Statistical Methods in Artificial Intelligence

NEXT ITEM RECOMMENDATION

Introduction

Next item recommendation is a sub field of sequential recommendation, which considers the order of single item with user general preference for the recommendation. These models are different from the general recommendation models which consider only the set of items without order.

This project aims at better utilization of sequential information in recommender systems. A new neural architecture, multi-scale Quasi-RNN for next item Recommendation (QR-Rec) task is implemented. Our model provides the best of both worlds by exploiting multi-scale convolutional features as the compositional gating functions of a recurrent cell. The model is implemented in a multi-scale fashion, i.e., convolutional filters of various widths are implemented to capture different union-level features of input sequences which influence the compositional encoder. The key idea aims to capture the recurrent relations between different kinds of local features, which has never been studied previously in the context of recommendation. Simple - gating functions are pre-learned via multi-scale convolution and then applied recursively in an auto-regressive fashion similar to a recurrent model.

Literature Survey

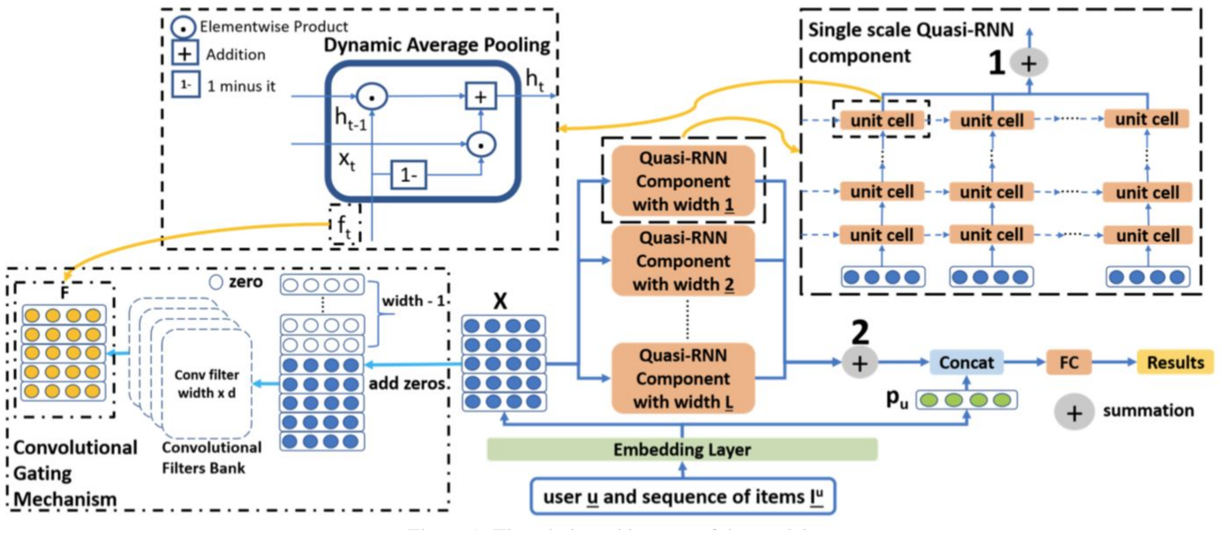
Early work in sequential recommendations mostly rely on the markov chain and feature-based matrix factorization approaches. Markov chain approaches failed to model union-level sequential patterns and did not allow skip behaviors in the item sequences. Factorization based approaches such as factorization machines model a sequence by the sum of its item vectors. However, these methods do not consider the order of items and are not specifically invented for sequential recommendations.

Restricted Bolzmann Machine (RBM) is the first successful 2- layers neural network that is applied to recommendation problems. Auto-encoder framework and its variant denoising auto- encoder also produce a good recommendation performance. Convolutional neural network (CNN) has been used to extract users' preferences from their reviews. None of these works is for sequential recommendation.

Recently, deep learning models have shown state-of-the-art recommendation accuracy in contrast to conventional models. Moreover, RNNs, a class of deep neural networks, have almost dominated the area of sequential recommendations.

By contrast, CNN based sequential recommendation models are more challenging and much less explored because convolutions are not a natural way to capture sequential patterns.

