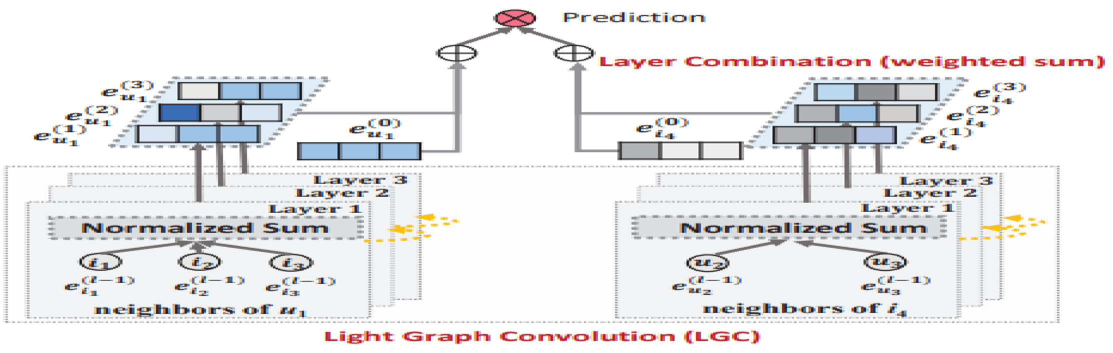
Advanced Neural Network Architectures for Sequential Recommendation

Group-12 Results & Analysis

1. Aamleen Ahmed
2. Ishaan Marwah
3. Jai Singh
4. Madhava Krishna
5. Sejal Kardam

LIGHT GCN MODEL

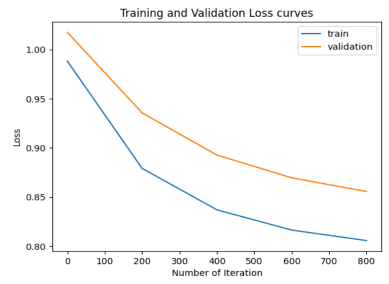
The paper aims to simplify the design of GCN to make it more concise and appropriate for a recommendation.
The proposed model, called LightGCN, uses a basic method called neighborhood aggregation to learn about users and items. It does this by looking at how users and items are connected in a graph and making a weighted average of their embeddings.

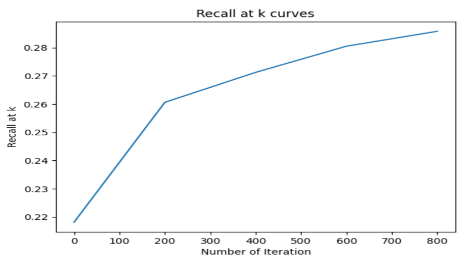


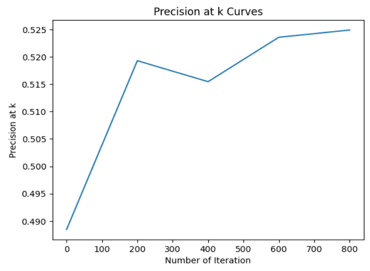
An Illustration of Light GCN Model

1. Movielens Dataset(100 K)

```
Parameters: Number of Iterations: 1000, Epochs: 10, K=10
```

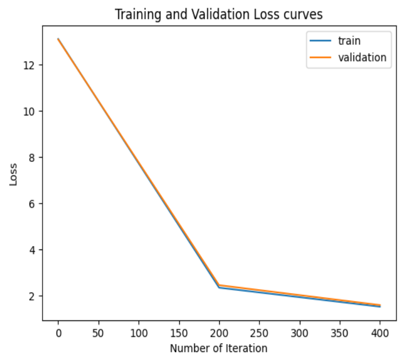


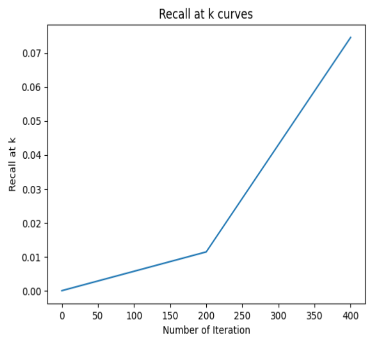


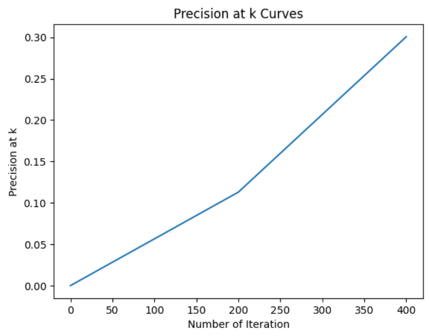


```
Recall at K = 0.30392 and Precision at K = 0.51476

Parameters: Number of Iteration:500, Epochs: 10, K=10
```

