BEYOND GREEDY RANKING: SLATE OPTIMIZATION VIA LIST-CVAE

Anonymous authors
Paper under double-blind review

ABSTRACT

The conventional approach to solving the recommendation problem greedily ranks individual document candidates by prediction scores. However, this method fails to optimize the slate as a whole, and hence, often struggles to capture biases caused by the page layout and document interdependencies. The slate recommendation problem aims to directly find the optimally ordered subset of documents (i.e. slates) that best serve users’ interests. Solving this problem is hard due to the combinatorial explosion of document candidates and their display positions on the page. In this paper, we introduce List Conditional Variational Auto-Encoders (List-CVAE), which learn the joint distribution of documents on the slate conditioned on user responses, and directly generate full slates. Experiments on simulated and real-world data show that List-CVAE outperforms greedy ranking methods consistently on various scales of documents corpora.

1 INTRODUCTION

Recommendation systems modeling is an important machine learning area in the IT industry, powering online advertising, social networks and various content recommendation services (Schafer et al., 2001; Lu et al. 2015). In the context of document recommendation, its aim is to generate and display an ordered list of “documents” (called a “slate” in Swaminathan et al., 2017; Sunehag et al., 2015) to users, based on both user preferences and documents content. For large scale recommender systems, a common scalable approach at inference time is to first select a small subset of candidate documents $S$ out of the entire document pool $D$. This step is called “candidate generation”. Then a function approximator such as a neural network (e.g., a Multi-Layer Perceptron or MLP) called the “ranking model” is used to predict probabilities of user engagements for each document in the small subset $S$ and greedily generates a slate by sorting the top documents from $S$ based on estimated prediction scores (Covington et al., 2016). This two-step process is widely popular to solve large scale recommender problems due to its scalability and fast inference at serving time. The candidate generation step can decrease the number of candidates from millions to hundreds or less, effectively dealing with scalability when faced with a large corpus of documents $D$. Since $|S|$ is much smaller than $|D|$, the ranking model can be reasonably complicated without increasing latency.

However, there are two main problems with this approach. First the candidate generation and the ranking models are not trained jointly, which can lead to having candidates in $S$ that are not the highest scoring documents of the ranking model. Second and most importantly, the greedy ranking of documents on the slate suffers from numerous biases that come with the visual presentation of the slate and context in which documents are presented. For example, there exists positional biases caused by users paying more attention to prominent slate positions (Joachims et al., 2005), and contextual biases, due to interactions between documents presented together in the same slate, such as competition and complementarity, relative attractiveness, etc. (Yue et al., 2010).

In this paper, we study the slate recommendation problem beyond the greedy ranking paradigm. We consider a slate “optimal” when it maximizes some type of user engagement feedback, a typical desired scenario in recommendation systems. For example, given a database of song tracks, the optimal slate can be an ordered list (in time or space) of $k$ songs such that the user ideally likes every song in that list (in the order they are presented). Another example considers news articles, the optimal slate has $k$ ordered articles such that every article is read by the user. In general, optimality can be defined as a desired user response vector on the slate and the proposed model should be...
agnostic to these problem-specific definitions. Solving the slate recommendation problem directly differs from ranking in that first, it does not assume that more relevant documents should necessarily be put in earlier positions in the slate. Second, the entire slate is used as a training example instead of single documents, preserving numerous biases encoded into the slate that might influence user responses. Finally, our model directly generates slates, again taking into account all the influential biases the model learned through training.

In this paper, we apply Conditional Variational Auto-Encoders (CVAEs) (Kingma et al., 2014; Kingma and Welling, 2013; Jimenez Rezende et al., 2014) to model the distributions of all documents in the same slate conditioned on the user response. All documents in a slate along with their positional, contextual biases are jointly encoded into the latent space, which is then sampled and combined with desired conditioning for direct slate generation, i.e. sampling from the learned conditional joint distribution. Therefore, the model first learns which slates give which type of responses and then directly generates similar slates given a desired response vector as the conditioning at inference time. We call our proposed model List-CVAE. The key contributions of our work are:

1. To the best of our knowledge, this is the first model that provides a conditional generative modeling framework for slate recommendation by direct generation. It does not necessarily require a candidate generator at inference time and is flexible enough to work with any visual presentation of the slate as long as the ordering of display positions is fixed throughout training and inference times.