Architecture



Elementwise Product
Addition
1 minus it
Dynamic Average Pooling
h_t
h_{t-1}
x_t
1-
f_t
Single scale Quasi-RNN component
1
2
unit cell
Quasi-RNN Component with width 1
Quasi-RNN Component with width 2
Quasi-RNN Component with width L
F
zero
width - 1
X
add zeros
Conv filter width x d
Convolutional Gating Mechanism
Convolutional Filters Bank
Concat
FC
Results
p_u
summation
Embedding Layer
user u and sequence of items I^u

Fig : The whole architecture of the model

Methodology

Given U, the sets of users, I , the sets of items, each user is associated with a sequence of items he/she has interacted with in the past, I^u , we propose a sequential recommender that recommends considering the user behaviour as a sequence of ordered items instead of a set of items.

1. Embedding Layer

Through embedding layer, each item in the sequence of length L is mapped into a latent space with dimension d, as $x^u_{t-i+1} \in \mathbb{R}^d$ and i ∈ [1, 2, ..., L] at time step t − i + 1, the output of this layer $X^u_t \in \mathbb{R}^{d \times L}$ is a sequence embedding matrix of user u before time step t which concatenates all the item embeddings in the sequence.

$$X_t^u = [x_{t-L+1}^u; \cdots; x_{t-1}^u; x_t^u]$$

Also, each user u is mapped into the same latent space to obtain a user profile representation as $p_u \in \mathbb{R}^d$.

2. Multi-Scale Quasi-RNN

As the core module of QR-Rec, through convolution on input sequences, each Quasi-RNN component with a specific filter width (1 to L) with latent dimension d, can then obtain different union-level features.

2.1 Convolution Gating Mechanism

Convolutional gating mechanism aims to generate gates for the pooling layer. Like convolution layers in CNNs, it allows fully parallel computation across both mini batches and sequence dimension. Here we deduce the formulations below.

$$F_t^u = \sigma(W_{f,t}^u * X_t^u)$$

This is the simplest equation for width of 1.

$W_{f,t}^u \in \mathbb{R}^{k \times d \times m}$ is the convolution filters bank.

m is the number of filters and it is set the same as d for easy implementation.

$*$ denotes a masked convolution along the timestep dimension.

X_t^u is the input sequence embedding matrix.

$F_t^{\bar{u}} \in \mathbb{R}^{k \times m}$ is the forget gate matrix.

σ is the sigmoid function.

All the superscripts u and subscripts t indicate for user u before time step t here.

Here we generalize the formulation for the gates for width of N .

$$f_t^{u,N} = \sigma(W_{f,t}^{u,1} x_{t-N+1}^u + \ldots + W_{f,t}^{u,N-1} x_{t-1}^u + W_{f,t}^{u,N} x_t^u)$$

where $f_t^{u,N} \in \mathbb{R}^m$ is the forget gate vector, $W_{f,t}^{u,i} \in \mathbb{R}^{k \times d \times m}$ is the convolution filters bank where i ∈ [1, 2, ..., N] for u before time step t, and $x_i^u \in \mathbb{R}^d$ is the input vector where i ∈ [t − N + 1, t − N + 2, ..., t] indicates time step.

2.2 Dynamic Average Pooling

We seek to obtain a function controlled by gates that can mix states across time steps, but also acts independently on each channel of the state vector. Dynamic Average Pooling only consists of a forget gate for controlling the information flow within the recurrent structure, and gives the general equation:

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{x}_t$$

where $\mathbf{h}_t \in \mathbb{R}^d$ is the hidden state.

$\mathbf{f}_t \in \mathbb{R}^m$ is the forget gate.

$\mathbf{x}_t \in \mathbb{R}^d$ is the input item embedding at time step t.

2.3 Hierarchical Summation Aggregation

Inside QR-Rec, aggregation needs placing at two points: 1)The output of a specific scale Quasi-RNN component; 2) The output of the whole multi-scale Quasi-RNN module.

$$\mathbf{h}_{w,t}^u = \sum_{t_w=1}^{L} h_{t_w}^u$$

$$\mathbf{o}_t^u = \sum_{w=1}^{L} h_{w,t}^u$$

where $h_{t_w}^u \in \mathbb{R}^d$ is the hidden state obtained of unit cell of a Quasi-RNN component

where t_w ∈ [1, 2, ..., L] is time step for scale w.