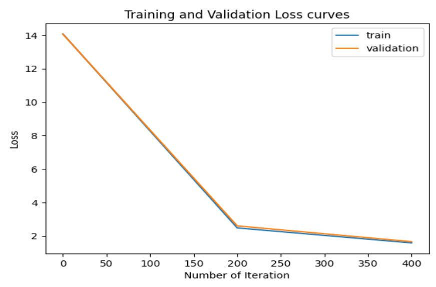


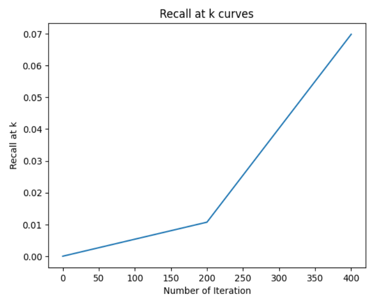


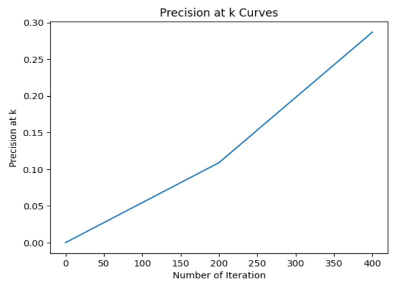


```
Recall at K=0.08509 and Precision at k=0.31086

Parameters: Number of Iterations: 600, Epochs: 10, K=10
```

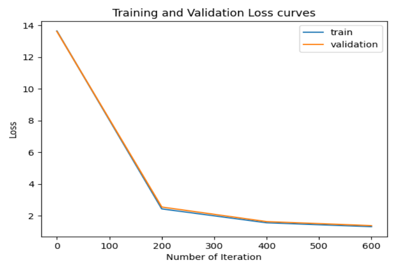


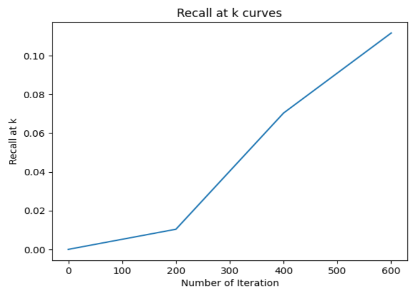


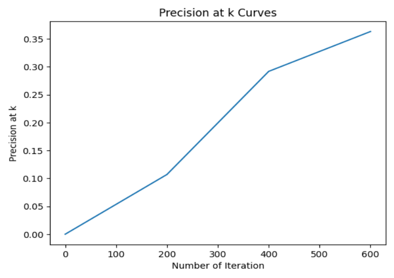


```
Recall at K=0.10052 and Precision at K=0.33419

Parameters: Number of Iterations: 700, Epochs: 10, K=10
```

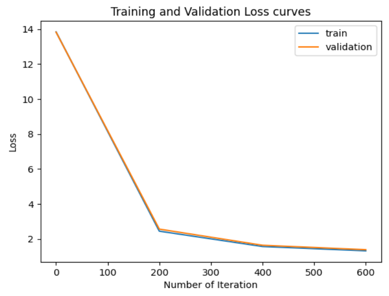


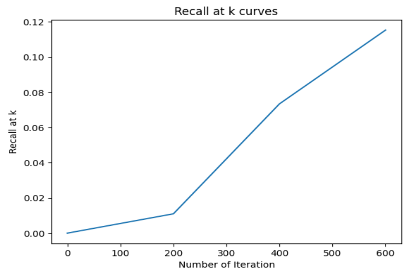


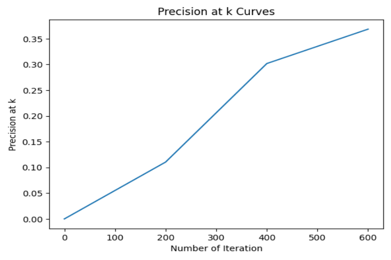


```
Recall at K=0.1198 and Precision at K=0.35972
```

```
Parameters: Number of Iteration: 800, Epochs: 10, K=10
```

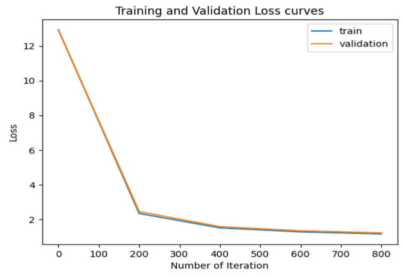


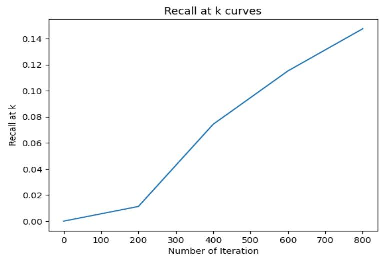


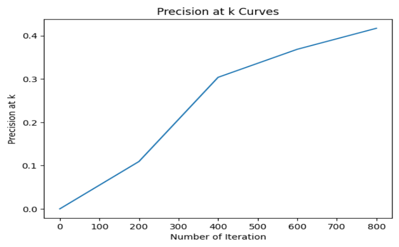


```
Recall at K=0.13726 and Precision at K=0.38427
```

```
Parameters: Number of Iteration = 900, Epochs = 10,K=10
```