2. To deal with the problem at scale, we introduce an architecture that uses pretrained document embeddings combined with a negatively downsampled $k$-head softmax layer within the List-CVAE model, where $k$ is the slate size.

The structure of this paper is the following. First we introduce related work using various CVAE-type models as well as other approaches to solve the slate generation problem. Next we introduce our List-CVAE modeling approach. The last part of the paper is devoted to experiments on both simulated and the real-world datasets.

## RELATED WORK

![Graph](image)

Figure 1: Comparison of related variants of VAE models. Note that user variables are not included in the graphs for clarity. (a) VAE; (b) CVAE-CF with auxiliary variables; (c) Joint Variational Auto-Encoder-Collaborative Filtering (JV AE-CF); (d) JMV AE; and, (e) List-CVAE (ours) with the whole slate as input.

Traditional matrix factorization techniques have been applied to recommender systems with success in modeling competitions such as the Netflix Prize (Koren et al., 2009). Later research emerged on using autoencoders to improve on the results of matrix factorization (Wu et al., 2016; Wang et al., 2015) (CDAE, CDL). More recently several works use Boltzmann Machines (Abdollahi and Nasraoui, 2016) and variants of VAE models in the Collaborative Filtering (CF) paradigm to model recommender systems (Li and She, 2017; Lee et al., 2017; Liang et al., 2018) (Collaborative VAE, JMV AE, CVAE-CF, JV AE-CF). See Figure 1 for model comparisons. In this paper, unless specified otherwise, the user features and any context are routinely considered part of the conditioning variables (in Section 4.1 Personalization Test, we test List-CVAE generating personalized slates for different users). These models have primarily focused on the greedy approach of modeling independently each document or pairs of documents in the slate and applying greedy ordering at inference time.

Our model is also using a VAE type structure and in particular, is closely related to the Joint Multimodel Variational Auto-Encoder (JMV AE) architecture (Figure 1d). However, we use whole
slates as input instead of single documents, and directly generate slates instead of using greedy ranking by prediction scores.

Other relevant work from the Information Retrieval (IR) literature are listwise ranking methods, e.g. (Cao et al., 2007; Xia et al., 2008; Shi et al., 2010; Huang et al., 2015). These methods use a listwise loss function that takes context and position into account. They also eventually assign a prediction score for each document and greedily rank them at inference time.

In the Reinforcement Learning (RL) literature, Sunehag et al. (2015) view the whole slates as actions and use a deterministic policy gradient update to learn a policy that generates these actions, given concatenated document features as input.

Finally, the framework proposed by (Wang et al., 2016) predicts user engagement for document and position pairs. It optimizes whole page layouts but may suffer from poor scalability due to combinatorial explosion.

3 Theory

3.1 Problem Setup

We formally define the slate recommendation problem as follows. Let $D$ denote a corpus of documents and let $k$ be the slate size. Then let $r = (r_1, \ldots, r_k)$ be the engagement response vector from users where $r_i \in \mathcal{R}$ is the user response for document $d_i$. For example, if the problem is to maximize the number of clicks on a slate, then $r_i \in \{0, 1\}$ denotes whether the document $d_i$ is clicked, and an optimal slate $s = (d_1, d_2, \ldots, d_k)$ where $d_i \in D$ is such that $s$ maximizes $\sum_{i=1}^{k} r_i$.

