user preferences and side information data is extremely nonlinear with diverse distributions. Therefore, to overcome this obstacle and to promote the learning features from side information, that information's are appended into all the consecutive layer of the model as in Eq. (4):

$$\hat{R}_{U_i} = \rho\left(Z'\left(\rho\left(Z\left\{r_i^J, e_i\right\} + d\right), e_i\right) + d'\right) \tag{4}$$

where $\hat{R}_{U_i} \in \mathbb{R}^{\mathrm{M}}$ is the output layer where the user preference of concrete score resides, $Z \in \mathbb{R}^{\mathrm{H}\times(\mathrm{M}+\mathrm{T})}$ and $Z' \in \mathbb{R}^{\mathrm{N}\times\mathrm{H}}$ are the weight matrices, $r_i^J \in \mathbb{R}^{\mathrm{M}}$ is the combined score of item j in the sparse $\tilde{R}_{U_i}$, $e_i \in \mathbb{R}^{\mathrm{T}}$ is the users' U_i side information, $d \in \mathbb{R}^{\mathrm{H}}$ and $d' \in \mathbb{R}^{\mathrm{M}}$ are the bias vectors, and hyperbolic tangent function ρ.

Here each layer is enforced to incorporate the side information and to prevent side information to step over, a constraint is imposed that the dimension of the side information T must be smaller than the hidden layers' dimension, and the latter must be smaller than the input layers' dimension M [27], i.e., $T << H << M$.

Finally, a legitimate and personalized concrete rating value $\hat{R}_{U_i}$ is obtained by a fully connected network which is designed using the stacked autoencoder for the entire fictitious item set F for all users U in the matrix $\hat{Q}$. To learn compact representation, the following modified loss function is used to train the proposed models, since it does not make sense to predict loss function for those values that are zero in the user rating vector. Therefore, the objective function of the proposed CSSAE model by combining regularization terms to avoid overfitting the network and is shown as in Eq. (5):

$$L\left(\hat{R}, \tilde{R}\right) = \sum_{j \in R'} \left(CSSAE\left(\hat{R}\right)_j - \tilde{R}_j\right)^2 + \frac{\lambda}{2}\left(\|Z\|_F^2 + \left\|Z'\right\|_F^2 + \|w\|_F^2 + \|d\|_F^2 + \left\|d'\right\|_F^2\right) \tag{5}$$

where $\hat{R}$ is the complete set of concrete score instances such that $\hat{R} = R' \cup R''$, R' includes every well-known concrete score instance and R'' includes the unknown rated instances. The network $CSSAE(\cdot)$ is the fully connected expanded stacked autoencoder where CSSAE(R') is the predicted concrete rating of instance j and $\tilde{R}_j$, the actual concrete score of instance j. From Eq. (5), it can be obsesrved that the loss value has been computed only for the well-known combined rating instances set (R'). The stochastic gradient descent variant ADAM optimizer [48] is used to update the parameters such as $\|Z\|$, $\|Z'\|$ which are the

weight vectors of the stacked autoencoder, $\|w\|$ is the weight vectors of rating and sentiment scores of each item, weight vectors of bias are $\|d\|, \|d'\|$ and λ is the regularization hyper-parameter used to influence the learning degree that affects the generalizability of the model, and hence it is necessary to find out the proper value of λ which is determined by the experimental results. Furthermore, dropout regularization with a probability q is added at each layer, to prevail over the identity network.

Here, dual inputs such as rating and sentiment scores with the dimensionality of n are first mapped to merge into the intermediate layer, then mapped to the middle layer with the dimensionality of K, and then mapped back to predict input data. For each iteration, the CSSAE model is trained above all users and the whole model complexity of CSSAE is $O(un + unk) = O(un(1 + k))$. This is not effective when the number of users and items are very high. Toward this issue, the learning strategy as in [23] is used in the proposed system. Rather than calculating the gradients on the entire outputs, we only sample a subset of the negative items S_u^R and S_u^S from zero entries set O_u^R and O_u^S for each user and then computed the gradients on the items in $S_u^R \cup S_u^S \cup O_u^R \cup O_u^S$. To prevent data imbalance problem, the sizes of sampled data S_u^R and S_u^S is equal to O_u^R and O_u^S, respectively. So the overall complexity of the learning algorithm is linear in the size of O and the number of latent dimensions K.

Now the matrix $\hat{Q}$ is a filled matrix with users' concrete rating scores for the entire fictitious items that have been observed from both rating and review given for the context-based fictitious item by the user. By this proposed methodology the most corresponding relevant item for the user will be recommended by the prediction of concrete rating scores.

In-order to give recommendations in the online phase, the 2D concrete score matrix $\hat{Q}$ with fictitious items (from F) is reverted or mapped to a multidimensional matrix Q with real items (from I) and contextual situations (from C) accordingly.