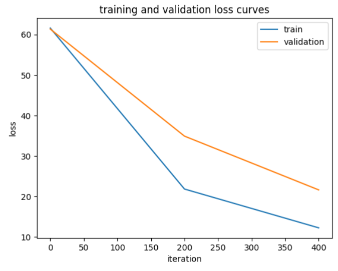


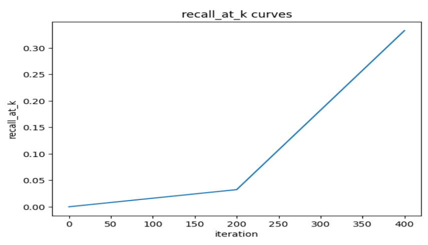


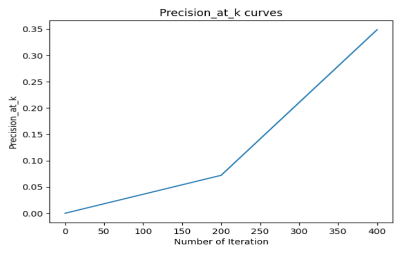
```
Recall at K=0.1517 and Precision at K=0.39381
```

2. Amazon Book Reading Dataset

```
Parameters: Number of Iteration= 500, Epochs=10,K=10
```

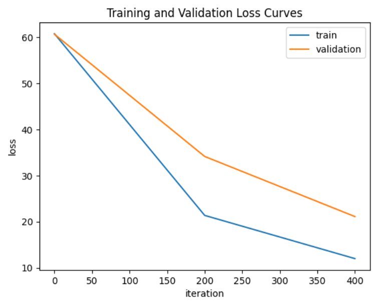


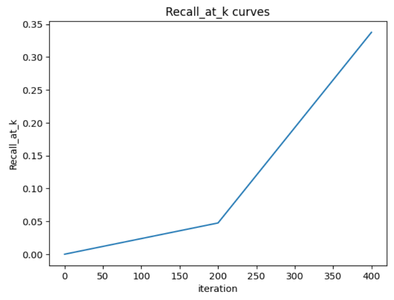


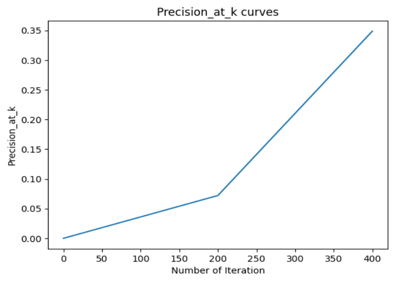


```
Recall at K=0.40337 and Precision at K=0.40639
```

```
Parameters: Number of Iteration = 600, Epochs = 10,K=10
```

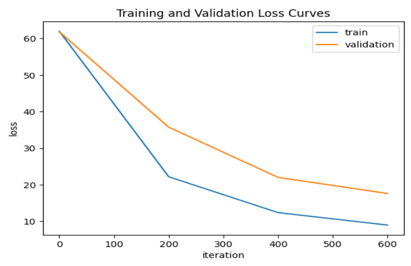


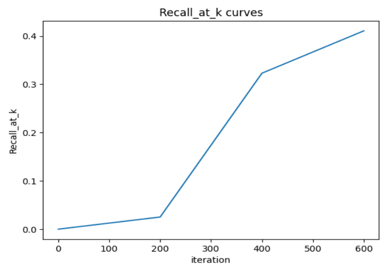


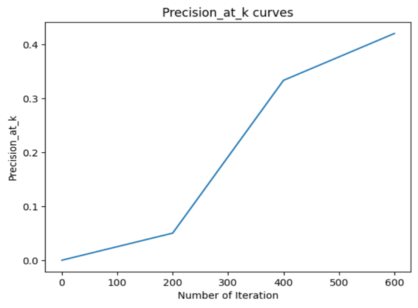


```
Recall at K=0.45246 and Precision at K=0.45584
```

```
Parameters: Numbers of Iteration=700, Epochs = 100, K=10
```

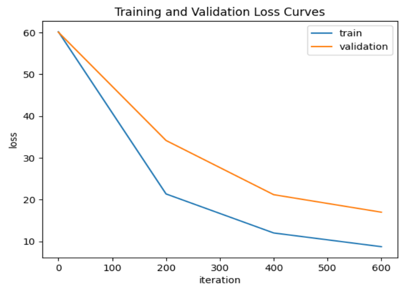


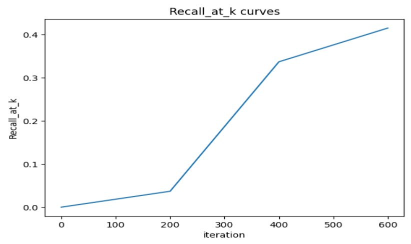


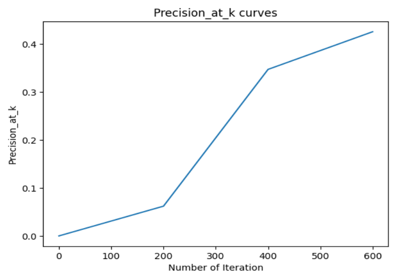


```
Recall at K=0.4759 and Precision at K=0.47782
```

```
Parameters Number of Iteration = 800 , Epochs = 10,K=10
```

