Attentive Contextual Denoising Autoencoder for Recommendation

Yogesh Jhamb
Department of Computer Engineering
Santa Clara University
Santa Clara, CA
yjhamb@scu.edu

Travis Ebesu
Department of Computer Engineering
Santa Clara University
Santa Clara, CA
tebesu@scu.edu

Yi Fang
Department of Computer Engineering
Santa Clara University
Santa Clara, CA
yfang@scu.edu

ABSTRACT

Personalized recommendation has become increasingly pervasive nowadays. Users receive recommendations on products, movies, point-of-interests and other online services. Traditional collaborative filtering techniques have demonstrated effectiveness in a wide range of recommendation tasks, but they are unable to capture complex relationships between users and items. There is a surge of interest in applying deep learning to recommender systems due to its nonlinear modeling capacity and recent success in other domains such as computer vision and speech recognition. However, prior work does not incorporate contexual information, which is usually largely available in many recommendation tasks. In this paper, we propose a deep learning based model for contexual recommendation. Specifically, the model consists of a denoising autoencoder neural network architecture augmented with a context-driven attention mechanism, referred to as *Attentive Contextual Denoising Autoencoder (ACDA)*. The attention mechanism is utilized to encode the contextual attributes into the hidden representation of the user's preference, which associates personalized context with each user's preference to provide recommendation targeted to that specific user. Experiments conducted on multiple real-world datasets from *Meetup* and *Movielens* on event and movie recommendations demonstrate the effectiveness of the proposed model over the state-of-the-art recommenders.

KEYWORDS

Recommender Systems, Denoising Autoencoders, Attention Mechanism

ACM Reference format:
Yogesh Jhamb, Travis Ebesu, and Yi Fang. 2018. Attentive Contextual Denoising Autoencoder for Recommendation. In *Proceedings of 2018 ACM SIGIR International Conference on the Theory of Information Retrieval, Tianjin, China, September 14–17, 2018 (ICTIR '18),* 8 pages.
https://doi.org/10.1145/3234944.3234956

1 INTRODUCTION

The information overload caused by the deluge of data has resulted in the need for recommender systems. The goal of a recommender system is to predict the unknown preferences of a user based on the known preferences of that user on certain items. Classic methods for recommender systems, such as content-based and collaborative filtering, have been effective in the past and they have offered a reasonable level of performance. However, these methods lack the ability to model complex nonlinear relationships that usually accompany the user-item interaction. With the recent success of deep learning in computer vision and speech recognition [8], there has been a surge of interest in applying deep learning methods to recommendation tasks [33]. The existing work in this domain is still quite limited, and furthermore, it does not utilize contextual information that is largely present in the real-world scenarios. Context provides additional information to the user-item interaction, which in turn improves the quality of the recommendation [5]. The attention mechanism [1] provides an intuitive way to incorporate context into the user-item interaction. Motivated by these factors, we propose a novel model for personalized recommendation based on the denoising autoencoder augmented with a context-driven attention mechanism. We call this model the *Attentive Contextual Denoising Autoencoder (ACDA)*.

Autoencoders [8] are feed-forward neural networks capable of learning a representation of the input data, also known as codings. The codings typically learnt by an autoencoder are of much lower dimensionality than the original input. Denoising autoencoders [8] are a variant of the basic autoencoder that add noise to the input and train the network to recover the original input at the output layer. This forces the network to discover robust features in the data representation, and prevents the model from learning the trivial identity function. The autoencoder architecture makes it suitable for use in recommender systems as the hidden layer captures the latent representation of the data, allowing the model to learn the latent factors associated with the user-item interaction. It has been shown [30] that the denoising autoencoder architecture is a nonlinear generalization of latent factor models [14, 18], which have been widely used in recommender systems. Therefore, we utilize denoising autoencoder as the main building block for the proposed *ACDA* model.

Context provides an added dimension to real-world applications. Recommender systems for movies, products, point-of-interests and services utilize context to provide a meaningful personalized recommendation [5]. For example, genre such as horror, drama, thriller, comedy etc., is an important context for movie recommendation as people generally like the same type of movies. Location and time-of-day are useful context to consider while recommending point-of-interests. There is existing work in the literature that provides contextual recommendations [12, 20, 35]. The *ACDA* model incorporates contextual information via the attention mechanism for personalized recommendation. We apply the *ACDA* model to two real-world problems of event recommendation [11] and movie recommendation. For the event recommendation task, we use the user *group* and event *venue* as the contextual attributes, whereas the movie *genre* is used as the contextual attribute for the movie recommendation task.

The attention mechanism has been instrumental in dealing with structured problems, such as machine translation and caption generation [1, 10, 27]. The objective of the mechanism is to highlight, or focus attention on, a certain subset of the data. The attention mechanism accepts a certain input and a context that accompanies the input. The output of the attention mechanism is considered as a summary of the input focusing on the information linked to the provided context. The attention mechanism is generally applied for two reasons–first, to provide for efficient processing of a high-dimensional input by processing only subsets of the input, and second, to focus on specific parts of the input in terms of its relevance. A classical example of the use of the attention mechanism is image captioning, where the mechanism focuses on certain subsets of the image to generate the suitable caption. The *ACDA* model utilizes the attention mechanism to apply the contextual attributes to the hidden representation of the user's preference. This helps the model to associate personalized context with each user's preference to provide recommendation targeted to that specific user.