3.6 On-Line Activity—Recommendation Process

The Algorithm for recommendation generation for the active user u_a as follows:

Algorithm 5: *Recommendation Process.*

Input: Active user u_a with preference vector < *rating, item_review* >, Offline predicted complete multidimensional concrete score matrix Q, N represents the total items to be recommended.

Output: List of Top –N recommendation Item for the user u_a in current contextual situation c.

Functional Specification:

1. Generate the current context situation $c \in \left(C_1 \times C_2 \times ... \times C_n\right)$ of the active user u_a from the current date by the algorithm (2).
2. If active user u_a already has some transaction history and with no alteration, then send top N real items with the highest concrete scores as recommendations from $\hat{R}_{u_a}$, the predicted vector of user u_a in the matrix Q to the recommender agent in the current contextual situation c.
3. If active user u_a is new and has a preference vector < *rating, item_review* > of size m. Then for u_a perform the processes as mentioned in Algorithm (2), (3) and (4) and generate the concrete score using CSSAE then send top N real items with the highest predicted

concrete scores as recommendations to the recommender agent in the current contextual situation c.
4. Send top N most transacted items in the current contextual situation c of the active user to the recommender agent, if there is no rating vector (New user cold start) for the active user ua.
5. If active user ua already has some transaction history and with some alteration, then perform Step 3.

4 Experimental Evaluation

4.1 Dataset Description

The 5-core Amazon datasets (2018) are organized into several product categories and broadly used in numerous previous works [18, 37]. To evaluate the performance of CSSAE, four sub-datasets from 5-core Amazon product review datasets are considered. The information of the datasets is shown in Table 6. Each dataset contains information such as verified (indicates whether the customer is a verified customer or not), Style (which indicates size, flavor, etc. of the product purchased by the user), and vote (number of the useful vote sent by the user) which is used as the users' side information in the proposed methodology. It also contains product metadata for each category such as product features (color, size), price, and technical details of each item.

4.2 Baselines

The proposed CSSAE model is compared against the following baselines methods from each of the following categories as Neighborhood-based Methods (ItemKNN), Model-based Methods (SVD), and Deep Learning Methods (I-AutoRec, CDAE, SDDRS). Each of the baseline models is explained as follows:

1. Item-KNN [38]: Item k-nearest neighborhood collaborative filtering model that estimates the rating for the active user based on the similarity between the k-nearest items.
2. SVD [39]: A matrix factorization technique termed Singular Value Decomposition (SVD) is a dimensionality reduction technique in the active item's neighborhood for Item-Based Collaborative Filtering.
3. I-AutoRec [22]: Collaborative denoising autoencoder that takes only the k-hot encoded rating vector of items as input.

Table 6 Statistics of Amazon 5-core datasets (2018)

Dataset	No. of user	No. of item	Rating and review
Cell_Phones_and_Accessories_5	157,212	48,186	1,128,437
Musical_Instruments_5	27,530	10,620	231,392
Office_Products_5	101,501	27,965	800,357
Automotive_5	193,651	79,437	1,711,519

4. CDAE [23]: Collaborative denoising autoencoder that appends the additional input in the form of user hidden factor into the rating vector for the recommendation.
5. SDDRS [28]: Stacked Discriminative Denoising Auto-Encoder which integrates matrix factorization method to append side information with rating information.

4.3 Evaluation Metric

In recommendation algorithms, accuracy measure is the term used to estimate the performance of the system. To demonstrate the greatest improvement in top-N predictions that are made with high relevant items by combining rating and reviews is estimated using decision-support information retrieval measures such as precision (proportion of relevant items among the retrieved items), recall (proportion of the total amount of relevant items that were retrieved). Here, N items are given to the user to match their interest from training data that was unrated. Therefore, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) are adopted in our experiments to estimate the accuracy of top-N item recommendation.

Let I_u represents the number of items that have been rated by user u in test data, and $\hat{I}_{N,u}$ denotes the N predicted items for user u with the highest value. Then the definition of Precision and Recall:

$$\text{Precision@}N = \left|I_u \cap \hat{I}_{N,u}\right| \Big/ N \tag{6}$$

$$\text{Recall@}N = \left|I_u \cap \hat{I}_{N,u}\right| \Big/ \hat{I}_{N,u} \tag{7}$$

To determine more consistent performance at all positions of recommended items, the Average precision value gives higher weights to the items adopted by the user in test results. With N recommended items, AP@N is defined as the weighted average of precisions as:

$$AP@N = \frac{\sum_{k=1}^{N} \text{Pr}\,ecision@k \times rel(k)}{\min\left\{N, \left|\hat{I}_{N,u}\right|\right\}} \tag{8}$$

where Precision@k is I_u's precision value at k in the top-N, and an indicator function $rel(k) = 1$ if the item at rank k is taken, else zero. Ultimately, (MAP@N) is the average of all user's AP scores.