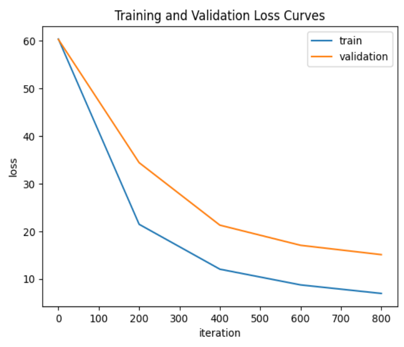


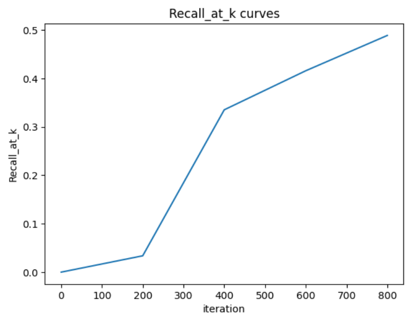


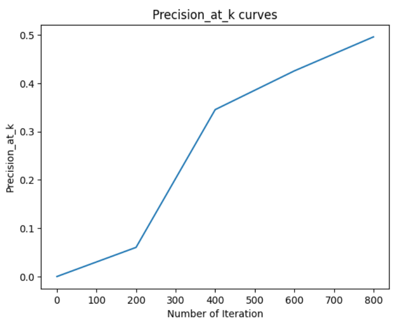


```
Recall at K=0.51766 and Precision at K=0.51518
```

```
Parameters: Number of Iteration = 900, Epoch = 10,K=10
```

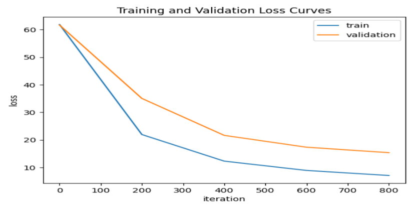


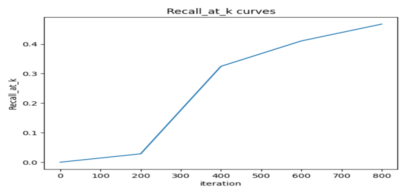


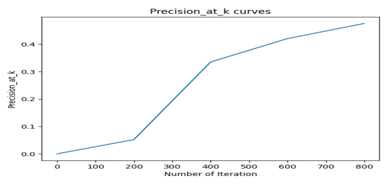


```
Recall at K=0.54401 and Precision at K=0.54145.

Parameters: Number of Iteration = 1000, Epochs = 10, K=10
```



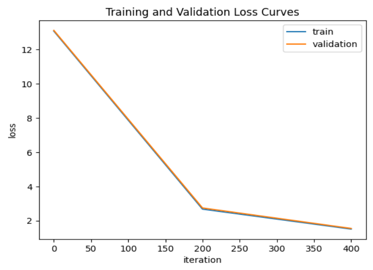


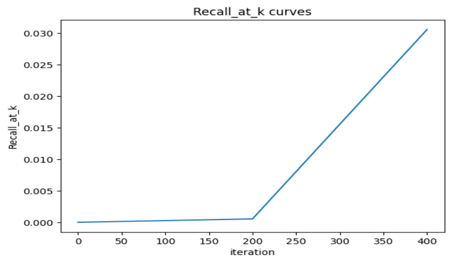


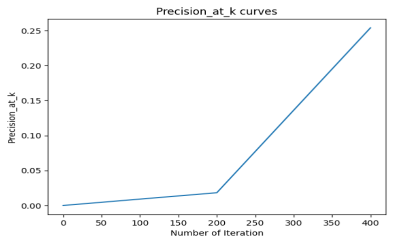
```
Recall at k=0.54623 and Precision at K=0.54375
```

(3) Learning from the sets of Items 2019

```
Parameters: Number of Iteration=500, Epochs = 10, K=10
```

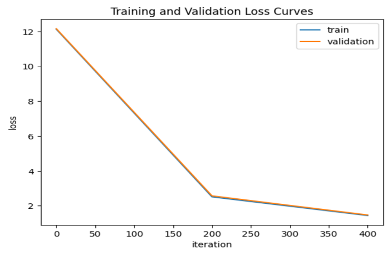


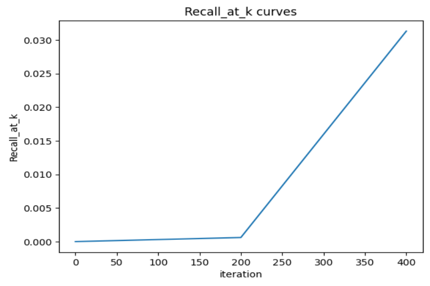


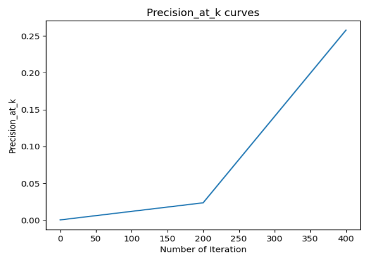


```
Recall at K=0.04748 Precision at K=0.29479

Parameters: Number of Iteration = 600, Epoch = 10, K= 10
```