$\mathbf{h}^u_{w,t} \in \mathbb{R}^a$ is the output of a Quasi-RNN component with width w where w ∈ [1, 2, ..., L] for u at t. Then another summation aggregation for Quasi-RNN is performed to obtain $\mathbf{o}^u_t \in \mathbb{R}^d$, the output for u before t.

3 . Prediction Layer and Recommendation

p_u, the user embedding or profile, which represents user general preference, is concatenated with $\bar{\mathbf{o}}^u_t$ to pass a fully connected(FC) layer to obtain scores for candidate items.

$$y^u_t = W^u_{s,t}[o^u_t; p_u] + b^u_{s,t}$$

where $W^u_{s.t} \in \mathbb{R}^{2\times d}$ and $b^u_{s.t} \in \mathbb{R}$ are parameters to learn for FC while y^u_t is the final scores, all for u before t.

Once the training is complete, u’s user embedding and his/her last L item sequence embeddings are given as inputs. We then recommend top N items with the highest scores of y^u_t.

Improvement Ideas

We can add a vertical convolution layer and concatenate it with the filter currently being used in CNN. CNN through two directions of filters achieves the state-of-the-art for sequential recommendation task.

We can use Ensemble Learning to improve the results or we can also use review data to extract more features for the quasi-RNN using various NLP algorithms.

Contribution of each team member

Mythri V. : Implemented the Embedding Layer, Convolution gating mechanisms for RNN. Added vertical filters to improve upon the performance.

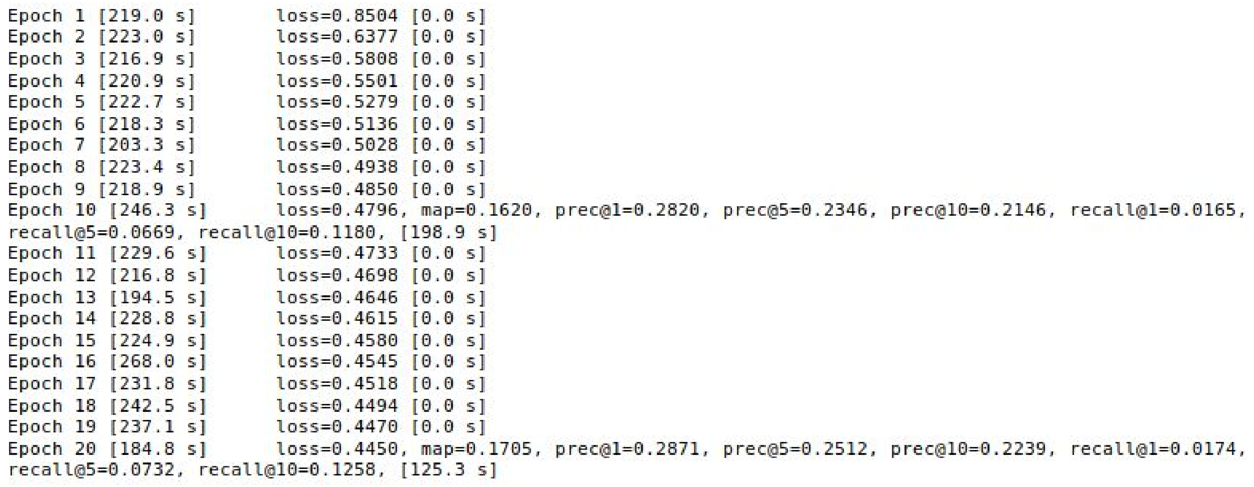
Shafiya Naaz: Contributed in adding Hierarchical Summation Aggregation on the model, and helped understand the multi scale convolutions.

Vishal Chugh: Contributed in data preprocessing, understanding recurrent Neural Network nuances for the model. Studied other existing models.