3.2 Variational Auto-Encoders

Variational Auto-Encoders (VAEs) are latent-variable models that define a joint density $P_\theta(x, z)$ between observed variables $x$ and latent variables $z$ parametrized by a vector $\theta$. Training such models requires marginalizing the latent variables in order to maximize the data likelihood $P_\theta(x) = \int P_\theta(x, z)dz$. Since we cannot solve this marginalization explicitly, we resort to a variational approximation. For this, a variational posterior density $Q_\phi(z|x)$ parametrized by a vector $\phi$ is introduced and we optimize the variational Evidence Lower-Bound (ELBO) on the data log-likelihood:

$$
\log P_\theta(x) = \text{KL}[Q_\phi(z|x)\|P_\theta(z|x)] + \mathbb{E}_{Q_\phi(z|x)}[-\log Q_\phi(z|x) + \log P_\theta(x, z)],
$$

$$
\geq -\text{KL}[Q_\phi(z|x)\|P_\theta(z)] + \mathbb{E}_{Q_\phi(z|x)}[\log P_\theta(x|z)],
$$

where KL is the Kullback–Leibler divergence and where $P_\theta(z)$ is a prior distribution over latent variables. In a Conditional VAE (CVAE) we extend the distributions $P_\theta(x, z)$ and $Q_\phi(z|x)$ to also depend on an external condition $c$. The corresponding distributions are indicated by $P_\theta(x, z|c)$ and $Q_\phi(z|x,c)$. Taking the conditioning $c$ into account, we can write the variational loss to minimize as

$$
\mathcal{L}_{\text{CVAE}} = \text{KL}[Q_\phi(z|x,c)\|P_\theta(z|c)] - \mathbb{E}_{Q_\phi(z|x,c)}[\log P_\theta(x,z|c)].
$$

3.3 Our Model

We assume that the slates $s = (d_1, d_2, \ldots, d_k)$ and the user response vectors $r$ are jointly drawn from a distribution $P_{D^k \times R^k}$. In this paper, we use a CVAE to model the joint distribution of all documents in the slate conditioned on the user responses $r$, i.e. $P(d_1, d_2, \ldots, d_k|r^*)$. At inference time, the List-CVAE model attempts to generate an optimal slate by conditioning on the ideal user response $r^*$.

As we explained in Section 1, “optimality” of a slate depends on the task. With that in mind, we define the mapping $\Phi: \mathcal{R}^k \mapsto \mathcal{C}$. It transforms a user response $r$ into a vector in the conditioning.
As usual with CV AEs, the decoder models a distribution \( z \). To shed light onto what is encoded in the latent space, we simplify the prior distribution of \( x \) variable with a learned prior distribution \( \mathcal{N}(\mu_0, \sigma_0) \) after training, if we sample \( z \) latent space while high response slates cluster towards a larger, centered area (Figure 3). Therefore training evolves, generated output slates with low total responses are pushed towards the edge of the latent space.

As usual with CVAEs, the decoder models a distribution \( P_{\theta}(s|z, c) \) that, conditioned on \( z \), is easy to represent. In our case, \( P_{\theta}(s|z, c) \) models an independent probability for each document on the slate \( s \). The concatenation of \( s \) and \( c \) makes the input vector to the encoder. \( z \in \mathbb{R}^m \) is the latent variable with a learned prior distribution \( \mathcal{N}(\mu_0, \sigma_0) \). The raw output from the decoder are \( k \) vectors \( x_1, x_2, \ldots, x_k \), each of which is mapped to a real document through taking the dot product with the matrix \( \Phi \) containing all document embeddings. Thus produced \( k \) vectors of logits are then passed to the negatively downsampled \( k \)-head softmax operation. \( c^* \) is the ideal condition whose concatenation with sampled \( z \) is the input to the decoder at inference time.