For every user, *Discounted Cumulative Gain (DCG@N)* is specified as:

$$DCG@N = \sum_{r=1}^{N} \frac{2^{rel(r)-1}}{\log_2(r+1)} \tag{9}$$

NDCG@N is the normalized discounted cumulative gain at N, (i.e.), the normalized DCG over the ideal *iDCG@N* which acquires the perfectly recommended items' position into account.

4.4 Experimental Setup

The experimentation was conducted on all four data sets to provide the implementation details. The experiment is done by splitting the dataset as 60% data in training, 20% for cross-validation, and 20% data for testing. The proposed model CSSAE is implemented in python Keras API from the TensorFlow library. Here the values in the rating vector are normalized from -1 to $+1$ in all the datasets and already the predicted sentiment score ranges from -1 to $+1$ and the feature vector which is the binary representation of user features are considered as the side information. The proposed model is trained on a training set (dual inputs i.e. rating and sentiment scores) are fed to a CSSAE algorithm together with the users' side information in the subsequent layer which produces a recommendation model that can be used to generate new predictions and then evaluated on the cross-validation set for selecting the appropriate values for the hyper-parameters. Finally, the model is evaluated on the test set that is fed to the learned model where predictions are generated for each user-item pair and then used in the top N evaluation metric as in Eq. (8) and Eq. (9) to produce evaluation results.

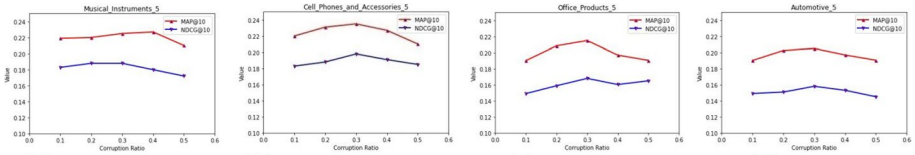
For further analysis, we examine how different parameter settings may impact the performance of the proposed CSSAE model using the scikit-learn, grid search capability. Particularly, the model sensitivity to the noise variance, the size of the hidden units, the learning rate, the epoch, and the algorithm for optimization are evaluated accordingly. It is observed that the hyper-parameters of noise variance and size of the hidden units make an impact on the outcomes, and hence the additional experiments were conducted to determine the optimal value of corruption ratio and hidden unit size as mentioned in 4.4.1. In the training phase, the model was found to converge when the epoch value was 250 and altered learning rates (0.1, 0.02, 0.002, 0.001, 0.01) were experimented with, and determined to be optimum when the learning rate $\eta = 0.001$. Similarly, the network is optimized with Adam Optimizer with a minibatch of size 30. We adopted hyperbolic tangents as a transfer function since the dataset are normalized from -1 to $+1$, and weight decay $l_2(0.002)$ is added for regularization.

For a fair comparison, we use Python scikit *Surprise* package prediction algorithms, to estimate the value of ItemKNN and SVD model. For ItemKNN, the number of neighbors *k*: [5, 10, 16, 26] and for SVD the parameters such as the number of iteration of the SGD procedure *n_epochs*: [100, 150, 200, 250, 300], the learning rate *lr_all*: [0.002, 0.005, 0.001, 0.01], the regularization term *reg_all*: [0.3, 0.4, 0.6, 0.8]} are fine-tuned to find the best value using grid search. The rest of the baselines using autoencoder are implemented using python Keras API. For I_AutoRec and CDAE, we use the hyper-parameter setting as the same as the proposed method as mentioned above. For SDDRS, we set corruption ratio $\alpha = 0.2$, regularization coefficients $\lambda_1 = 0.25$, $\lambda_2 = 1$, $\lambda_3 = 30$, β of indicator matrix I is set to 10, epochs = 500, and learning rate $\eta = 0.0002$.

4.4.1 Parameter Sensitivity

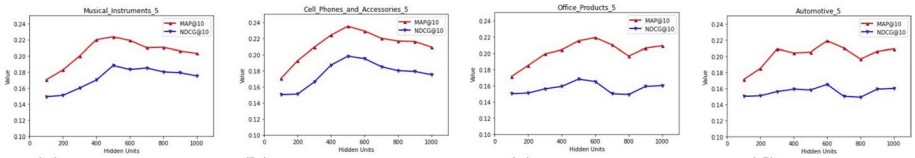
(a) *Impact of Corruption Ratio*

The issue of overfitting in the neural network is effectively addressed by corruption ratio with training samples with fewer sizes [40]. Hence to observe the impacts of corruption



(a) Musical_Instruments **(b)** Cell_Phones_Accessories **(c)** Office_Products **(d)** Automotive

Fig. 3 Corruption ratio selection



(a) Musical_Instruments **(b)** Cell_Phones_Accessories **(c)** Office_Products **(d)** Automotive

Fig. 4 Hidden unit size selection

ratio, a tuning parameter α that is used to constrict the consequence of adding noise on the stacked autoencoder is varied with different values. The initial value α is set to 0.1 and then it was gradually raised to 0.5. Figure 3 shows that if α increases with high value its performance gets degraded. The proposed model has the best performance when the α parameter is 0.25 on all the data sets.

