Sequential and Diverse Recommendation with Long Tail

Yejin Kim[1*], **Kwangseob Kim**[2], **Chanyoung Park**[3] and **Hwanjo Yu**[3]
[1]University of Texas Health Science Center at Houston
[2]Kakao Corp.
[3]Pohang University of Science and Technology
yejin.kim@uth.tmc.edu, lucas.kim@kakaocorp.com, {hwanjoyu, pcy1302}@postech.ac.kr

Abstract

Sequential recommendation is a task that learns a temporal dynamic of a user behaviour in sequential data and predicts items that a user would like afterward. However, diversity has been rarely emphasized in the context of sequential recommendation. Sequential and diverse recommendation must learn temporal preference on diverse items as well as on general items. Thus, we propose a sequential and diverse recommendation model that predicts a ranked list containing general items and also diverse items without compromising significant accuracy. To learn temporal preference on diverse items as well as on general items, we cluster and relocate consumed long tail items to make a pseudo ground truth for diverse items and learn the preference on long tail using recurrent neural network, which enables us to directly learn a ranking function. Extensive online and offline experiments deployed on a commercial platform demonstrate that our models significantly increase diversity while preserving accuracy compared to the state-of-the-art sequential recommendation model, and consequently our models improve user satisfaction.

1 Introduction

Users' feedback (e.g. click, view) in e-commerce naturally arrives one by one in *sequential* manner. *Sequential recommendation* is a task that learns a temporal dynamic of a user behaviour in the sequential data and predicts items that a user would like afterward. Given a user's historical data as a sequence, an objective of the sequential recommender systems is to recommend the next items that the user would be interested in.

Surprisingly, *diversity* [1] has been rarely emphasized in the context of sequential recommendation though it has been treated importantly in ordinary collaborative filtering-based recommender systems [Vargas and Castells, 2011; Oh *et al.*, 2011; Adomavicius and Kwon, 2012; Adomavicius and Kwon, 2011; Ziegler *et al.*, 2005; Christoffel *et al.*, 2015; Yin *et al.*, 2012; Antikacioglu and Ravi, 2017; Cheng *et al.*, 2017]. To recommend diverse items in the context of sequential recommendation, we must learn *temporal preference on diverse* items as well as on general items. Here, we define *general* item as a popular item that is frequently exposed to customers and *diverse* item as an unpopular and niche item that is rarely exposed to customers. Depending on what a user consumed previously, the preferred next diverse items can change. For example, Alice recently enjoyed popular twisted psychological thrillers ($Black\ Swan \rightarrow Memento \rightarrow Inception$) (Fig. 1). After watching *Memento* and *Inception*, Alice found the director *Christopher Nolan* attracts her attention, and thus Alice watched Nolan's other movie, *Dark Knight*. She also often watched diverse movies along with the general ones. After watching *Inception* Alice watched *Following* and *Quay*, which are *Nolan*'s less popular movies. Given another user, Bob, who also watched $Black\ Swan \rightarrow Memento \rightarrow Inception$, our goal is to recommend Bob a list of next movies that contains not only general ones but also diverse ones. In this case, the recommendation list should include *Dark Knight* and also some relevant diverse movies that have similar flavour with *Following* or *Quay* (rather than *Red Road* or *The Pledge*). To do so, sequential recommendation model should learn temporal preference on general items and also diverse items.

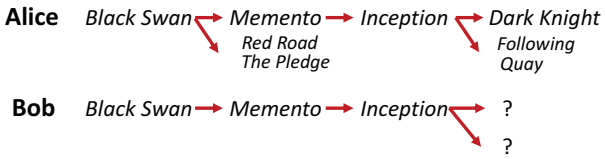

Figure 1: Example on movie sequences. After Bob watched *Inception*, we'd like to recommend both general movies (*Dark Knight*) and relevant diverse movies such as *Following* and *Quay* rather than *Red Road* and *The Pledge*

Thus, we propose a sequential and diverse recommendation model (S-DIV), that predicts a ranked list containing general items and also diverse items. One challenge here is that the diverse tail items are "cold" (i.e., too few occurrences) and consequently they give too much weights on implausible events and hurt the recommendation accuracy. Previous work removed these tail items during pre-processing step.

[*]This work is done during an internship at Kakao Corp.

[1]There are two types of diversity: aggregate or individual diversity. The aggregate diversity is for all recommended items across all users, which represents overall product variety and sales concentration [Adomavicius and Kwon, 2012; Adomavicius and Kwon, 2011; Anderson, 2004]. Whereas, the individual diversity is for recommended items to each individual user regardless of other users. Here, we focus on the aggregate diversity.