The *ADCA* model accepts the user's preference on existing items as input, which includes both positive and negative instances. The input is partially corrupted to learn a robust representation of the data. The input is mapped to an internal representation of lower dimensionality by the hidden layer, where the contextual parameters are applied via the attention mechanism to focus on the user-specific relevant context. The output of the model is the reconstructed user input, which is the predicted preference of the user. The model is trained to minimize the loss between the original corrupted input and the reconstructed input generated at the output layer. We use multiple real-world datasets to conduct comprehensive experiments for the proposed *ACDA* model. The datasets for the event recommendation task are obtained from *Meetup*[1], a popular Event-Based Social Network (EBSN). We use the publicly available *Movielens* 100K dataset for the movie recommendation task. The experimental results show that the proposed model performs better than current state-of-the-art baselines. The main contributions of this paper can be summarized as follows.

- We propose a novel *Attentive Contextual Denoising Autoencoder (ACDA)* model for recommendation. To the best of our knowledge, this is the first study that attempts to utilize the contextual information via attention mechanism in the deep architectures.
- We thoroughly evaluate our proposed approach on real-world datasets from *Meetup* and *Movielens* on two different tasks: event recommendation and movie recommendation. The results demonstrate the effectiveness of the proposed model compared to the other state-of-the-art models. The code and data are available at https://github.com/yjhamb/acda.git.

[1]http://www.meetup.com

2 RELATED WORK

2.1 Denoising Autoencoders for Recommender Systems

Recently, a surge of interest in applying deep learning to recommendation systems has emerged. Neural Matrix Factorization [9] address implicit feedback by jointly learning a matrix factorization and a feedforward neural network. Wang et al. [29] unify the generative and discriminative methodologies under the generative adversarial network [8] framework for item recommendation, and question answering. A recent survey [33] provides a comprehensive overview of deep learning for recommender systems.

Autoencoders [8] have been a popular choice of deep learning architecture for recommender systems. Specifically, denoising autoencoders [8] are based on an unsupervised learning technique to learn representations that are robust to partial corruption of the input pattern [26]. This eventually led to denoising autoencoders being used for collaborative personalized recommenders. One of the early works that applied deep learning to recommender systems is based on the Restricted Boltzmann Machines (RBM) [22]. The authors of the RBM study propose a method for rating prediction that uses Contrastive Divergence as the objective function to approximate the gradients. Wu et al. [30] propose a collaborative denoising autoencoder model that utilizes an additional input encoding for the user latent factor for recommendation based on implicit feedback. Chen et al. [3] introduced the marginalized denoising autoencoder model that offer better performance by reducing the training time. The AutoRec model [23] for collaborative filtering proposes two variants: user-based (U-AutoRec) and item-based (I-AutoRec) denoising autoencoders that respectively take the partially observed user vector or item vector as input. The study evaluates both models on the Netflix dataset and concludes that the I-AutoRec performs better than the U-AutoRec model due to the high variance in the number of user ratings. An existing study proposes a hierarchical Bayesian model called collaborative deep learning (CDL) [28], which is based on stacked denoising autoencoders. The CDL model tightly couples the deep representation learning of the content information and collaborative filtering for the ratings matrix in a unified model. Other forms of autoencoders have also been used for recommendation tasks. Li et al.[15] propose a variational autoencoder that learns the deep latent representation and the implicit relationship between users and items from ratings and content data. AutoSVD++ [34] is a recent study that combines contrastive autoencoders and matrix factorization to provide recommendations based on content data and implicit user feedback. Our model is different from the existing literature on denoising autoencoder based recommender systems, as we incorporate the contextual information via the attention mechanism into the denoising autoencoder architecture. This has not been studied in any existing work in the literature.