(b) *Impact of Hidden Unit Size*

The latent space representation in the hidden layer from the input has the greatest influence on the models' accuracy i.e., determined by the number of nodes in the hidden layer. To determine the optimal number of nodes in the hidden layer, the model is with varying values of k is experimented with. Fig. 4 depicts the experimental results. It is observed from the plots, that when k=600, the performance of the model remains stable and there is no major increase in the performance beyond it at the rate of increased training time.

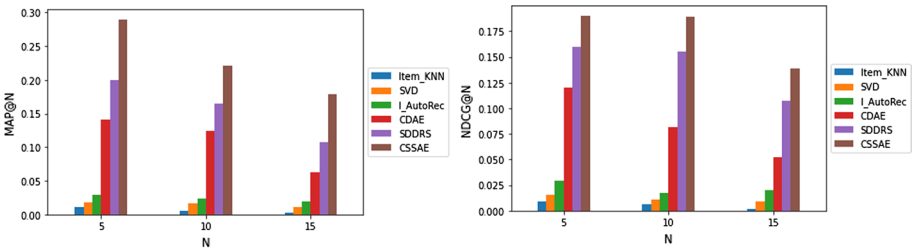
4.5 Results and Discussion

We conduct a series of experiments to evaluate the performance of the CSSAE model and the baselines are evaluated and recorded in terms of *MAP@10* using Eq. (8) and *NDCG@10* using Eq. (9) in Table 7 and Figs. 4, 5, 6, 7 and 8 shows the *MAP@N* and *NDCG@N* scores on 4 datasets as mentioned in Table 6.

From the results of Table 7, it is observed that from the compared baseline, the proposed model that combines both context-based rating and review into deep learning-based methods together are found to perform better than the other methods. According to the results of MAP@N and NDCG@N in Fig. 4, 5, 6, 7 and 8, CSSAE consistently outperforms other compared methods in all four datasets. The MAP@10 score and NDCG@10 score of CSSAE are at least 26% better than those of the autoencoder based methods such as CDAE and SDDRS since it takes only the user preference as ratings which do not reflect the concrete preference in many cases. The SDDRS method performs better than CDAE

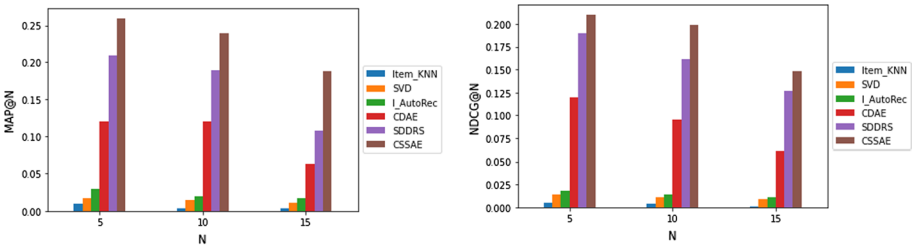
Table 7 The performance comparison of all methods in terms of map@10 and Ndcg@10

Dataset	Item-KNN	SVD	I-AutoRec	CDAE	SDDRS	CSSAE	Improv (%)
MAP@10							
Musical_Instruments_5	0.0052	0.0163	0.0239	0.1247	0.1649	***0.2207***	***33.9***
Cell_Phones_and_Accessories_5	0.0041	0.0145	0.0200	0.1199	0.1893	***0.2393***	***26.4***
Office_Products_5	0.0074	0.0201	0.0313	0.1314	0.1715	***0.2194***	***27.9***
Automotive_5	0.0062	0.0121	0.0307	0.1408	0.1563	***0.2025***	***29.6***
NDCG@10							
Musical_Instruments_5	0.0061	0.0109	0.0175	0.0816	0.1551	***0.1893***	***22.1***
Cell_Phones_and_Accessories_5	0.0035	0.0105	0.0144	0.0956	0.1609	***0.1987***	***23.5***
Office_Products_5	0.0081	0.0157	0.0234	0.1052	0.1243	***0.1643***	***32.2***
Automotive_5	0.0052	0.0102	0.0251	0.0867	0.1154	***0.1523***	***31.1***



(a) MAP@N on Musical_Instruments **(b)** NDCG@N on Musical_Instruments

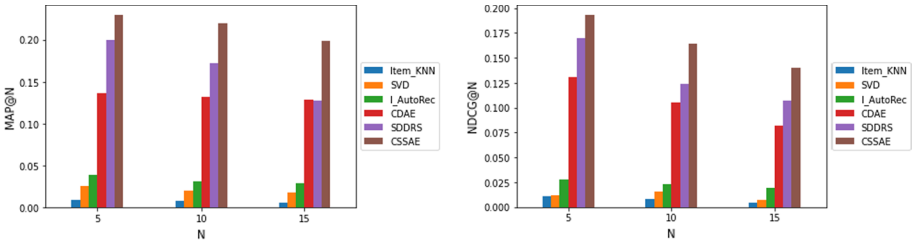
Fig. 5 The comparison of the performance of the Musical_Instruments_5 data set