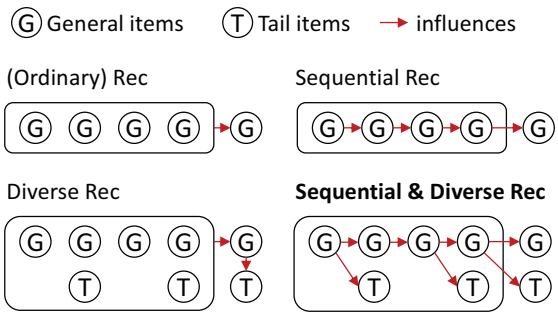

Figure 2: Comparing sequential and diverse recommendation model (S-DIV) with other recommendation models. Left: static models without considering temporal information. Right: sequential models that predict next items. Up: general models without diverse tail items. Down: diverse models that predict general items and tail items.

We tackle this problem by clustering the tail items and mapping them into content-based vector space. Besides, considering diversity in recommendation is technically challenging because accuracy and diversity are conflicting measures and thus simply increasing one will end up decreasing the other. Increasing diversity significantly while preserving (or negligibly losing) the state-of-the-art accuracy requires a careful design of optimization. To address this problem, we set a pseudo ground-truth ranking order as next general item (for accuracy) followed by relevant unpopular items (for diversity), and then we optimize a ranking function to preserve the pseudo ground-truth ranking order using recurrent neural network (RNN) and listwise learning-to-rank. Extensive online and offline experiments deployed on a commercial blog platform demonstrate that S-DIV significantly increases diversity while preserving the accuracy compared to the state-of-the-art sequential recommendation model, which consequently improves user satisfaction. Our implementation is accessible at https://github.com/yejinjkim/seq-div-rec for reproducibility. Our main contributions are:

- To the best of our knowledge, our work is the first to address the recommendation of *diverse* items in the context of *sequential* recommendation.
- We substantially increase *diversity* while preserving the state-of-the-art accuracy.
- Our method is an *end-to-end* approach that derives a ranking function to directly recommend general items and diverse items at the same time.

2 Related work

2.1 Diverse Recommendation

Typical approach for increasing recommendation diversity is a post-processing heuristic that re-ranks recommended items based on certain diversity metrics [Ziegler *et al.*, 2005; Christoffel *et al.*, 2015; Adomavicius and Kwon, 2012]. That is, the re-ranking methods generate a candidate set of items based on an accuracy measure, then select and re-arrange these items to maximize a diversity measure (Fig. 2). Some studies utilize long-tail items to further improve diversity. Clustering approaches [Park and Tuzhilin, 2008; Park, 2013; Bradley *et al.*, 2000; Yin *et al.*, 2012; Oh *et al.*, 2011] are proposed that attempt to leverage long-tail items directly into a recommendation list. They cluster tail items in preprocessing phase, treat the clusters as general items, and apply ordinary recommendation models so that the recommendation list contain the tail items as well.

However, these heuristic methods generate recommendation lists using limited candidate items and a single diversity metric regardless of correlation with user preference on general items. Moreover, such a re-ranking heuristic always requires extensive tuning engineering effort. To tackle this problem, the diverse recommendation requires a robust end-to-end model that is based on i) a supervised approach with ground truth of diverse items and ii) a learning-to-rank approach that directly learns a ranking function that imposes high values on diverse items as well as on general items. The first attempt on this method is a subset retrieval technique [Cheng *et al.*, 2017]. It learns a ranking function for selecting a set of diverse items using structural support vector machines. Likewise, our proposed model is also based on learning-to-rank framework with ground-truth tail items.

2.2 Sequential Recommendation

Recently RNN has been recognized as useful due to its ability to model variable-length sequences. The most representative work is the RNN-based recommendation [Hidasi *et al.*, 2015], which predict next item based on the sequence of previous items using RNN. A feature-rich version of the this model incorporates content feature vector as separate input [Hidasi *et al.*, 2016]. Likewise, we build our sequential model using the RNN framework like the previous studies. We do not limit the historical data as a short sequence without user ID as [Hidasi *et al.*, 2015] but let it be a user's implicit feedback sequence of any length (say a few days to a few hundreds days). Another line of study is next-basket recommendation, which is a task to formulate a customer who purchases a series of baskets of items [Rendle *et al.*, 2010; Yu *et al.*, 2016; Wang *et al.*, 2015]. These studies are similar with our models in that both aim to predict multiple relevant items in sequential manner. However, the next-basket recommendation does not consider a relative order among multiple relevant items, which is required for our models to preserve accuracy on predicting general items.

3 Sequential and Diverse Recommendation

We propose S-DIV, a sequential and diverse recommendation model that predicts next general items together with relevant diverse items. To make an end-to-end model without the re-ranking heuristic, we formulate our problem as supervised learning to rank.