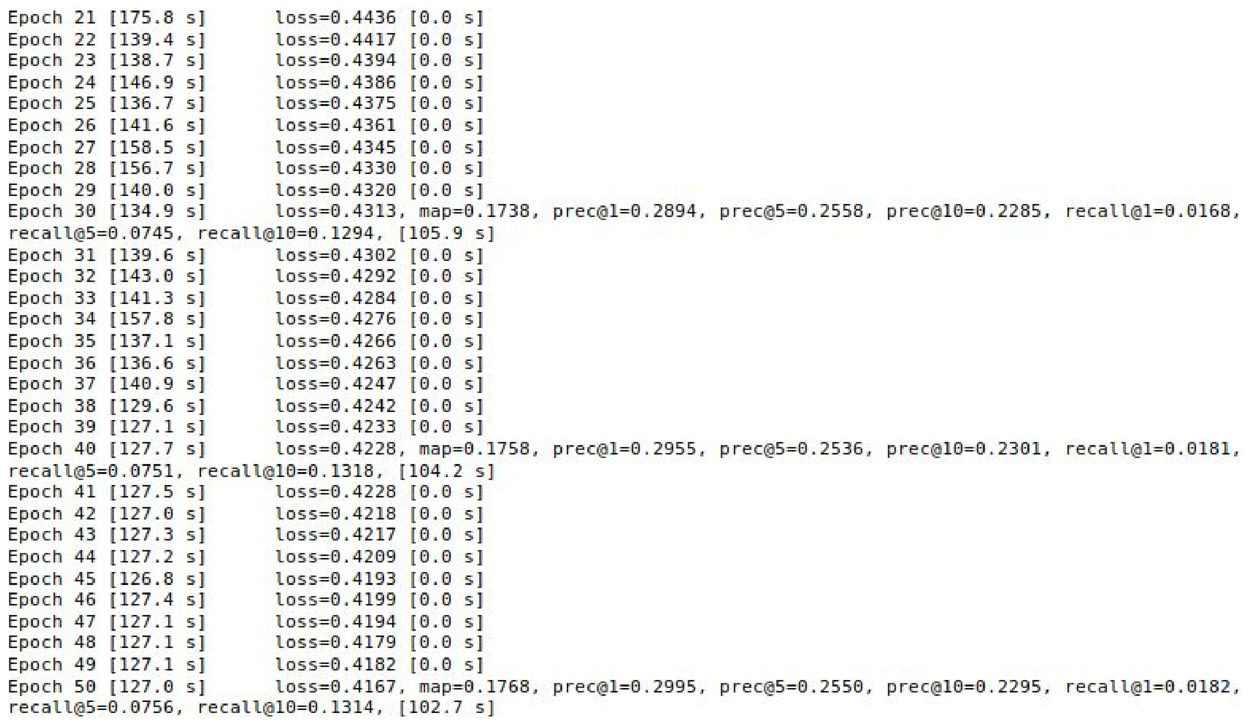
Smriti Bhati: Implemented Dynamic Average Pooling, Fully-Connected layer. Added vertical filters to improve upon the performance.