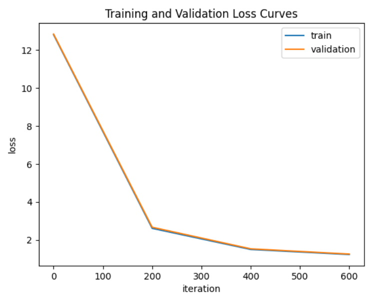


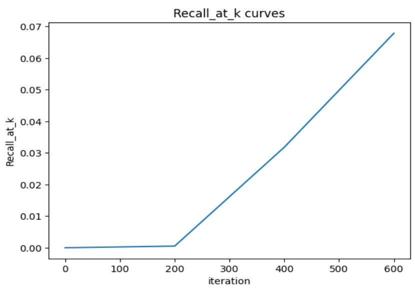


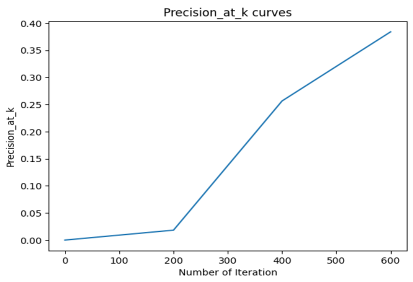


```
Recall at K=0.06567 Precision at K=0.34745

Parameters: Number of Iteration = 700, Epochs = 10, K=10
```

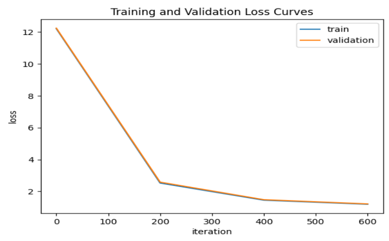


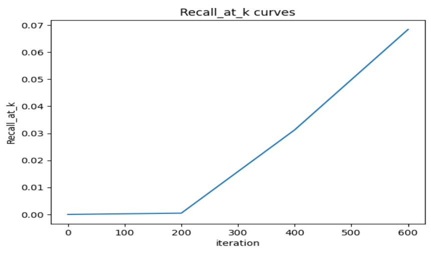


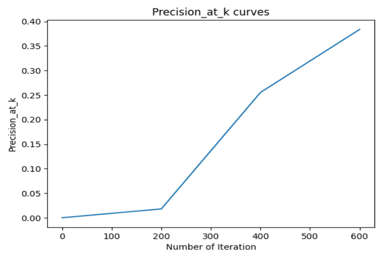


```
recall_at_k 0.08052, precision_at_k 0.38234

Parameters: Number of Iteration=800, Epochs = 10,k=10
```

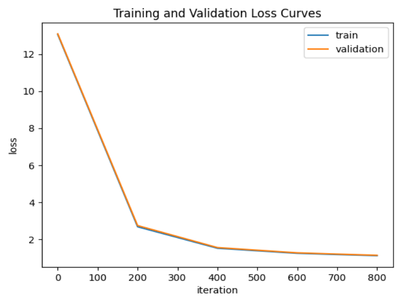


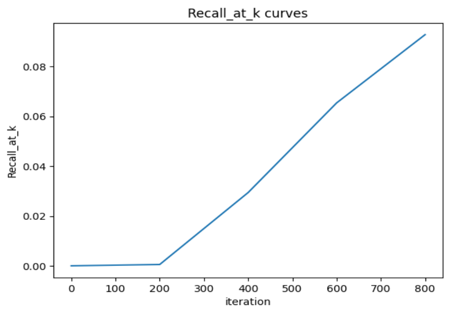


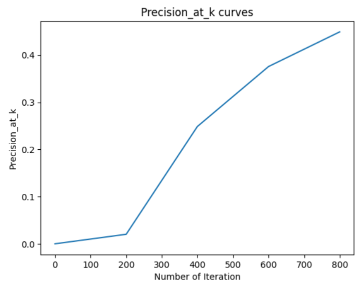


```
recall_at_k 0.09661, precision_at_k 0.4168

Parameters: Number of Iteration = 900, Epochs = 10, K=10
```