3.1 Problem Formulation

In sequential recommendation, there is a mass of users, and each user consumes a series of items. Let $\mathbb{U}$ be a set of users, and $\mathbb{G}$ be a set of *general items*. For a user $U \in \mathbb{U}$, let $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_J)$ be an (general) item history from time 1 to J after omitting superscript U for simplicity. $\mathbf{x}_j$ is a vector that represents an item $G \in \mathbb{G}$ consumed at time j. Given an input of the item history, a ranking function f imposes scores on items, which indicates how likely each item can be the next recommended items, i.e., $\mathbf{s}_j = f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_j)$ or $\mathbf{s}_j = f(\mathbf{x}_j)$ in short. Here, $\mathbf{s}_j$ is a vector that represents ranking scores of all items including general items and diverse items at time j. We can rank the items using the scores in

$\mathbf{s}_j$ and generate a recommendation list R with top-N items at time j. Obviously, $\mathbf{s}_j$ should have high values on i) an item consumed at $j+1$ (i.e., $\mathbf{x}_{j+1}$) for accuracy and ii) relevant diverse items for diversity, so that both of them are ranked at top positions at the same time. To solve this supervised learning to rank problem, a good ground-truth ranking (or label) sequence is necessary, in which each ranking at specific time is ordered as next general item followed by relevant diverse items. A challenge is that there is no explicit ground-truth ranking sequence that includes diverse items. So, we discuss how to derive a reasonable pseudo ground-truth ranking sequence.

3.2 Pseudo Ground-truth Ranking Sequence

Because there is no explicit ground truth on a user's preference on diverse items, we instead infer a pseudo ground truth that implicitly reflects an actual ground truth.

Consumed long-tail items. A good ground truth for diverse items should be not only diverse but also relevant to a target user. A recommendation list containing items that are diverse but not accordant with the target user's taste does not help increase user satisfaction. We find long-tail items [Anderson, 2004] that users have consumed in the past are a good source of pseudo ground truth to predict future diverse items because the tail items are both diverse and highly relevant to the target user. In details, a user's consuming behavior follows a probabilistic relationship: $p(consumed) \sim p(seen) \cdot p(relevant)$ [Vargas and Castells, 2011]. The consumed tail items are obviously *consumed* and rarely *seen* since they are less exposed than popular items. We can infer that the tail items are so *relevant* (i.e., $p(relevant)$ is high) that users consume them as actively searching them. However we cannot directly use the individual tail items as a target to predict during training due to too few number of occurrences.A traditional treatment to the tail items is deleting or ignoring them in pre-processing phase. Instead, we make use of the tail items by clustering and relocating (Fig. 3).

Clustering. Apart from $\mathbb{G}$, there is a separate set $\mathbb{T}$ of *tail items* whose number of occurrences for each is less than negligibly small number τ (say 3). We first assign the tail items $T \in \mathbb{T}$ into a cluster $\overline{T} \in \overline{\mathbb{T}} = \{\overline{T}_1, \overline{T}_2, \ldots, \overline{T}_K\}$. Specifically, we perform K-means clustering ($|T| \geq K = |\overline{T}|$) using tail item's content vector so that contextually or semantically similar items are grouped together. We replace tail item T in the item history with the corresponding cluster $\overline{T}$ (i.e., if T belongs to cluster $\overline{T}$, then we use $\overline{T}$ instead of T). For example, let us assume an item sequence $G_1 \rightarrow T_2 \rightarrow G_3 \rightarrow T_4 \rightarrow T_5 \rightarrow G_6$ is given, where T_2, T_4, T_5 are tail items clustered into $\overline{T}_2, \overline{T}_2, \overline{T}_3$, respectively (Fig. 3). Traditional sequential recommendation approach treats it as (i) $G_1 \rightarrow G_3 \rightarrow G_6$ after deleting T_2, T_4, and T_5 in pre-processing phase. Instead, we treat it as (ii) $G_1 \rightarrow \overline{T}_2 \rightarrow G_3 \rightarrow \overline{T}_2 \rightarrow \overline{T}_3 \rightarrow G_6$. Clusters of tail items can effectively represent characteristics of similar individual tail items. The number of occurrences of each cluster is a sum of the number of occurrences of individual tail items that belong to the cluster. Thus, the cluster can mitigate the cold-start problem by providing more occurrences than individual tail items do [Park and Tuzhilin, 2008]. When we test our model, we replace the predicted tail cluster to an actual tail item that is closest to the centroid of the cluster.

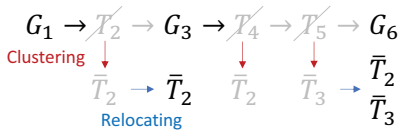

Figure 3: Clustering and relocating. Each tail item T is replaced with assigned cluster $\overline{T}$. Tail cluster $\overline{T}$ is then relocated to the next following general item G.