Conclusion

Extensive experiments show that QR-Rec is strong across various datasets and several quantitative analyses reveal the intuitions behind model components and hyperparameters.



```
Epoch 1 [219.0 s]       loss=0.8504 [0.0 s]
Epoch 2 [223.0 s]       loss=0.6377 [0.0 s]
Epoch 3 [216.9 s]       loss=0.5808 [0.0 s]
Epoch 4 [220.9 s]       loss=0.5501 [0.0 s]
Epoch 5 [222.7 s]       loss=0.5279 [0.0 s]
Epoch 6 [218.3 s]       loss=0.5136 [0.0 s]
Epoch 7 [203.3 s]       loss=0.5028 [0.0 s]
Epoch 8 [223.4 s]       loss=0.4938 [0.0 s]
Epoch 9 [218.9 s]       loss=0.4850 [0.0 s]
Epoch 10 [246.3 s]      loss=0.4796, map=0.1620, prec@1=0.2820, prec@5=0.2346, prec@10=0.2146, recall@1=0.0165,
recall@5=0.0669, recall@10=0.1180, [198.9 s]
Epoch 11 [229.6 s]      loss=0.4733 [0.0 s]
Epoch 12 [216.8 s]      loss=0.4698 [0.0 s]
Epoch 13 [194.5 s]      loss=0.4646 [0.0 s]
Epoch 14 [228.8 s]      loss=0.4615 [0.0 s]
Epoch 15 [224.9 s]      loss=0.4580 [0.0 s]
Epoch 16 [268.0 s]      loss=0.4545 [0.0 s]
Epoch 17 [231.8 s]      loss=0.4518 [0.0 s]
Epoch 18 [242.5 s]      loss=0.4494 [0.0 s]
Epoch 19 [237.1 s]      loss=0.4470 [0.0 s]
Epoch 20 [184.8 s]      loss=0.4450, map=0.1705, prec@1=0.2871, prec@5=0.2512, prec@10=0.2239, recall@1=0.0174,
recall@5=0.0732, recall@10=0.1258, [125.3 s]
```



```
Epoch 21 [175.8 s]      loss=0.4436 [0.0 s]
Epoch 22 [139.4 s]      loss=0.4417 [0.0 s]
Epoch 23 [138.7 s]      loss=0.4394 [0.0 s]
Epoch 24 [146.9 s]      loss=0.4386 [0.0 s]
Epoch 25 [136.7 s]      loss=0.4375 [0.0 s]
Epoch 26 [141.6 s]      loss=0.4361 [0.0 s]
Epoch 27 [158.5 s]      loss=0.4345 [0.0 s]
Epoch 28 [156.7 s]      loss=0.4330 [0.0 s]
Epoch 29 [140.0 s]      loss=0.4320 [0.0 s]
Epoch 30 [134.9 s]      loss=0.4313, map=0.1738, prec@1=0.2894, prec@5=0.2558, prec@10=0.2285, recall@1=0.0168,
recall@5=0.0745, recall@10=0.1294, [105.9 s]
Epoch 31 [139.6 s]      loss=0.4302 [0.0 s]
Epoch 32 [143.0 s]      loss=0.4292 [0.0 s]
Epoch 33 [141.3 s]      loss=0.4284 [0.0 s]
Epoch 34 [157.8 s]      loss=0.4276 [0.0 s]
Epoch 35 [137.1 s]      loss=0.4266 [0.0 s]
Epoch 36 [136.6 s]      loss=0.4263 [0.0 s]
Epoch 37 [140.9 s]      loss=0.4247 [0.0 s]
Epoch 38 [129.6 s]      loss=0.4242 [0.0 s]
Epoch 39 [127.1 s]      loss=0.4233 [0.0 s]
Epoch 40 [127.7 s]      loss=0.4228, map=0.1758, prec@1=0.2955, prec@5=0.2536, prec@10=0.2301, recall@1=0.0181,
recall@5=0.0751, recall@10=0.1318, [104.2 s]
Epoch 41 [127.5 s]      loss=0.4228 [0.0 s]
Epoch 42 [127.0 s]      loss=0.4218 [0.0 s]
Epoch 43 [127.3 s]      loss=0.4217 [0.0 s]
Epoch 44 [127.2 s]      loss=0.4209 [0.0 s]
Epoch 45 [126.8 s]      loss=0.4193 [0.0 s]
Epoch 46 [127.4 s]      loss=0.4199 [0.0 s]
Epoch 47 [127.1 s]      loss=0.4194 [0.0 s]
Epoch 48 [127.1 s]      loss=0.4179 [0.0 s]
Epoch 49 [127.1 s]      loss=0.4182 [0.0 s]
Epoch 50 [127.0 s]      loss=0.4167, map=0.1768, prec@1=0.2995, prec@5=0.2550, prec@10=0.2295, recall@1=0.0182,
recall@5=0.0756, recall@10=0.1314, [102.7 s]
```

Team Number: 36

Team Members

Smriti Bhati	**(2018201093)**
Mythri V	**(2018201028)**
Shafiya Naaz	**(2018201062)**
Vishal Chugh	**(2018202021)**