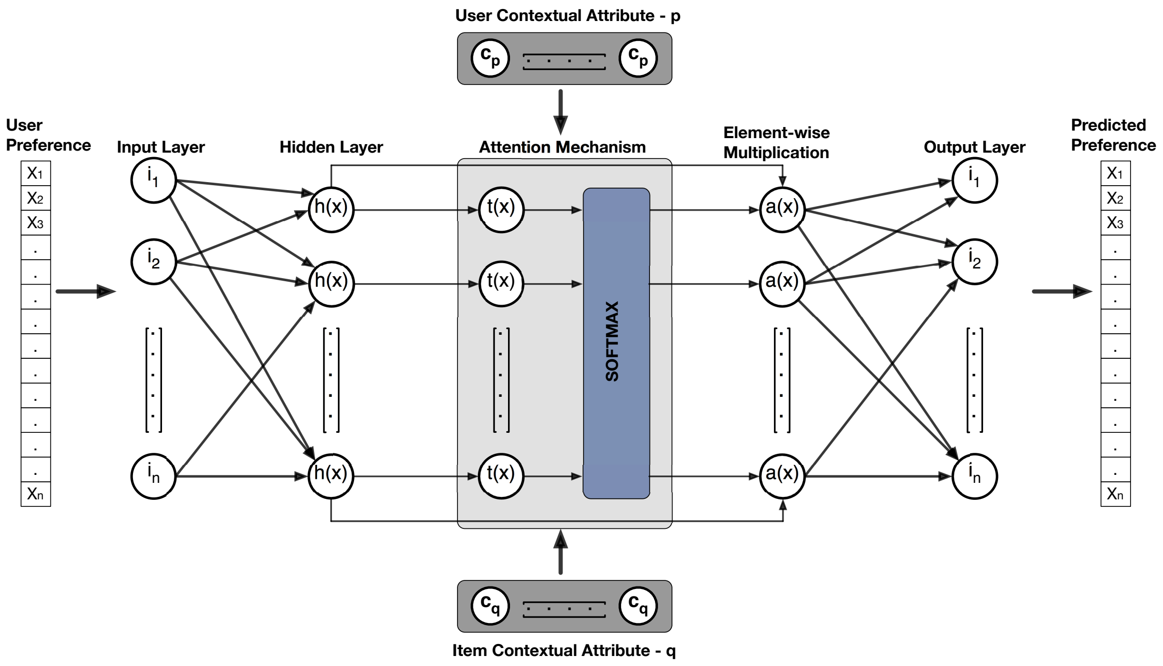

Figure 1: Attentive Contextual Denoising Autoencoder

2.2 Attention Mechanism

The attention mechanism has been widely adopted in deep learning for tasks related to image recognition and natural language processing [1, 32]. The significance of the attention mechanism has been highlighted in a study [4], where the mechanism has been applied to structured output problems that involve multimedia content. However, there has been minimal work in the literature that applies the attention mechanism to recommender systems. The existing works utilize the attention mechanism for recommender systems are based primarily on the Convolutional Neural Network (CNN) and the Recurrent Neural Network (RNN) architectures. Seo et al. [24] integrate a local and global attention mechanism with a CNN to model review text in the hopes of producing more interpretable recommendations. Likewise, Chen et al. [2] introduce item- and component-level attention to address multimedia collaborative filtering on implicit datasets. Factorization machines [21] combine higher order pairwise interactions between features, but treat each feature with equal weight. Motivated by this idea, Xiao et al. [31] learn to weigh the importance of each feature with an attention mechanism. Ebesu et al. [6] leverage a memory network with a neural attention mechanism to weigh the importance of each user in the neighborhood. The attention mechanism has also been used for hashtag recommendation using a CNN based model [7]. Yet another study [25] utilizes an attention-based CNN for personalized recommendation based on review text. Although there has been some recent work that utilizes the attention mechanism for recommender systems, to the best of our knowledge no work exists that integrates the attention mechanism with the denoising autoencoder architecture to process contextual data for personalized recommendation.

3 ATTENTIVE CONTEXTUAL DENOISING AUTOENCODER

First, we present the *Attentive Contextual Denoising Autoencoder (ACDA)* model, which is a generic framework for contextual recommendation. Later we explain how this flexible framework is applied to the event recommendation and movie recommendation tasks.

3.1 The Architecture

The proposed model, as illustrated in Figure 1, is based on the denoising autoencoder neural network architecture. The model takes as input a vector indicating the preference of a user u on all the items i in the dataset. Assuming that there are m users and n items, the autoencoder takes as input a vector $\mathbf{x} \in \mathbb{R}^n$, which is the known preference of the user u on the n items. We corrupt the input vector $\mathbf{x}$ using mask-out/drop-out corruption to obtain $\tilde{\mathbf{x}}$. The corruption method randomly overwrites some of the dimensions of $\mathbf{x}$ with 0 using the probability ρ. To offset the effect of the corruption of certain dimensions, we scale the remaining dimensions by applying a factor δ to the original value: $P(\tilde{\mathbf{x}}_\theta = 0) = \rho$; $P(\tilde{\mathbf{x}}_{\bar{\theta}} = \delta\tilde{\mathbf{x}}) = 1 - \rho$, where $\delta = 1/(1-\rho)$. The symbol θ denotes the dimensions that are set to 0, whereas $\bar{\theta}$ denotes the dimensions that are scaled. This corruption method is similar to the one used in [30]

The corrupted vector $\tilde{\mathbf{x}}$ is fed into the model to generate the latent hidden representation $h \in \mathbb{R}^k$ using the encoding function $e(\cdot)$. The dimensionality of the hidden representation is represented by $k \ll n$, which is the number of hidden units in the model. The input user preference is corrupted only while training the model, and not during cross-validation and test.

$$h(\tilde{\mathbf{x}}) = e\big(\mathbf{W} \cdot \tilde{\mathbf{x}} + \mathbf{b}\big) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{k \times n}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^k$ is a bias vector. The encoding function $e(\cdot)$ is set to the *ReLU* function [19] as it performs well due to its suitability for sparse data ($ReLU(x) = \max(0, x)$).

The hidden representation $h(\tilde{\mathbf{x}})$ is input into the attention mechanism layer, where the contextual attributes are applied. The objective of the attention mechanism is to summarize the input representation based on the provided context. The context may be associated with the user or item. Our model is flexible enough to accommodate as many contextual attributes as desired. However, we have specified two contextual attributes (p and q) in our model for ease of presentation. The attention mechanism applies a weighted user-context $\mathbf{c}_p$ and item-based context $\mathbf{c}_q$ for any given contextual parameters p and q to the output of the hidden layer with a nonlinear activation function $f(\cdot)$. Mathematically, this is denoted as:

$$t(\tilde{\mathbf{x}}) = f\big(\mathbf{W}_h \cdot h(\tilde{\mathbf{x}}) + \mathbf{W}_p \cdot \mathbf{c}_p + \mathbf{W}_q \cdot \mathbf{c}_q\big) \tag{2}$$

where $\mathbf{W}_h$ is a $\mathbb{R}^{k \times k}$ weight matrix, where k is the number of units in the attention layer, which is the same as the number of units in the hidden layer. $\mathbf{W}_p$ and $\mathbf{W}_q$ are weight matrices of dimensions $\mathbb{R}^{k \times |p|}$ and $\mathbb{R}^{k \times |q|}$ respectively, with $|p|$ and $|q|$ being the number of contextual parameters. $h(\tilde{\mathbf{x}})$ is the output of the hidden layer. We selected *tanh* as the attention mechanism activation function ($f(\cdot)$) as it gave us the best results ($tanh(x) = (\exp(x) - \exp(-x))\ /\ (\exp(x) + \exp(-x))$).