Since during training the tail clusters are surrogates of tail items, it is not desirable that each tail cluster occurs more frequently than general items. Some sophisticated clustering methods can balance the number of items in a cluster not to exceed the threshold τ [Park, 2013]. However, these methods rely on discrete optimization or computationally expensive operations, and do not work well with our web-scale data. Instead, we use mini-batch K-means clustering [Sculley, 2010], and set the number K of clusters large enough (i.e., $\lfloor \#\text{tail items}/\tau \rfloor$) so that the number of items in each cluster roughly does not exceed τ.

Relocating. However, we cannot use the new sequence (ii) $G_1 \rightarrow \overline{T}_2 \rightarrow G_3 \rightarrow \overline{T}_2 \rightarrow \overline{T}_3 \rightarrow G_6$ as it stands. A majority of existing (sequential) recommendation models are trained using the sequence (i) $G_1 \rightarrow G_3 \rightarrow G_6$ without tail items and predict next general items. On the other hand, if our model is trained with the new sequence (ii) containing tail clusters, it predicts either next general items or next tail items. It can cause two problems: First, our models' accuracy for predicting either next general item or tail items is not comparable to the existing recommendation models' accuracy for predicting next general item only, and thus our model may lose accuracy for predicting next general item, which is the most important task in recommendation. Secondly, our goal is to predict both next general items and tail items at the same time in one recommendation list, not to predict either next general items or next tail items. To address these problems, the sequence (ii) should maintain the general items' original sequence for preserving accuracy (i.e. $G_1 \rightarrow G_3 \rightarrow G_6$), and also it should consist of an ordered set of general items and tail items (not a single item) so that our models can predict both items at the same time. To do so, we relocate the tail cluster $\overline{T}$ to next to upcoming general item $G \in \mathbb{G}$ so that the general item G and tail cluster $\overline{T}$ occur concurrently (Fig. 3). This concurrency of general item and tail cluster is beneficial to capture user's preference on general items and also tail items while maintaining accuracy for predicting next general items. For example, we treat the sequence (ii) as $G_1 \rightarrow< G_3, \overline{T}_2 >\rightarrow< G_6, \overline{T}_2, \overline{T}_3 >$ where $\overline{T}_2$ is positioned right after G_3, and $\overline{T}_2, \overline{T}_3$ are positioned right after G_6. That means, after consuming G_1 the user is likely to consume G_3 along with $\overline{T}_2$. Similarly, after consuming G_3, the user is likely to consume G_6 along with $\overline{T}_2$ and $\overline{T}_3$. So, the input sequence is general item history $G_1 \rightarrow G_3$ (note that input sequence does not need to contain tail items), and the label sequence is a sequence of rankings $< G_3, \overline{T}_2 >\rightarrow< G_6, \overline{T}_2, \overline{T}_3 >$, where each ranking is ordered as the next general item followed by relevant tail clusters.

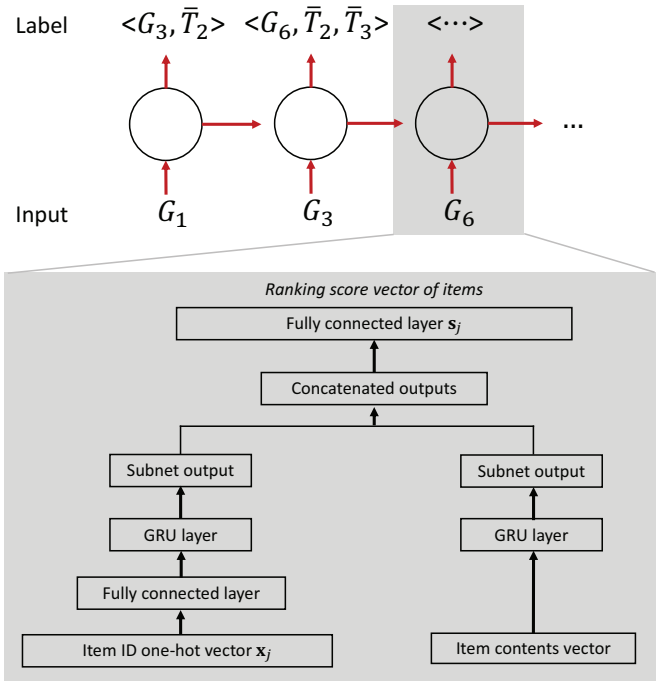

Figure 4: Input and label sequences from example in Fig. 3 (up) and model architecture at specific time j (down). Input sequence is each user's item history, and label sequence is a sequence of rankings on next upcoming general item and relevant tail clusters. Model architecture at specific time j consists of two subnetworks – item ID one-hot GRU and content vector GRU – concatenating layer and final fully connected layer.