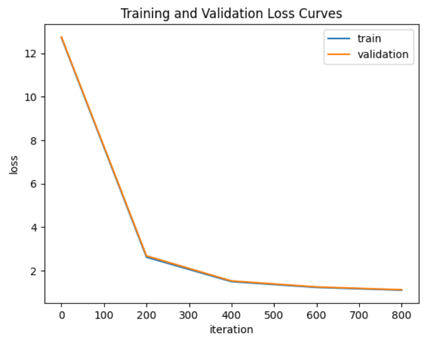


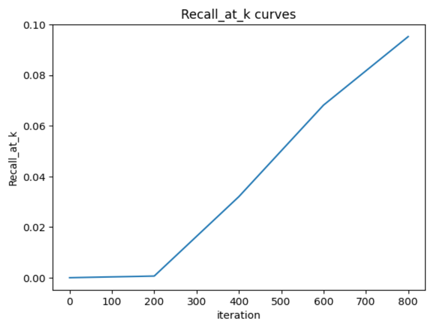


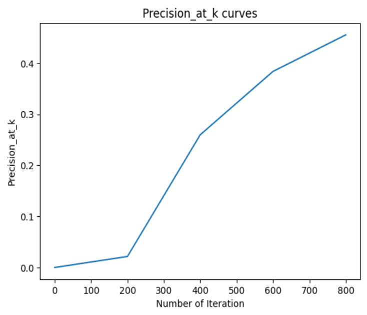


```
recall_at_k 0.10525, precision_at_k 0.42954

Parameters: Number of Iteration = 1000, Epochs = 10, K=10
```



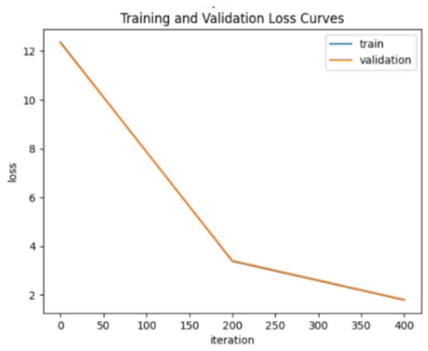


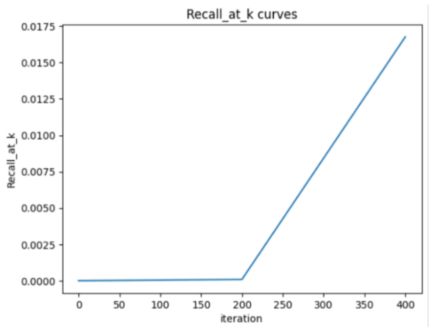


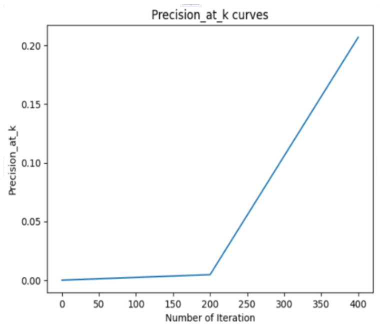
```
recall_at_k 0.12016, precision_at_k 0.4511
```