(a) MAP@N on Cell_Phones_and_Accessories **(b)** NDCG@N on Cell_Phones_and_Accessories

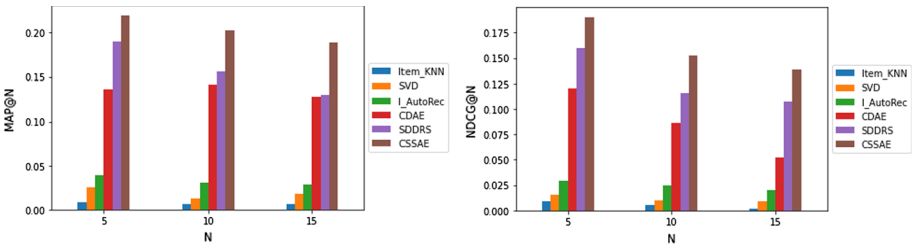
Fig. 6 The comparison of the performance of Cell_Phones_and_Accessories_5

since it is based on the stacked denoising autoencoder that uses the MF technique, and the results imply its importance to recommender systems. The deep learning-based methods such as CDAE and I-AutoRec perform at-least 25% better than SVD and Item-KNN on all metrics since these methods are unable to learn the nonlinear relationship from data. The MAP@10 score and NDCG@10 score of SVD outperform better than neighborhood-based method Item-KNN.



(a) MAP@N on Office_Products **(b)** NDCG@N on Office_Products

Fig. 7 The comparison of the performance of Office_Products_5



(a) MAP@N on Automotive **(b)** NDCG@N on Automotive

Fig. 8 The comparison of the performance of Automotive_5

The proposed method CSSAE performed better on all the datasets which ensures that it gives Top N recommendation that exactly matches the current context situation of the active user which are generated using predicted concrete user preferences that it learns from both user-specific rating and review for a context-based item using the Stacked Auto-Encoder framework that incorporates the users' side information as an additional input to the sparse input vector which tackles cold start problem and extensively outperformed the state-of-the-art baseline methods.

4.6 Ablation Study

An ablation analysis is conducted to further analyze the efficiency of the proposed CSSAE model, based on the contribution of the various components. To do this, two variants of the model have been developed where all the experimental parameters are the same on each data set. Such various variants are further compared to the default settings based approach as defined in Sect. 3. Figure 9 shows the ablation effects on all the datasets.

- *Default CSSAE Method:* In this setting, all the prevailing components of the CSSAE model are used in Top-N recommendation generation as illustrated in Sect. 3.
- *Without time feature:* In this variant, the process of context generation from the date of purchase as explained in Algorithm(2) and the process of item splitting to create context-based fictitious items *(F)* by Algorithm(3) has been removed and hence directly the original items *(I)* from the dataset are used as input into the model. Therefore, in

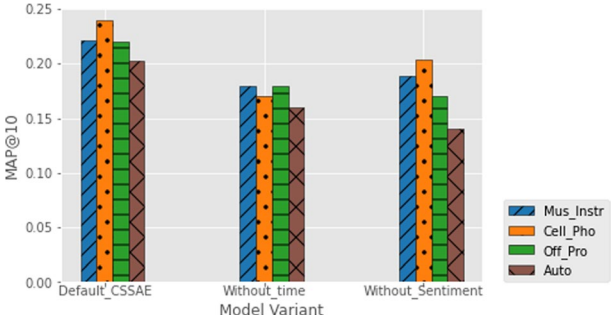


Fig. 9 MAP@10 results of the model variants on all the datasets

the online recommendation phase, the items are recommended without considering the current context situation of the user. This is particularly performed to examine the impact of the user rating in a particular context situation associated with the item. From this analysis, it can be concluded from Fig. 9 that, the model without time feature cannot learn the nonlinear relationship from the user preference that is given exactly to a context-based item. Hence, considering the context-based items' user preference in the proposed CSSAE model increases the performance by at least 22% on all datasets by predicting and recommending the exact item depending upon the context situation of the active user. This clearly shows the influences of user-fictitious item interaction for better learning of user and item representation.

- *Without Sentiment Score:* In this variant, the process of sentiment score prediction from reviews by Algorithm (4) is removed and only the rating value given by the user to a context-based fictitious item is considered for rating prediction. To do this, the expanded dual input layer in the CSSAE model is detached and thereby the user emotions expressed via text are not considered together with rating to predict the concrete preference of the user. Hence it is impossible for the model to significantly explore deep semantic associations between the rating and sentiment. Here the user-item interaction occurs only with the rating value at the prediction layer which consequently degrades the performance of the model by at least 19% when compared to the default CSSAE method on all datasets as shown in Fig. 9 since it does not consider the concrete preference of the user by combining both the rating and sentiment value into stacked autoencoder.