3.3 GRU-based Recommendation

We define our recommendation model's ranking function f using a gated recurrent unit (GRU).

GRU. The GRU is one of RNNs that employs a gating recurrent unit to maintain memory due to vanishing gradient problem [Cho *et al.*, 2014]. The GRU learns how much to update and forget the previous hidden state. We set the GRU as a basic sequential recommendation model since GRU shows promising accuracy in sequential recommendation due to its ability to model variable-length sequences [Hidasi *et al.*, 2015; Hidasi *et al.*, 2016]. Moreover, the GRU is particularly suitable for diverse recommendation because we can design GRU architecture to learn tail item preference together with general item preference *seamlessly*. Also the GRU is flexible to incorporate other types of input, so we can easily add auxiliary information such as content feature, which can help increase diversity.

Item content feature. We also feed an item content vector into our model as a separate input. The item content is beneficial for diversity because by mapping items into a content (latent) space, the items are not limited to specific item IDs but generalized to certain broad contexts with high variability. We focus on extracting content vectors from blog articles' texts. After extracting nouns from title and text, we use the word2vec model to derive each word's vector representation [Mikolov *et al.*, 2013]. We set the size of the content vector as 120 after several trials. We then aggregate all the word vectors in one text as a weighted sum of TF-IDF scores. For the tail clusters, we use a centroid of each cluster.

Architecture. Our model's architecture consists of two parallel subnetworks (in which each takes input from either item ID one-hot vector or content vector), concatenating layer, and final fully connected layer (Fig. 4). This architecture is with respect to input-label pair at specific time in the sequence. Specifically, for the item ID side, we first apply a fully connected layer and a GRU layer to the item ID input. Likewise, for the item content side, we apply a GRU layer to the item content vector. We take the best merging structure from [Hidasi *et al.*, 2016]: the outputs from the two separate GRU models are concatenated. Finally, we apply another fully connected layer to transform the concatenated output into the item ranking scores $\mathbf{s}_j$. At time j, the GRU layers in the item side and content side take inputs from previous GRU layers at $j-1$ together with current item information at j. They output predicted ranking scores and deliver accumulated information to the next GRU layers at $j+1$. As the GRU layer memorizes the previous behaviour of the target user, our model can be aware of when the user explores long tail and which area (or topic) the user is interested in.

3.4 Loss Function by Permutation Probability

To optimize our GRU-based ranking model, we need a loss function that can rank high both next general item and relevant tail clusters in the pseudo ground-truth ranking. However, a challenge is to impose high ranking scores on multiple items while preserving their relative order. For implicit feedback data, most ranking loss functions, such as pairwise Bayesian personalized ranking (BPR) loss [Rendle *et al.*, 2009], cannot model the relative order among multiple relevant items. Thus, we use a listwise learning-to-rank approach specifically motivated by ListMLE [Xia *et al.*, 2008]. It trains the model to derive ranking scores that agree with a given ground-truth ranking order of items. Here the ranking order has one-to-one correspondence with permutation, so, we can represent a likelihood of the ranking order as a permutation probability, which is defined with Plackett-Luce model [Xia *et al.*, 2008]. In our model, the pseudo ground-truth ranking $\boldsymbol{\pi}$ is a partial order of items that consists of an accurate general item followed by relevant tail items; the other irrelevant items are not ordered. So, we define a top-n version of the permutation probability. For a user U and at time j (subscript for U and j are omitted for simplicity), the top-n permutation probability is

$$P(\boldsymbol{\pi}|\mathbf{s}) = \prod_{i=1}^{n} \frac{\exp s_{\pi_i}}{\sum_{l=i}^{n} \exp s_{\pi_l} + \sum_{\pi' \notin \boldsymbol{\pi}} \exp s_{\pi'}} \qquad (1)$$

where s_π be a ranking score of item $\pi \in \mathbb{G} \cup \overline{\mathbb{T}}$ from ranking scores vector $\mathbf{s} = f(\mathbf{x})$, and $\boldsymbol{\pi} =< \pi_1, \ldots, \pi_n >$ be a top-n ranking order (or partial permutation) out of all $|\mathbb{G}| + |\overline{\mathbb{T}}|$ items. We aim to maximize the log-likelihood $\log P(\boldsymbol{\pi}_j|\mathbf{s}_j)$ of the ranking model $\mathbf{s}_j = f(\mathbf{x}_j)$ with input $\mathbf{x}_j$ and label $\boldsymbol{\pi}_j$ for time 1 to J. So, we define our loss function L as a negative of the log-likelihood for time 1 to J: $L = -\frac{1}{J}\sum_{j=1}^{J} \log P(\boldsymbol{\pi}_j|f(\mathbf{x}_j))$. Minimizing this loss function L makes the recommendation model $f(\mathbf{x}_j)$ to give the highest value on the next general item and the second highest value on the following relevant tail item, and so on for

all items in $\boldsymbol{\pi}_j$. This listwise approach additionally allows a ranked list a room to contain diverse items. We minimize L using gradient descent with respect to parameters in $f(\mathbf{x}_j)$ so that the ranking scores vector $\mathbf{s}_j$ is accordant with the pseudo ground-truth ranking $\boldsymbol{\pi}_j$.