The output of the attention activation function, $t(\tilde{\mathbf{x}})$, is then fed into a *softmax* layer. Finally, the *softmax* output is combined with the hidden layer output via element-wise multiplication ($\otimes$) to generate the final output of the attention mechanism, which is denoted as $a(\tilde{\mathbf{x}})$.

$$a(\tilde{\mathbf{x}}) = softmax\big(t(\tilde{\mathbf{x}})\big) \otimes h(\tilde{\mathbf{x}}) \tag{3}$$

where $t(\tilde{\mathbf{x}})$ is the output of the attention activation function, and $h(\tilde{\tilde{\mathbf{x}}})$ is the hidden layer output. The *softmax* function is defined as: $softmax(x_1, x_2, ..., x_n) = \exp(x_i)\ /\ \sum_{j=1}^{n} \exp(x_j)$.

Essentially, the attention mechanism serves to apply a weighted arithmetic mean to the hidden layer representation, with the weight representing the relevance based on the provided context.

The internal latent representation of the input with the applied context is reconstructed back to the original form using a decoding function $d(\cdot)$.

$$\hat{\mathbf{x}} = d\big(\mathbf{W}' \cdot a(\tilde{\mathbf{x}}) + \mathbf{b}'\big) \tag{4}$$

where the dimension of $\mathbf{W}'$ and $\mathbf{b}'$ is the same as $\mathbf{W}$ and $\mathbf{b}$. We would like to point out that the reconstruction of the original input, or the reverse mapping, may be constrained by sharing parameters $\mathbf{W}' = \mathbf{W}^T$. However, we did not do so as we got better results by having different $\mathbf{W}'$ and $\mathbf{b}'$ parameters at the decoding step.

We selected the *sigmoid* function ($\sigma(x) = 1\ /\ 1 + \exp(-x)$) as the decoding function $d(\cdot)$ as it constraints an input to the $0 - 1$ output range. This gives us the probability associated with each item at the output, and we use this value for ranking in personalized recommendation. The *ADCA* model is generic and it can be applied to a rating prediction task by simply selecting any other nonlinear function as the decoding function $d(\cdot)$.

The parameters of the model are trained by minimizing the mean squared error between the original corrupted input vector $\tilde{\mathbf{x}}$ and the reconstructed vector $\hat{\mathbf{x}}$.

$$\min_{\mathbf{W}, \mathbf{W}', \mathbf{W}_h, \mathbf{W}_p, \mathbf{W}_q, \mathbf{b}, \mathbf{b}'} \frac{1}{m} \sum_{u \in U} \|\tilde{\mathbf{x}}_u - \hat{\mathbf{x}}_u\|^2 \tag{5}$$

The parameters are updated using the stochastic gradient descent variant ADAM optimizer [13]. We also used dropout [8] for regularization to prevent overfitting and improve generalization capacity. We set the dropout rate to be 0.2, which means that 20% of the hidden units are dropped at random at each training step to prevent co-adaptation.

3.2 Top-N Recommendation

The proposed *ACDA* model can be applied to both rating prediction and top-N recommendation by simply changing the decoding function $d(\cdot)$. We set the decoding function $d(\cdot)$ to the *sigmoid* function for top-N recommendation, and apply the generic *ACDA* model to the event recommendation and movie recommendation tasks.

The event recommendation task utilizes the RSVP [2] data from *Meetup*. Users indicate their preference to an event by providing an RSVP, which is used to recommend future events to the user. For the event recommendation task, we utilize the user *group* and event *venue* as the contextual attributes. Users typically organize themselves into groups in an Event Based Social Network (EBSN) such as *Meetup*, and each event is hosted at a physical venue. The user's preference on existing events in the training set is input into the model as a binary k-hot encoded vector with a true value for the positive event preferences and false for the negative or unknown event preferences. The input preference is corrupted as discussed earlier in section 3.1. In addition to corrupting the input, we also include a fixed number of negative samples by encoding them as positive in the input vector. The negative samples are selected randomly from the training set and negative sample inclusion is only performed during training, not during evaluation on the cross-validation and test sets. The output of the model is the personalized top-N event recommendation for the user.

For the event recommendation task, the contextual attributes of the model are set as: $\mathbf{c}_p = \mathbf{u}_g$ and $\mathbf{c}_q = \mathbf{i}_v$, where $\mathbf{u}_g \in \mathbb{R}^{|p|}$ denotes the groups that the user belongs to. The parameter $\mathbf{i}_v \in \mathbb{R}^{|q|}$ denotes the venues associated with the events. The parameters $|p|$ and $|q|$ are the number of groups and venues respectively.