Due to space limitations, we have reported only the results of *MAP@10* on all datasets in Fig. 9. From Fig. 9, it is concluded that without using time feature and sentiment score in the proposed CSSAE methodology degrades the performance in recommending the most appropriate item depending upon the users' context situation. Besides, the model CSSAE integrates the users' side information based on sparse ratings to enrich each user features to deal with the cold start problem, the achieved results can show that the model CSSAE can generate accurate Top-N recommendation to the user.

4.7 Scalability

One major challenge that most collaborative filtering has to overcome is scalability, as the dataset mostly has millions of users and items. An effective algorithm needs to be trained in a decent length of time and produce fast Recommendations during the evaluation phase. All our

experiments are carried out with one NVIDIA Tesla K40C GPU which can train Large Neural Networks in just a few minutes. For Musical_Instruments_5 and Office_Products_5 datasets, each epoch takes only 3.1 s and 3.9 s respectively, and each run takes 250 epochs. It takes around 5.7seconds and 6.3 s for the Cell_Phones_and_Accessories_5 and Automotive_5 dataset and needs 250 epochs to get conventional performance. This shows that CSSAE is very scalable since Automotive_5 datasets are far larger than the other three datasets.

The efficiency measure in terms of average training duration demonstrates that SVD is the method of dimensionality reduction, such that the computation time outperforms other methods while the accuracy is poor. As context-based fictitious item training is taken place in the CSSAE, the computation time is comparatively high with higher accuracy with exact context situation based recommendation generation compared to the other systems. I_Autorec and CDAE have medium performance in terms of time and accuracy. The training duration of SDDRS is comparatively low whereas the accuracy is low than the proposed method. The Item-KNN model has a very low accuracy relative to all other approaches in all the datasets, with a longer test period relative to its training time.

5 Conclusion and Future Work

In this paper, context-specific sentiment based stacked autoencoder (CSSAE) model was proposed to enhance the top-N recommendation in a context-aware environment by integrating both the users' rating and review into the prediction of users' concrete preferences using Stacked Autoencoder. To do so, the proposed system extracts contextual information and turns out the item to a context-based fictitious item for which the concrete user preference is predicted by integrating an additional layer which merges both the rating and reviews as an input layer by expanding the autoencoder and also to overcome sparsity users' side information is appended in each layer of the network. Compared with all the top-N recommendation generation methods which consider only the rating as an actual preference, the proposed method obtains concrete preference of the user by combining rating with reviews thereby enhances the context-aware top-N prediction accuracy on all datasets.

The proposed CSSAE method utilizes item splitting, a contextual prefiltering technique which assumes that certain items may have different estimation in distinct contexts. Consequently, it was noticed that item splitting is constructive, but most of the real-world data are inadequate as they lack exact contextually labeled ratings. This model has been validated to perform well in the e-commerce recommendations, but it can still be applied in any domain that includes user ratings and text feedback. Our work can be further enhanced to generate polarity scores from reviews using deep learning architectures such as LSTM-RNN, CNN, etc. to improve the accuracy of sentiment analysis, and incorporating other types of user preference criteria might reflect a more specific and focused Top-N items recommendation for a particular user.

Appendix

In this section, the basic architectural components of the auto-encoder have been described and the changes introduced to implement the proposed *Context-specific Sentiment based Stacked AutoEncoder* (CSSAE) method to fit with the recommendation problem is

described in detail with an example with a table provided with the parameters of each layer and the layer input shapes.

An autoencoder consists of three components

Encoder: An encoder is a feedforward, fully connected neural network that a compress i.e. encodes the input into a latent space representation.
Code: This part of the network contains the reduced representation of the input that is fed into the decoder.
Decoder: Decoder is also a feedforward network like the encoder and has a similar structure to the encoder. This network is responsible for reconstructing the input back to the original dimensions from the code.

The following are the changes made to the CSSAE model to fit with the recommendation problem

However, to consider the multiple feedback scenario, each context-based item has multiple feedback that is required as input to the network. Therefore, as shown in Fig. 2 the traditional stacked autoencoder was expanded by integrating an additional layer which holds two input layer for both the rating and sentiment data. Thus, the additional layer acts as the input layer in the context-specific sentiment based stacked autoencoder, and Table 8 provides the parameters of each layer and the layer input shapes.

Note: The shape of the input layer is the total number of context-based items in the training set is **19366.**

The first layer acts as a dual input layer i.e. two input layers (R_Score[0][0] and S_Score[0][0]) which are linked to their respective **custom layer** where elementwise_weights are trained for each node. Hence the particular node in the custom layer elementwise_weights_14[0][0] which is connected to R_Score has the value $r_{ij} \times w_j^1$ and elementwise_weights_15[0][0] which is connected to S_Score has the value $s_{ij} \times w_j^2$ and both which in turn is fed into the **lambda layer** for merging two inputs which hold $r_{ij} \times w_j^1 + s_{ij} \times w_j^2$.