4 Experiments

4.1 Data

We perform offline experiments and online A/B tests on users' historical logs on clicking blog articles from a commercial blog platform, *Kakao* (https://brunch.co.kr), which is one of largest blog platforms in South Korea with 4 million daily views. For training, we collect 2.2 million active users' click logs during 8 days. The total number of articles is 263,016 after discarding sequences of length one (i.e., only one item) and items without text (e.g. articles with only images). The maximum length of sequence is 796 and average length is 3.6. The content-based features are extracted from the blog article's main text and its title.

4.2 Hyperparameters

We perform experiments with different setting of tail thresholds. After sorting items in decreasing order of the number of occurrences, we consider top 10%, 20%, and 50% of items as general items and the remaining as tail items. τ is the number of occurrences at 10%, 20%, and 50% percentile. We choose the number K of clusters as $\lfloor \#\text{tail items}/\tau \rfloor$. To compare the effect of separating tail set from general set, we perform experiment when $\tau = 0$, at which we regard all the items as general items. The size N of a recommendation list is set to 20. To discover best performing length l_{emb} of item ID input embedding (at fully connected layer) and length l_{hid} of hidden unit in GRU layer, we first run experiments as randomly assigning them in $500 \leq l_{emb} \leq 2500$ and $50 \leq l_{hid} \leq 750$ ($l_{hid} \leq l_{emb}$), and then perform grid search within promising ranges. After 100 trials, we found $l_{emb} = 900, l_{hid} = 550$ performs best. We set dropout rate as 0.1, mini-batch size as 1024, and the number of epochs as 20. We use adaptive subgradient optimizer. In the offline experiments, we split the data into 70% for training, 10% for validation, and 20% for test by user IDs.

4.3 Baselines

Item nearest neighbor (item-NN) is a simple but powerful baseline for sequential recommendation [Hidasi *et al.*, 2015]. Item-NN recommends a set of items with highest similarity. The similarity is measured as the number of sequences in which the two items occur together divided by the product of each item's support.

GRU4rec is the most widely used GRU model for sequential recommendation [Hidasi *et al.*, 2015] (without considering diversity). It focuses on improving accuracy to predict next accurate general item. It uses pairwise learning to rank (i.e., BPR loss) for optimization.

GRU4rec+CB is an extension of GRU4rec with content-based features. It incorporates content as a parallel structure to increase accuracy of recommendation [Hidasi *et al.*, 2016].

Re-ranking is a post-processing model based on given ranked list [Adomavicius and Kwon, 2012]. We apply this re-ranking model on top of GRU4rec. For specific configuration, we select top $3N$ items, sort them in increasing order of probability of being seen, and select top N items whose ranking value is larger than a median of the top $3N$ items.

S-DIV is our proposed model with item ID input.

S-DIV+CB is our proposed model with item ID and content feature input.

Top %	Model	Accuracy		Diversity	F_{div}
		MAP	NDCG		
10%	Item-NN	0.121	0.171	9.7%	0.107
	GRU4rec	**0.205**	**0.280**	7.4%	0.109
	GRU4rec+CB	0.204	0.279	9.6%	0.130
	Reranking	0.180	0.257	7.4%	0.105
	S-DIV	**0.205**	**0.280**	59.2%	0.305
	S-DIV+CB	**0.205**	**0.280**	78.8%	**0.325**
20%	Item-NN	0.119	0.169	19.5%	0.148
	GRU4rec	0.198	0.271	9.5%	0.129
	GRU4rec+CB	**0.199**	**0.272**	18.1%	0.190
	Reranking	0.174	0.248	9.6%	0.124
	S-DIV	**0.199**	**0.271**	22.4%	0.211
	S-DIV+CB	**0.199**	**0.271**	64.0%	**0.303**
50%	Item-NN	0.117	0.165	45.8%	0.187
	GRU4rec	0.191	0.260	9.8%	0.129
	GRU4rec+CB	**0.193**	**0.263**	30.2%	0.236
	Reranking	0.167	0.238	9.8%	0.124
	S-DIV	0.191	0.260	10.0%	0.131
	S-DIV+CB	**0.193**	**0.263**	38.7%	**0.258**
100%	GRU4rec	0.186	0.254	9.6%	0.127
	GRU4rec+CB	0.187	0.255	26.5%	0.219
	Reranking	0.163	0.232	9.7%	0.121

Table 1: Offline experiments results. Accuracy, diversity, and their trade-off of baselines and proposed methods.