We also apply the *ACDA* model to movie recommendation, which is also treated as a top-N recommendation task. The *Movielens* dataset contains the movie ratings on a scale of $1 - 5$, which we convert to a binary scale. The movie binary scale indicates a user's preference on existing movies, which is used to recommend other movies to the user. We select the movie *genre* as a contextual attribute for our model. The *genre* is associated with each movie, and certain movies have multiple genres associated with them. Similar to the event recommendation task, the user's preference is partially corrupted and input into the model as a binary k-hot encoded vector. We also used negative samples during training. The output of the model is the personalized top-N movie recommendation for the user.

Since we have only one item-related contextual attribute for the movie recommendation task, we update the model as: $\mathbf{c}_q = \mathbf{i}_r$

[2]RSVP is a French expression, which means "please respond"

Table 1: Data Statistics

Dataset	Observations	Sparsity	Positive	Negative	Users	Items
Meetup-NYC	73,816	0.9998	70,170	3,646	19,122	36,054
Meetup-SFO	48,972	0.9998	43,637	5,335	18,957	14,445
Meetup-DC	36,451	0.9998	33,541	2,901	10,384	12,359
Meetup-Chicago	22,915	0.9996	20,826	2,089	8,118	9,133
Movielens	100,004	0.9835	15,095	84,909	671	9,066

where $\mathbf{i}_r \in \mathbb{R}^{|q|}$ denotes the genres associated with the movies preferred by the user. The parameter $|q|$ is the number of genres.

4 EXPERIMENTS

4.1 Datasets

We evaluate the proposed *Attentive Contextual Denoising Autoencoder (ACDA)* model on real-world datasets from *Meetup* and *Movielens*. The Meetup dataset is for events from *New York*, *San Francisco*, *Washington DC* and *Chicago*. These cities were selected as they are the major metropolitan areas in the United States and they have a vibrant Meetup community. The event data was collected by using the Meetup API[3] between January 2016 and May 2016. We also analyzed our model against the publicly available *Movielens (100K)* dataset. The *Movielens* dataset consists of movie ratings provided by the user on the $1 - 5$ scale. We converted the numeric rating score to a binary rating for the purpose of top-N recommendation. A score of 5 is converted to a binary rating of 1, and anything less than a 5 is converted to 0. The statistics of the datasets used for the experiments are given in Table 1.

4.2 Experimental Setup

We split the datasets to use 60% as the training set, 20% as the cross-validation set, and 20% as the test set. The evaluation metrics include $P@5$, $P@10$, $R@5$, $R@10$, $NDCG@5$, $NDCG@10$, $MAP@5$ and $MAP@10$ [17]. These are common metrics for top-N recommendations. We consider baselines methods from each of the following categories for comparison against the proposed *ACDA* model: Neighborhood-based Methods (*UserKNN*, *ItemKNN*), Model-based Methods (*BiasedMF*, *BPR-MF*, *SVD++*), and Deep Learning Methods (*CDAE*, *U-AutoRec*). The results are presented in this section and we discuss our findings in detail.

We use *Librec*[4], a recommender library, to obtain results for the neighborhood and model-based methods. We use our own implementation of the deep learning baseline models. The parameter values for the existing methods are similar to the proposed method (to the extent possible).

- *User-KNN*: User k-nearest neighborhood collaborative filtering method that predicts the user preference based on the similarity with the k nearest users. We selected $k = 10$ as it gave the best results.
- *Item-KNN*: Item k-nearest neighborhood collaborative filtering method that predicts the user preference based on the similarity with the k nearest items. We set the value of $k = 10$ to be consistent with *User-KNN*.
- *BPR-MF*: Bayesian personalized ranking method that utilizes pairwise loss to provide top-N item recommendation using matrix factorization (MF). The latent factor count is set to $l = 50$ as it offered the best performance.
- *Biased-MF*: Basic matrix factorization that includes global mean, user bias and item bias. We set the latent factor count $l = 50$ to be consistent with the *BPR-MF* method.
- *SVD++*: State-of-the-art matrix factorization method that incorporates implicit feedback from the user into the baseline SVD model for better accuracy. We set the latent factor count $l = 50$ to be consistent with the *BPR-MF* method.
- *CDAE*: Collaborative filtering technique based on denoising autoencoders that incorporates the user latent factor as additional input [30].
- *U-AutoRec*: Collaborative filtering technique based on denoising autoencoders [23] that has two variants: *I-AutoRec*, which accepts the k-hot encoded item preference vector consisting of users as input, and *U-AutoRec* that accepts the k-hot encoded user preference vector of items. We compared against the *U-AutoRec* variant as it is similar to our proposed *ACDA* model in terms of the user preference on items being provided as input.

We evaluate the proposed models to incorporate the influence of the different contextual attributes for the event and movie recommendation tasks.

- *ACDA-V*: This is the variant of the proposed generic *ACDA* model that incorporates only the event *venue* as a contextual attribute for the event recommendation task.
- *ACDA-G*: A variant of the proposed generic *ACDA* model that just incorporates the user *group* as a contextual attribute for the event recommendation task.
- *ACDA-GV*: This model includes both the user *group* and event *venue* as contextual attributes of the *ACDA* model for the event recommendation task.
- *ACDA-R*: This model includes the movie *genre* as a contextual attribute of the *ACDA* model for the movie recommendation task.