Hence corresponding item nodes in the following intermediate layer (lambda_7) has the value as mentioned in Eq. (3)

$$r_{ij}^I = r_{ij} \times w_j^1 + s_{ij} \times w_j^2$$

where r_{ij}^I be the combined rating value of an item j rated by user i which is obtained from both r_{ij}, the rating and s_{ij} the sentiment score obtained from the review specified by user i to item j.

This intermediate layer (lambda_7) which in turn is linked to consecutive M encoding layers that are being concatenated with User_sideInfm that are employed to find out the hidden representation of the items. The final encoding layer (LatentSpace) is linked to M successive decoding layers that are used to decode the hidden factors acquired from corresponding encoders. The Final decoding layer (UserScorePred) acts as an output where the actual concrete ratings of the items are predicted.

We have selected the intermediate layer (lambda_7) and its successive set of layers as an autoencoder because the nodes in the intermediate layer (lambda_7) has the merger of values of both the rating and sentiment scores as mentioned in Eq. (3)

Table 8 The following table provides the parameters of each layer and the layer input shapes

Table. 8: The following table provides the parameters of each layer and the layer input shapes.

```
Model: "functional_9"
__________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
R_Score (InputLayer)         [(None, 19366)]           0
__________________________________________________________________________________________________
S_Score (InputLayer)         [(None, 19366)]           0
__________________________________________________________________________________________________
elementwise_weights_14 (Element (None, 19366)          19366     R_Score[0][0]
__________________________________________________________________________________________________
elementwise_weights_15 (Element (None, 19366)          19366     S_Score[0][0]
__________________________________________________________________________________________________
lambda_7 (Lambda)            (None, 19366)             0         elementwise_weights_14[0][0]
                                                                 elementwise_weights_15[0][0]
__________________________________________________________________________________________________
User_sideInfm (InputLayer)   [(None, 30)]              0
__________________________________________________________________________________________________
EncLayer1_sideinfm_concat (Conc (None, 19396)          0         lambda_7[0][0]
                                                                 User_sideInfm[0][0]
__________________________________________________________________________________________________
EncLayer1 (Dense)            (None, 7589)              147203833 EncLayer1_sideinfm_concat[0][0]
__________________________________________________________________________________________________
Dropout_1 (Dropout)          (None, 7589)              0         EncLayer1[0][0]
__________________________________________________________________________________________________
latent_sideinfm_concat (Concate (None, 7619)           0         Dropout_1[0][0]
                                                                 User_sideInfm[0][0]
__________________________________________________________________________________________________
LatentSpace (Dense)          (None, 3028)              23073360  latent_sideinfm_concat[0][0]
__________________________________________________________________________________________________
Dropout_2 (Dropout)          (None, 3028)              0         LatentSpace[0][0]
__________________________________________________________________________________________________
DecLayer1 (Dense)            (None, 7589)              22987081  Dropout_2[0][0]
__________________________________________________________________________________________________
UserScorePred (Dense)        (None, 19366)             146987940 DecLayer1[0][0]
==================================================================================================
Total params: 340,290,946
Trainable params: 340,290,946
Non-trainable params: 0
```

$$r_{ij}^{I} = r_{ij} \times w_j^1 + s_{ij} \times w_j^2$$

Hence this intermediate layer acts as the input to the autoencoder, which encodes the input to find out the hidden representation of the items. Therefore, the latent feature vector (code) in the stacked autoencoder fits the ratings and reviews simultaneously. The final decoding layer (UserScorePred) acts as an output where the actual concrete ratings of the context-based items are predicted.

Note

In the proposed methodology, Sentiment Analysis is performed using Textblob on Users' review on context-based items in-order to predict the Numerical sentiment score. This numerical sentiment score is given as the input into stacked autoencoder, a fully-connected neural network that was expanded by integrating an additional layer that holds two input layers for both the rating and sentiment data, where the training takes place end-to-end.