4.4 Evaluation Measures

Accuracy. We measure accuracy to examine whether our models maintain or lose accuracy compared to other baselines that are solely designed for improving accuracy. We use mean average precision (MAP) and normalized discounted cumulative gain (NDCG). MAP is a macro-averaging precision measure when a user is interested in finding many relevant items. NDCG uses graded relevance as a measure of gain, and the gain is discounted according to its ranking position. We empirically set the relevance level of general items and tail items as 1 and 0.8, respectively, because users usually care accurate general items more than diverse tail items.

Diversity and trade-off. We follow evaluation measures in [Cheng *et al.*, 2017]. Diversity is defined as a proportion of exposed items across all recommendation lists among total items. Specifically, it is measured as the total number of distinct items that have been recommended to at least one user divided by the total number of items [Adomavicius and Kwon, 2011; Yin *et al.*, 2012; Ziegler *et al.*, 2005]; it is between 0 and 1. We use F-score to evaluate trade-off between accuracy and diversity [Cheng *et al.*, 2017]. We define F_{div} as a harmonic mean of conflicting accuracy and diversity. We use the MAP as the accuracy in F_{div} because the MAP is between 0 and 1 (so comparable to diversity between 0 and 1).

4.5 Offline Experiments Results

Preserving accuracy. We observed that our models do not lose significant accuracy compared to the baselines that are solely designed for increasing recommendation accuracy. For all the tail threshold cases, MAP and NDCG of S-DIV and S-DIV + CB are comparable to that of GRU4rec and GRU4rec

Rank	Recommended items from GRU4rec
1	**Top 10 Men's Watches**
2	Top 5 Affordable Mechanical Watches
3	Top 4 Men's Luxury Watches
17	Controversial World History - Imphal Battle
18	The Camping Trip
Rank	Recommended items from S-DIV + CB
1	Top 5 Affordable Mechanical Watches
2	**Top 10 Men's Watches**
3	Top 4 Men's Luxury Watches
17	*Old Memories, Everlasting Fragrance*
18	World Best 11 Drinks

Table 2: Case study on recommendation lists derived from GRU4rec and S-DIV+CB. Bold item is the correct general item. Italic item is from long tail.

+ CB (Table 1). On the other hand, the other two baselines for increasing diversity, item-NN and GRU4rec + re-ranking lose significant accuracy.

Improving diversity. In terms of diversity, our proposed models utilize the tail items far more than the baselines do, and consequently increase recommendation diversity greatly for all the tail threshold cases. At 50% tail threshold, S-DIV+CB shows diversity of 0.3868 (i.e., 101,746 items out of total 263,016 items appear), whereas GRU4rec + CB shows diversity of 0.3022 (i.e., 79,472 items) (Table 1).

Effect of content-based feature. We found that incorporating content-based features increases diversity consistently since the content features map a specific item ID into a generalized concept with high variability. We also observed that the content features increase accuracy for high tail threshold (50%) when the number of general items used in the training phase is relatively larger than that of other tail thresholds. This is because the content features provide supplementary information for cold items that have too few occurrences.

Case study. To examine the recommended items in detail, we retrieve two top-20 recommendation lists from GRU4rec and S-DIV + CB at the 50% tail threshold given the same input in the test set (Table 2). We observed that the recommendation list derived from S-DIV + CB contains a tail item, *Old Memories* as well as the correct general item, *Top 10 Men's Watches*. The recommendation list derived from GRU4rec also contains the correct general item but does not contain any tail items. Note that the two recommendation lists have several items in common (e.g. *Top 5 Affordable Mechanical Watches*), which indicates that S-DIV + CB can catch the preference on general items accurately as much as GRU4rec does while learning the preference on tail items.

4.6 Online A/B Tests Results

To further demonstrate the diversity improvement of our models, we conduct online A/B tests on our commercial blog platform. We launch GRU4rec, S-DIV, GRU4rec + CB, and S-DIV+ CB. We also run a baseline model that recommends manually curated items by human editors, which are mainly most popular items for the last 1 hour written by certified bloggers. This baseline shows empirically stable click-through rate (CTR) for all times and thus becomes a good anchor for comparing with other models.