(4)Personality dataset

Parameters: Number of Iteration=500, Epochs=10, K=10

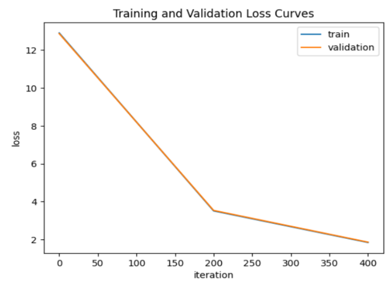


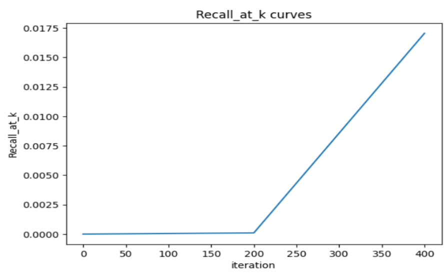


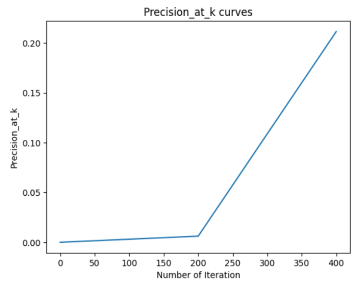


Recall at K=0.03357 and Precision at K = 0.25467

Parameters: Number of Iteration = 600, EPOCHS = 10, K = 10

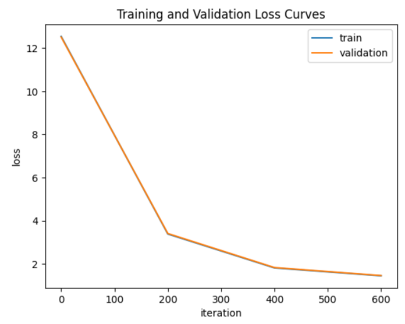


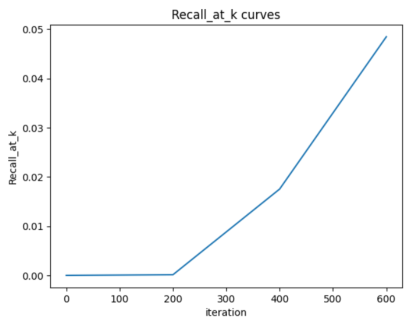


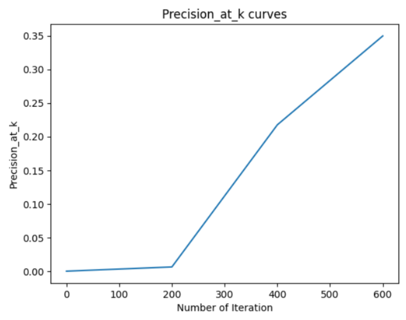


Recall at K = 0.04958 and Precision at K=0.30712

Parameters: Number of Iteration = 700, EPOCHS = 10, K = 10

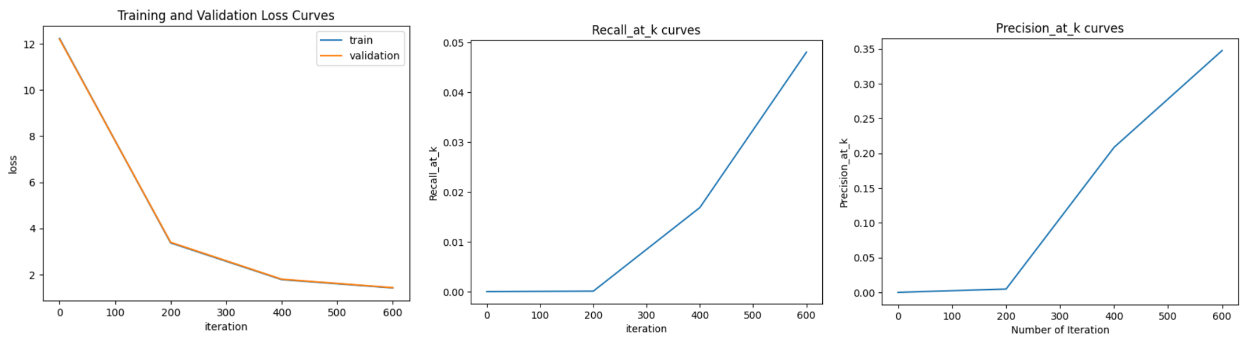






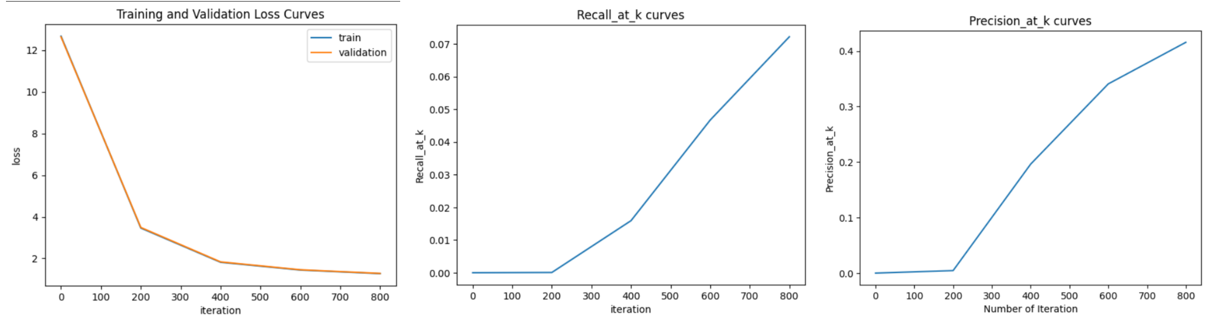
Recall at K = 0.06309 and Precision at K=0.34817

Parameters: Number of Iteration = 800, EPOCHS = 10, K = 10



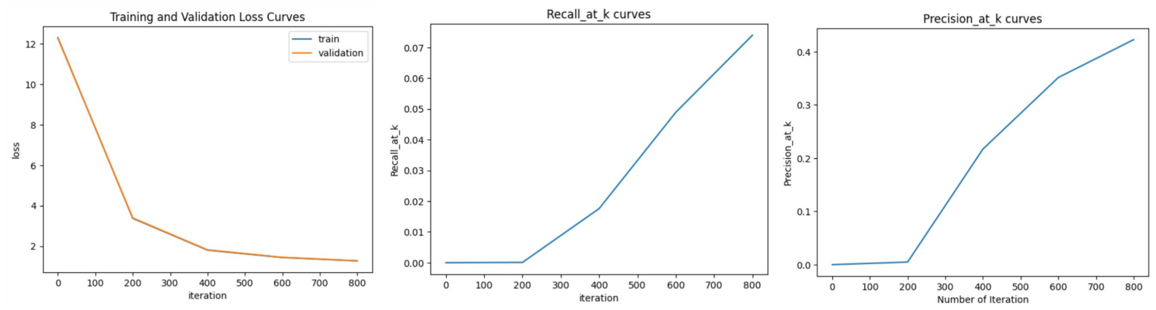
Recall at K = 0.07741 and Precision at K = 0.38285