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
2. Wang CD, Deng ZH, Lai JH, Philip SY (2018) Serendipitous recommendation in e-commerce using innovator-based collaborative filtering. IEEE Trans Cybern 49(7):2678–2692
3. T., Kun, Shuyan C., and Aemal J. K. (2018) Personalized travel time estimation for urban road networks: a tensor-based context-aware approach. Expert Syst Appl 103:118–132
4. Macedo AQ, Marinho LB, Santos RL (2015) Context-aware event recommendation in event-based social networks. In: Proceedings of the 9th ACM conference on recommender-systems, pp 123–130
5. Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Recommender systems-handbook. Springer, Boston, pp 217–253
6. Baltrunas L, Ricci F (2009) Context-based splitting of item ratings in collaborative filtering. In: Proceedings of the third ACM conference on recommender systems, pp 245–248
7. Raghavan S, Gunasekar S, Ghosh J (2012) Review quality aware collaborative filtering. In: Proceedings of the sixth ACM conference on recommender systems, pp 123–130
8. Cheng Z, Ding Y, Zhu L, Kankanhalli M (2018) Aspect-aware latent factor model: rating prediction with ratings and reviews. In: Proceedings of the 2018 world wide web conference, pp 639–648
9. Mobasher B, Burke R, Bhaumik R, Williams C (2007) Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness. ACM Trans Internet Technol 7(4):23
10. Hart-Davidson W, Michael ML, Christopher K, Michael W (2010) A method for measuring helpfulness in online peer review. In: Proceedings of the 28th ACM international conference on design of communication, pp 115–121
11. Zhang Y, Zhang H, Zhang M, Liu Y, Ma S (2014) Do users rate or review? Boost phrase-level sentiment labeling with review-level sentiment classification. In: Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval, pp 1027–1030
12. Cambria E, Schuller B, Xia Y, Havasi C (2013) New avenues in opinion mining and sentiment analysis. IEEE Intell Syst 28(2):15–21
13. Pappas N, Popescu-Belis A (2013) Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, pp 773–776
14. Zhang Y, Lai G, Zhang M, Zhang Y, Liu Y, Ma S (2014) Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In: Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval, pp 83–92
15. Faridani S (2011) Using canonical correlation analysis for generalized sentiment analysis, product recommendation and search. In: Proceedings of the fifth ACM conference on recommender systems, pp 355–358
16. Ganu G, Elhadad N, Marian A (2009) Beyond the stars: improving rating predictions using review text content. In: WebDB, vol 9, pp 1–6
17. Qiu J, Liu C, Li Y, Lin Z (2018) Leveraging sentiment analysis at the aspects level to predict ratings of reviews. Inf Sci 451:295–309
18. Cen Y, Zou X, Zhang J, Yang H, Zhou J, Tang,J (2019) Representation learning for attributed multiplex heterogeneous network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1358–1368
19. Huang R, Wang N, Han C, Yu F, Cui L (2020) TNAM: a tag-aware neural attention model for Top-N recommendation. Neurocomputing 385:1–12

20. Unger M, Tuzhilin A, Livne A (2020) Context-aware recommendations based on deep learning frameworks. ACM Trans Manag Inform Syst 11(2):1–15
21. Wu B, Wen W, Hao Z, Cai R (2020) Multi-context aware user-item embedding for recommendation. Neural Netw 124:86–94
22. Sedhain S, Menon AK, Sanner S, Xie L (2015) Autorec: autoencoders meet collaborative filtering. In: Proceedings of the 24th international conference on world wide web, pp 111–112
23. Wu Y, DuBois C, Zheng AX, Ester M (2016) Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the ninth ACM international conference on web search and data mining, pp 153–162
24. Wang H, Wang N, Yeung DY (2015) Collaborative deep learning for recommender systems. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1235–1244
25. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
26. Strub F, Mary J, Gaudel R (2016) Hybrid collaborative filtering with autoencoders. arXiv: 1603.00806
27. Wang M, Wu Z, Sun X, Feng G, Zhang B (2019) Trust-aware collaborative filtering with a denoising autoencoder. Neural Process Lett 49(2):835–849
28. Wang K, Xu L, Huang L, Wang CD, Lai JH (2019) SDDRS: stacked discriminative denoising autoencoder based recommender system. Cogn Syst Res 55:164–174
29. Dessì D, Dragoni M, Fenu G, Marras M, Recupero DR (2019) Evaluating neural word embeddings created from online course reviews for sentiment analysis. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing, pp 2124–2127
30. Phuong TM, Phuong ND (2019) Graph-based context-aware collaborative filtering. Expert Syst Appl 126:9–19
31. Ye Q, Law R, Gu B (2009) The impact of online user reviews on hotel room sales. Int J Hosp Manag 28(1):180–182
32. Feldman R (2013) Techniques and applications for sentiment analysis. Commun ACM 56(4):82–89
33. Poria S, Gelbukh A, Agarwal B, Cambria E, Howard N (2014) Sentic demo: a hybrid concept-level aspect-based sentiment analysis toolkit. In: ESWC 2014
34. Liu B (2010) Sentiment analysis and subjectivity. Handbook Nat Lang Process 2(2010):627–666
35. Loria S (2017) TextBlob: simplified text processing [a Python (2 and 3) library for processing textual data]
36. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, pp 1096–1103
37. Zheng L, Noroozi V, Yu PS (2017) 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, pp 425–434
38. Sarwar B, Karypis G, Konstan J, Riedl J (2001). Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web, pp 285–295
39. Vozalis MG, Margaritis KG (2007) Using SVD and demographic data for the enhancement of generalized collaborative filtering. Inf Sci 177(15):3017–3037
40. Scholkopf B, Platt J, Hofmann T (2006). Greedy layer-wise training of deep networks. In: International conference on neural information processing systems. MIT Press, pp 153–160

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.