To shed light onto what is encoded in the latent space, we simplify the prior distribution of \( z \) to be a fixed Gaussian distribution \( \mathcal{N}(0, I) \) in \( \mathbb{R}^2 \). We train List-CVAE and plot the predictive prior \( z \). As training evolves, generated output slates with low total responses are pushed towards the edge of the latent space while high response slates cluster towards a larger, centered area (Figure 3). Therefore after training, if we sample \( z \) from its prior distribution \( \mathcal{N}(0, I) \) and generate the corresponding output slates, they more likely to have high total responses.
We train this model as a CV AE by minimizing the sum of the reconstruction loss and the KL-divergence term:

\[
L = \beta \text{KL}[Q_\phi(z | s, c) || P_\theta(z | c)] - \mathbb{E}_{Q_\phi(z | s, c)}[\log P_\theta(s | z, c)],
\]

where \(\beta\) is a function of the current training step \cite{higgins2017beta}. During inference, output slates are generated by first sampling \(z\) from the conditionally learned prior distribution \(\mathcal{N}(\mu^*, \sigma^*)\), then concatenating with the ideal condition \(c^* = \Phi(r^*)\), thus passed into the decoder generating \((x_1, \ldots, x_k)\) from the learned \(P_\theta(s | z, c^*)\), and finally picking the argmax over the dot-products with the full embedding matrix independently for each \(i = 1, \ldots, k\).

4 Experiments

4.1 Simulation Data

Setup: The simulator generates a random matrix \(W \sim \mathcal{N}(\mu, \sigma)^{k \times n \times k \times n}\) where each element \(W_{i,d_i,j,d_j}\) represents the interaction between document \(d_i\) at position \(i\) and document \(d_j\) at position \(j\), and \(n = |D|\). It simulates biases caused by layouts of documents on the slate (below, we set \(\mu = 1\) and \(\sigma = 0.5\)). Every document \(d_i \in D\) has a probability of engagement \(A_i \sim \mathcal{U}(0, 1)\) representing its innate attractiveness. User responses are computed by multiplying \(A_i\) with interaction multipliers \(W(i, d_i, j, d_j)\) for each document presented before \(d_i\) on the slate. Thus the user response

\[
r_i \sim \mathcal{B}\left(\sum_{j=1}^i W_{i,d_i,j,d_j}\right)\]

for \(i = 1, \ldots, k\), where \(\mathcal{B}\) represents the Bernoulli distribution.

During training, all models see uniformly randomly generated slates \(s \sim \mathcal{U}\{1, n\}^k\) and their generated responses \(r\). During inference time, we generate slates \(s\) by conditioning on \(c^* = (1, \ldots, 1)\). We do not require document de-duplication since repetition may be desired in certain applications (e.g. in an online advertisement session). Moreover List-CVAE should learn to produce the optimal slates whether those slates contain duplication or not from learning the training data distribution.
**Evaluation:** For evaluation, we cannot use offline ranking evaluation metrics such as Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2000), Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto 1999) or Inverse Propensity Score (IPS) (Little and Rubin 2002), etc. These metrics either requires prediction scores for individual documents or assumes that more relevant documents should appear in earlier ranking positions, unfairly favoring greedy ranking methods.

Instead, we evaluate the expected number of clicks over the distribution of generated slates and over the distribution of clicks on each document:

$$
\mathbb{E}[\sum_{i=1}^{k} r_i] = \sum_{s \in \{1, \ldots, n\}^k} \mathbb{E}[\sum_{i=1}^{k} r_i | s] P(s) = \sum_{s \in \mathcal{D}^k} \sum_{r \in \mathcal{R}^k} \sum_{i=1}^{k} r_i P(r | s) P(s). \tag{6}
$$

In practice, we distill the simulated environment of Eq. 5 using cross-entropy loss onto a neural network model that officiates as our new simulation environment. The model consists of an embedding layer, which encodes documents into 8-dimensional embeddings. It then concatenates the embeddings of all the documents that form a slate and follows this concatenation with two hidden layers and a final softmax layer that predicts the slate response amongst the $2^k$ possible responses. Thus we call it the “response model”. We use the response model to predict user responses on 100,000 sampled output slates for evaluation purposes. This allows us to accurately evaluate our output slates by List-CVAE and all other baseline models.