We did not include the basic *ACDA* model (without the contextual attributes) into the comparison as that is basically the *U-AutoRec* model, which we have considered as a baseline method. The proposed *ACDA* models are trained on training set 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, the results of which are published for comparison with the baselines. The proposed models are developed and trained using Google's tensorflow library[5]. We conducted additional experiments to determine the optimal value for the hidden unit size and corruption ratio hyper-parameters. The results of the additional experiments are provided in Section 4.3.

We selected the $epoch = 200$ during training as we found the model to converge at this point. We experimented with different learning rates $(0.1, 0.01, 0.05, 0.001, 0.005)$ and found the learning rate $\alpha = 0.001$ to work best. To prevent the model from just training on positive samples, we paired the positive samples of a user with a configurable number of negative or unknown samples for the user.

[3]http://www.meetup.com/meetup_api/
[4]http://www.librec.net
[5]http://www.tensorflow.org

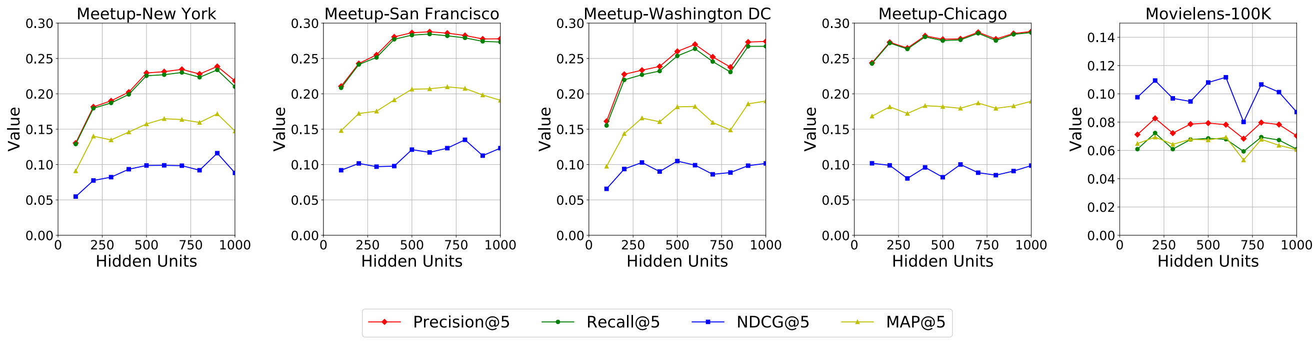

Figure 2: Hidden Unit Count Selection

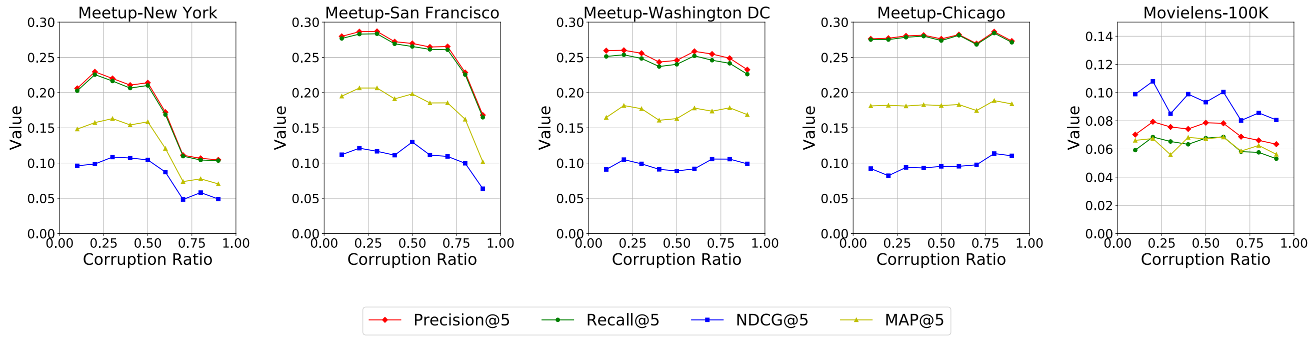

Figure 3: Corruption Ratio Selection

Table 2: Experimental Results - New York

Method	P@5	P@10	R@5	R@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.0069	0.0034	0.0220	0.0237	0.0223	0.0232	0.0186	0.0181
ItemKNN	0.0076	0.0038	0.0241	0.0254	0.0213	0.0222	0.0194	0.0191
Biased-MF	0.0003	0.0002	0.0001	0.0002	0.0002	0.0003	0.0008	0.0007
BPR-MF	0.0501	0.0342	0.1266	0.1524	0.0766	0.0825	0.1061	0.1089
SVD++	0.0005	0.0004	0.0003	0.0006	0.0004	0.0006	0.0001	0.0001
CDAE	0.1035	0.1477	0.1123	0.1657	0.0707	0.0791	0.0788	0.0994
U-AutoRec	0.0527	0.0804	0.0508	0.0790	0.0272	0.0305	0.0334	0.0517
ACDA-V	0.1086	0.1844	0.1066	0.1834	0.0523	0.0541	0.0739	0.1151
ACDA-G	0.1781	0.2320	0.1760	0.2309	0.0860	0.0871	0.1337	0.1738
ACDA-GV	**0.2295**	**0.2905**	**0.2255**	**0.2990**	**0.0987**	**0.0994**	**0.1574**	**0.2116**

4.3 The Effect of Hidden Units and Corruption Ratio

To investigate the effect of the number of hidden units on the performance, we experimented with different values of k, ranging from 100 to 1000 in increments of 100. The results are provided in Figure 2. As observed from the plots, we found that the performance of the model plateaus after $k = 500$, with higher values offering no significant gain in performance at a cost of increased training time. While there are certain metrics, such as the *NDCG@5*, that perform slightly better at higher values k, we set $k = 500$ as a default choice.