Similar to the offline experiments, we train our models using a training set from around 2.2 million active users' historical click logs for the last 8 days before the start date of the A/B tests. We regard top 20% of frequent items as general items. We test our models and baselines with randomly selected 5% of total active users, which would be negligible size for service stability. To ensure the substantial size of the tested users for each model, we run two separated sets of experiments: i) comparing GRU4rec vs. S-DIV and ii) comparing GRU4rec + CB vs. S-DIV + CB, together with the curated-popular-items baseline. Each A/B test lasts for 2 days on both mobile and desktop sites. We recommend 10 articles out of 20 articles after filtering out items that the user has already read. We use the same evaluation measure for diversity but use CTR for accuracy.

	CTR Ratio	Diversity
Baseline	1	0.090%
GRU4rec	1.212	0.932%
S-DIV	1.118	2.393%
Baseline	1	0.082%
GRU4rec + CB	1.022	3.452%
S-DIV + CB	1.007	6.811%

Table 3: Online experiments results. Accuracy (CTR) and diversity of baselines and proposed methods.

As a result, similar to the offline experiments, we demonstrate that our models increase diversity while preserving accuracy for predicting accurate general items. We observed that CTR values of S-DIV and S-DIV + CB is comparable to that of GRU4rec and GRU4rec + CB, respectively (Table 3). Instead of directly reporting CTR values, we compute a CTR ratio as CTR value of a certain model divided by a CTR value of the baseline that corresponds to the same time period. Meanwhile, we also observed that diversity of S-DIV and S-DIV + CB is larger than that of GRU4rec and GRU4rec + CB, respectively (Table 3). Particularly, S-DIV + CB generates the most diverse recommendation and thus mitigates the severe long tail problem. Note that the diversity values of the online experiments is much smaller than that of the offline experiments because new articles are being created very fast, and the size of test set for each model is less than 5% of total active users during 2 days.

5 Conclusion

We propose a sequential and diverse recommendation model that predicts a ranked list containing general items and also tail items as learning temporal preference on them. To mitigate the cold-start problem of long tails, we cluster the tail items and incorporate content-based vector. To increase diversity while preserving accuracy, we make the pseudo ground truth to contain both an accurate general item and relevant tail items at the same time. We use RNN to directly learn a ranking function with this ground truth, and optimize the network using listwise learning-to-rank loss to preserve the relative order among the general item and tail items. Extensive online and offline experiments on a commercial blog platform demonstrate that our models significantly increase diversity while preserving accuracy compared to the state-of-the-art sequential recommendation model.

Acknowledgements

This work was supported in part by the Kakao, CPRIT (RR180012), and NIH (R01GM124111).

References

[Adomavicius and Kwon, 2011] Gediminas Adomavicius and YoungOk Kwon. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *(DiveRS 2011) RecSys*, pages 3–10, 2011.

[Adomavicius and Kwon, 2012] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE TKDE*, 24(5):896–911, 2012.

[Anderson, 2004] Chris Anderson. The long tail. *Wired magazine*, 12(10):170–177, 2004.

[Antikacioglu and Ravi, 2017] Arda Antikacioglu and R Ravi. Post processing recommender systems for diversity. In *KDD*, pages 707–716. ACM, 2017.

[Bradley *et al.*, 2000] PS Bradley, KP Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, pages 1–8, 2000.

[Cheng *et al.*, 2017] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. Learning to recommend accurate and diverse items. In *WWW*, pages 183–192. ACM, 2017.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Christoffel *et al.*, 2015] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *RecSys*, pages 163–170. ACM, 2015.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[Hidasi *et al.*, 2016] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*, pages 241–248. ACM, 2016.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Oh *et al.*, 2011] Jinoh Oh, Sun Park, Hwanjo Yu, Min Song, and Seung-Taek Park. Novel recommendation based on personal popularity tendency. In *ICDM*, pages 507–516. IEEE, 2011.

[Park and Tuzhilin, 2008] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *RecSys*, pages 11–18. ACM, 2008.

[Park, 2013] Yoon-Joo Park. The adaptive clustering method for the long tail problem of recommender systems. *IEEE TKDE*, 25(8):1904–1915, 2013.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

[Sculley, 2010] David Sculley. Web-scale k-means clustering. In *WWW*, pages 1177–1178. ACM, 2010.

[Vargas and Castells, 2011] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *RecSys*, pages 109–116. ACM, 2011.

[Wang *et al.*, 2015] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. Learning hierarchical representation model for nextbasket recommendation. In *SIGIR*, pages 403–412. ACM, 2015.

[Xia *et al.*, 2008] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *ICML*, pages 1192–1199. ACM, 2008.

[Yin *et al.*, 2012] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *VLDB*, 5(9):896–907, 2012.

[Yu *et al.*, 2016] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *SIGIR*, pages 729–732. ACM, 2016.

[Ziegler *et al.*, 2005] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW*, pages 22–32. ACM, 2005.