**Baselines:** Our experiments compare List-CVAE with several greedy ranking baselines that are popularly deployed in industry productions, namely Greedy MLP, Position MLP, Pairwise MLP, Greedy Long Short-Term Memory (LSTM) models as well as the randomly generated slates as a sanity check. **List-CVAE** generates slates $s = \arg\max_{s \in \{1, \ldots, n\}^k} P_b(s | z, c^*)$. The encoder and decoder of List-CVAE, as well as all the greedy MLP-type models consist of two fully-connected neural network layers of the same size. **Greedy MLP** trains on $(d_i, r_i)$ pairs and outputs the greedy slate consisting of the top $k$ highest $P(r | d)$ scoring documents. **Position MLP** uses position in the slate as a feature during training time and sets it simply to 0 for fast performance in inference time. **Pairwise MLP** is an MLP model with a pairwise ranking loss $\mathcal{L} = \alpha \mathcal{L}_x + (1 - \alpha)\mathcal{L}(\hat{P}(x_1) - \hat{P}(x_2) + \eta)$ where $\mathcal{L}_x$ is the cross entropy loss and $(x_1, x_2)$’s are pairs of documents randomly selected with different user responses from the same slate. We sweep on hyperparameters $\alpha$ and $\eta$ in addition to the shared MLP model structure sweep. **Greedy LSTM** is an LSTM model with fully-connect layers before and after the recurrent middle layers. We tune the hyperparams on the number of layers and their respective widths. We use sequences of documents that form slates as inputs at training time, and uses single examples as inputs with sequence length 1 at inference time, which is similar to scoring documents as if they are in the first position of a slate of size 1. Then we greedily rank the documents based on their prediction scores.

**Small-scale experiment** ($n = 100, 1000, k = 10$):

We use the trained document embeddings from the response model for List-CVAE and all the baseline models. For List-CVAE, we also use trained priors $P_b(z | c) = \mathcal{N}(\mu^*, \sigma^*)$ where $\mu^*, \sigma^* = f_{\text{prior}}(c^*)$ and $f_{\text{prior}}$ is modeled by a small MLP (16, 32). Additionally, since we found little difference between different hyperparameters, we fixed the width of all hidden layers to 128, the learning rate to $10^{-3}$ and the number of latent dimensions to 16. For all other baseline models, we sweep on the learning rates, model structures and any model specific hyperparameters such as $\alpha, \eta$ for Position MLP and the forget bias for the LSTM model.

Figure 4a, 4b shows the performance comparison when the number of documents $n = 100, 1000$ and slate size to $k = 10$. While List-CVAE is not quite capable of reaching a perfect performance of 10 clicks (which is probably even above the optimal upper bound), it easily outperforms all other greedy ranking baselines after only a few training steps.

**Personalization test** ($|U| = 50, n = 100, k = 10$):

We add user features into the conditioning $c$, by adding a set $U$ of 50 different users to the simulation engine and permuting the innate attractiveness of documents and their interactions matrix $W$ by a
Figure 4: Small-scale experiments. The shaded area represent the 95% confidence interval over 20 independent runs. We compare List-CVAE against Greedy MLP, Position MLP, Pairwise MLP, Greedy LSTM and Random baselines on medium-scale synthetic data.

We sweep over hidden layers of 512 or 1024 units in List-CVAE, and all baseline MLP structures.

Figure 4c show that slates generated by List-CVAE have on average higher clicks than those produced by the greedy baseline models although the convergence took much longer to reach.

4.2 REAL-WORLD DATA

Due to a lack of publicly available large scale slate datasets, we use the data provided by the RecSys 2015 YOOCHOOSE Challenge (Ben-Shimon et al., 2015). This dataset consists of 9.2M user purchase sessions around 53K products. Each user session contains an ordered list of products on which the user clicked, and whether they decided to buy them. The List-CVAE model can be used on slates with temporal ordering. Thus we form slates of size 5 by taking consecutive clicked products. We then build user responses from whether the user bought them. We remove slates with no positive responses such that after removal they only account for 50% of the total number of slates. After filtering out products that are rarely bought, we get 375K slates of size 5 and a corpus of 10,000
candidate documents. Figure 5a shows the user response distribution of the training data. Notice that in the response vector, 0 denotes a click and 1 denotes a buying action. For example, (1,0,0,0,1) means the user clicked on all five products but bought only the first and the last products.