We also experimented with different values of the corruption ratio ranging from 0.1 to 0.9 in increments of 0.1. The results, depicted in Figure 3, indicate that the performance degrades with higher values of the corruption ratio. The only exception to this is the *Meetup-Chicago* dataset, which does not have a observable degradation in performance at higher values of the corruption ratio. Therefore, we default the value of the corruption ratio $\rho = 0.2$.

4.4 Baseline Comparisons

Tables 2, 3, 4, 5, 6 contains the results of the different methods, with the best results highlighted in boldface. A general observation is that, other than a few exceptions, the results on the *precision*, *recall*, *NDCG* and *MAP* metrics were consistent across all the datasets. The proposed model performed well on the *Meetup* and *Movielens* datasets, which demonstrates its effectiveness on top-N recommendation tasks.

Table 3: Experimental results - San Francisco

Method	P@5	P@10	R@5	R@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.0166	0.0127	0.0401	0.0615	0.0317	0.0390	0.0232	0.0255
ItemKNN	0.0155	0.0131	0.0394	0.0591	0.0303	0.0381	0.0229	0.0245
Biased-MF	0.0004	0.0004	0.0001	0.0003	0.0005	0.0004	0.0002	0.0001
BPR-MF	0.0552	0.0376	0.1486	0.1809	0.0860	0.0977	0.1217	0.1254
SVD++	0.0014	0.0009	0.0011	0.0017	0.0017	0.0016	0.0009	0.0007
CDAE	0.1109	0.1877	0.1098	0.1909	0.0519	0.0564	0.0804	0.1129
U-AutoRec	0.1045	0.1525	0.1020	0.1513	0.0634	0.0686	0.0730	0.1084
ACDA-V	0.1793	0.2623	0.1773	0.2616	0.0765	0.0789	0.1226	0.1770
ACDA-G	0.1879	0.3061	0.1649	0.3049	0.0602	0.0622	0.1004	0.1825
ACDA-GV	**0.2864**	**0.3708**	**0.2830**	**0.3692**	**0.1211**	**0.1266**	**0.2065**	**0.2743**

Table 4: Experimental results - Washington DC

Method	P@5	P@10	R@5	R@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.0013	0.0006	0.0039	0.0045	0.0023	0.0025	0.0016	0.0017
ItemKNN	0.0011	0.0005	0.0042	0.0042	0.0027	0.0027	0.0018	0.0019
Biased-MF	0.0003	0.0001	0.0003	0.0022	0.0002	0.0010	0.0007	0.0003
BPR-MF	0.0588	0.0466	0.1188	0.1509	0.0688	0.0753	0.1068	0.1098
SVD++	0.0007	0.0007	0.0001	0.0006	0.0012	0.0011	0.0007	0.0004
CDAE	0.0983	0.1860	0.0914	0.1812	0.0459	0.0515	0.0663	0.1089
U-AutoRec	0.0905	0.1136	0.0847	0.1083	0.0560	0.0606	0.0731	0.0885
ACDA-V	0.1737	0.2498	0.1676	0.2455	0.0836	0.0863	0.1177	0.1713
ACDA-G	0.1956	0.2833	0.1886	0.2792	0.0777	0.0794	0.1198	0.1886
ACDA-GV	**0.2600**	**0.3472**	**0.2536**	**0.3430**	**0.1049**	**0.1092**	**0.1816**	**0.2476**

Table 5: Experimental results - Chicago

Method	P@5	P@10	R@5	R@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.0065	0.0041	0.0204	0.0213	0.0160	0.0170	0.0122	0.0125
ItemKNN	0.0062	0.0037	0.0193	0.0202	0.0157	0.0166	0.0118	0.0120
Biased-MF	0.0004	0.0003	0.0002	0.0006	0.0002	0.0005	0.0008	0.0001
BPR-MF	0.0498	0.0311	0.1640	0.1925	0.0952	0.1059	0.1277	0.1322
SVD++	0.0020	0.0014	0.0005	0.0012	0.0020	0.0020	0.0010	0.0008
CDAE	0.1271	0.2097	0.1260	0.2095	0.0428	0.0438	0.0724	0.1297
U-AutoRec	0.0879	0.1209	0.0878	0.1209	0.0476	0.0479	0.0621	0.0855
ACDA-V	0.2354	0.3356	0.2339	0.3353	0.0805	0.0796	0.1493	0.2261
ACDA-G	**0.2866**	**0.3994**	**0.2849**	**0.3988**	**0.1375**	**0.1395**	**0.2052**	**0.2835**
ACDA-GV	0.2771	0.3856	0.2752	0.3849	0.0821	0.0847	0.1819	0.2655

Table 6: Experimental results - Movielens 100K

Method	P@5	P@10	R@5	R@10	NDCG@5	NDCG@10	MAP@5	MAP@10
UserKNN	0.0200	0.0185	0.0052	0.0082	0.0200	0.0204	0.0109	0.0089
ItemKNN	0.0195	0.0164	0.0055	0.0079	0.0205	0.0194	0.0097	0.0081
Biased-MF	0.0008	0.0008	0.0002	0.0004	0.0007	0.0008	0.0002	0.0002
BPR-MF	0.0761	0.0603	**0.1022**	**0.1477**	0.0907	0.0997	0.0611	0.0691
SVD++	0.0017	0.0011	0.0001	0.0001	0.0016	0.0013	0.0010	0.0006
CDAE	0.0595	0.0887	0.0533	0.0869	0.0664	0.0782	0.0428	0.0613
U-AutoRec	0.0691	0.0986	0.0609	0.0962	0.0932	0.1017	0.0631	0.0718
ACDA-R	**0.0827**	**0.1106**	0.0723	0.1070	**0.1094**	**0.1189**	**0.0694**	**0.0835**

First, we discuss the performance of the baseline methods. We considered three different categories of the baseline methods: neighborhood-based, model-based and deep learning based methods. Among these categories, we found the deep learning based baseline methods to perform better than the others. In general, across the baseline methods, the *CDAE* deep learning based method performs better on the precision and recall metrics. The *BPR-MF* is better on the *NDCG* metric. The *CDAE* method is based on the denoising autoencoder, and the results signify its importance to recommender systems. The good performance of the *BPR-MF* method may be attributed to the use of the pairwise loss function. We also observe good results for the *BPR-MF* method against the *Movielens* dataset. However, we found *U-AutoRec* to perform better than *CDAE* against the *Movielens* dataset. This suggests that the user latent factor included in the *CDAE* model does not help to improve the performance against the *Movielens* dataset, but it does so against the *Meetup* dataset. When considering the neighborhood methods, we found both (*UserKNN* and *ItemKNN*) to be similar in performance.

Comparing the baseline methods to the proposed models for the event recommendation task, we observed that all three variants of the proposed *ACDA* model perform better than the baselines. While a variant of the *ACDA* model with some of the contextual attributes may perform better, in general the model with more contextual attributes offers the better performance. As we can see for the *Meetup* datasets, the *ACDA-GV* model offers a better performance in three of the four cities. We also observe that the significance of the contextual attributes is not equal. The influence of the user *group* contextual attribute is higher than the event *venue* attribute, and the model *ACDA-G* performs better than the *ACDA-V* model. This implies that additional contextual parameters may improve the performance further in some cases, however, this may not be always true. With regard to the movie recommendation task, we utilize the movie *genre* as a contextual attribute. The model *ACDA-R* performs better on all metrics except the *recall*. The *BPR-MF* method is better on the *recall* metric, perhaps due to the fact that it uses a pairwise loss function. We intend to evaluate the performance of the proposed models using pairwise loss in the future. The results, which are consistent across all datasets, reinforce our assertion that the proposed *ACDA* model performs well on recommendation tasks.

5 CONCLUSION AND FUTURE WORK

We propose a deep learning architecture for contextual recommendation based on denoising autoencoder augmented with a context-driven attention mechanism. We also perform comprehensive experiments demonstrating that the proposed *ACDA* model outperforms the state-of-the-art baselines on event recommendation and movie recommendation tasks.

We understand that this preliminary study can be extended in many directions and we plan to do so in future work. First of all, we will investigate other types of loss functions including pairwise and listwise losses which demonstrate good performance for ranking tasks especially with implicit user feedback [16]. Secondly, we will explore deeper architectures by adding more layers and experimenting with different activation functions. Last, we will conduct experiments in other domains of contextual recommendation.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*.

[3] Minmin Chen, Kilian Weinberger, Fei Sha, and Yoshua Bengio. 2014. Marginalized denoising auto-encoders for nonlinear representations. In *ICML*.

[4] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*.

[5] Rong Du, Zhiwen Yu, Tao Mei, Zhitao Wang, Zhu Wang, and Bin Guo. 2014. Predicting activity attendance in event-based social networks: Content, context and social influence. In *UbiComp*.

[6] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR*.

[7] Yuyun Gong and Qi Zhang. 2016. Hashtag Recommendation Using Attention-Based Convolutional Neural Network. In *IJCAI*.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*.

[10] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

[11] Yogesh Jhamb and Yi Fang. 2017. A dual-perspective latent factor model for group-aware social event recommendation. *Information Processing & Management* 53, 3 (2017).

[12] Houda Khrouf and Raphaël Troncy. 2013. Hybrid event recommendation using linked data and user diversity. In *RecSys*.

[13] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

[14] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42 (2009).

[15] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*.

[16] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* (2009).

[17] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge University Press.

[18] Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic matrix factorization. In *NIPS*.

[19] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

[20] Maria S Pera and Yiu-Kai Ng. 2013. A group recommender for movies based on content similarity and popularity. *IPM* (2013).

[21] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[22] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*.

[23] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*.

[24] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In *RecSys*.

[25] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Representation Learning of Users and Items for Review Rating Prediction Using Attention-based Convolutional Neural Network. In *MLRec*.

[26] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*.

[27] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

[28] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*.

[29] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*.

[30] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*.

[31] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *IJCAI*.

[32] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

[33] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *arXiv:1707.07435* (2017).

[34] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *SIGIR*.

[35] Wei Zhang, Jianyong Wang, and Wei Feng. 2013. Combining latent factor model with location features for event-based group recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 910–918.