Medium-scale experiment \((n = 10,000, k = 5)\):

Similarly to the previous section, we train a two-layer response model that officiates as a new semi-synthetic simulation environment. We use the same hyperparameters used previously. Figure 5b shows that List-CVAE outperforms all greedy baseline models within 500 training steps, which corresponds to having seen less than \(10^{-11}\%\) of all possible slates.

Large-scale experiment \((n = 1 million, 2 millions, k = 5)\):

We synthesize 1,990k documents by adding independent Gaussian noise \(N(0, 0.01)^k\) to the original 10k documents and label the synthetic documents by predicted responses from the response model. The new pool of candidate documents consists of 10k original documents and 1,990k synthetic ones, totaling 2 million documents. To match each of the \(k\) decoder outputs \((x_1, x_2, \ldots, x_k)\) with real documents, we uniformly randomly downsample the negative document examples keeping in total only 1000 logits (the dot product outputs in the decoder). At inference time, we pick the argmax for each of \(k\) dot product outputs with the full embedding matrix without sampling. This technique speeds up the total training and inference time for 2 million documents to merely 4 minutes on 1 GPU.
for both the response model with 40k training steps and List-CVAE with 5k training steps. To order to
stress test our model, we used embedding dimension 16 and masked out 50% of embedding features
to mimic the real world scenario where we observe only a subset of all features. We ran 2 experiments
with 1 million and 2 millions documents respectively. From the results shown in Figure 5c and 5d,
List-CVAE steadily outperforms all other greedy baselines again. The greatly increased number
of training examples helped List-CVAE really learn all the interactions between documents and their
respective positional biases. The resulting slates were able to receive close to 5 buyings on average
due to the limited complexity provided by the response model.

**Generalization test:** In practice, we may not have any close-to-optimal slates in the training data.
Hence it is crucial that List-CVAE is able to generalize to unseen optimal conditions. To test its
generalization capacity, we use the medium-scale experiment setup on RecSys 2015 dataset and
eliminate from the training data all slates where the total user response exceed some ratio $h$ of the
slate size $k = 5$, i.e. $\sum_{i=1}^{k} r_i \geq hk$ for $h = 40\%, 60\%, 80\%, 100\%$. Figure 6 shows test results on
increasingly difficult training sets from which to infer on the optimal slates. Without seeing any
optimal slates (Figure 6a) or slates with 4 or 5 clicks (Figure 6b), List-CVAE can still produce close
to optimal slates. Even training on slates with only 1 or 2 total clicks ($h = 60\%$), List-CVAE still
surpasses the performance of all greedy baselines within 1000 steps (Figure 6c). Thus demonstrating
the strong generalization power of the model.
5 DISCUSSION

The List-CVAE model moves away from the conventional greedy ranking paradigm and provides the first conditional generative modeling framework that approaches slate recommendation problem using direct slate generation. By modeling the conditional probability distribution of documents in a slate directly, this approach not only picks up automatically the positional and contextual biases between documents but also gracefully avoids the problem of combinatorial explosion of possible slates when the candidate set is large. The framework is flexible and can incorporate different types of conditional generative models. In this paper we showed its superior performance over greedy baseline models with a conditional VAE model.

In addition, the List-CVAE model has good scalability. We designed an architecture that uses pretrained document embeddings combined with a negatively downsampled $k$-head softmax layer that greatly speeds up the training and scales easily to millions of documents.

REFERENCES


Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. 2015.


David Ben-Shimon, Michael Friedman, Alexander Tsikinovsky, and Johannes Hörle. Recsys challenge 2015 and the yoochoose dataset, 2015.