Parameters: Number of Iteration = 900, EPOCHS = 10, K = 10



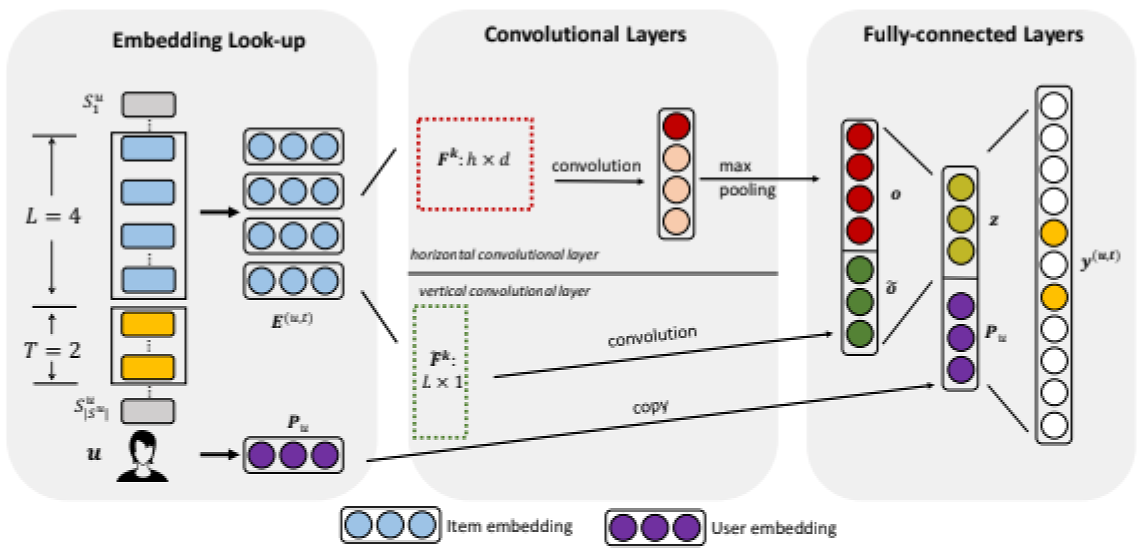
Recall at K = 0.08716 and Precision at K = 0.40228

Parameters: Number of Iteration = 1000, EPOCHS = 10, K = 10



Recall at K = 0.1002 and Precision at K = 0.43562

Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding

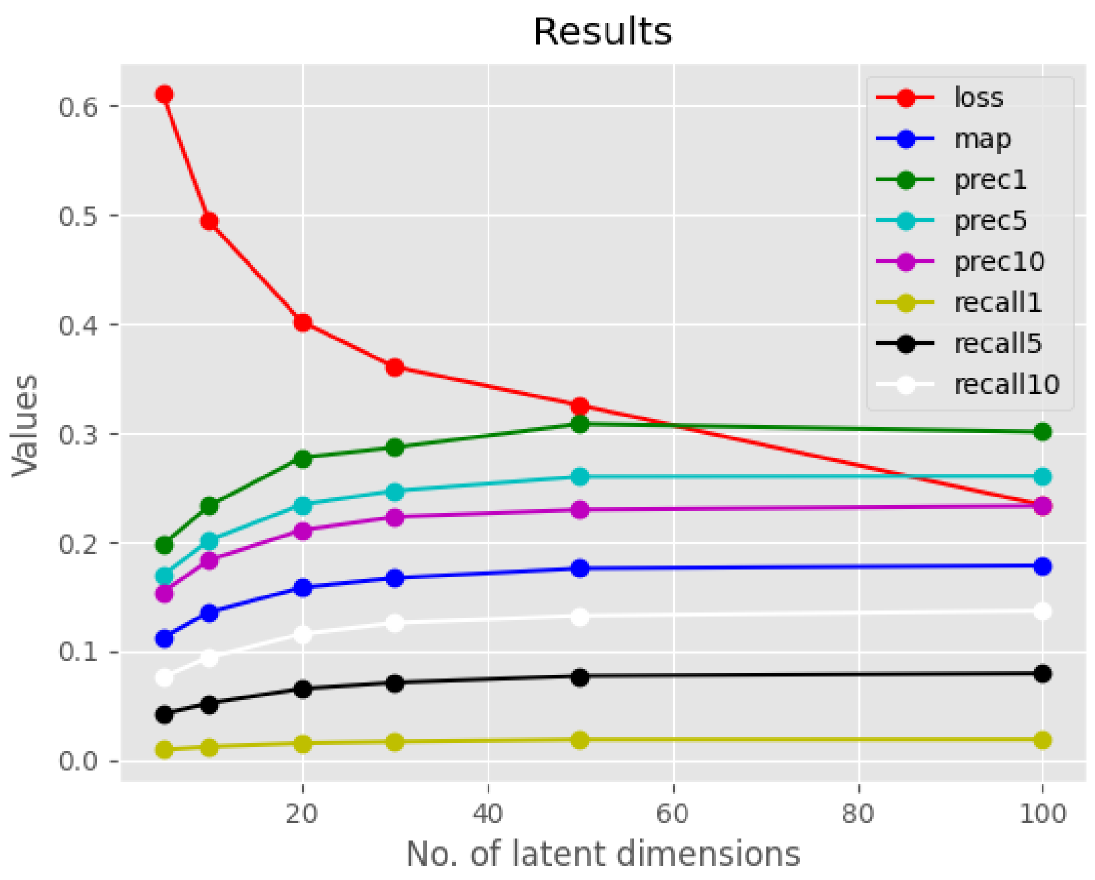


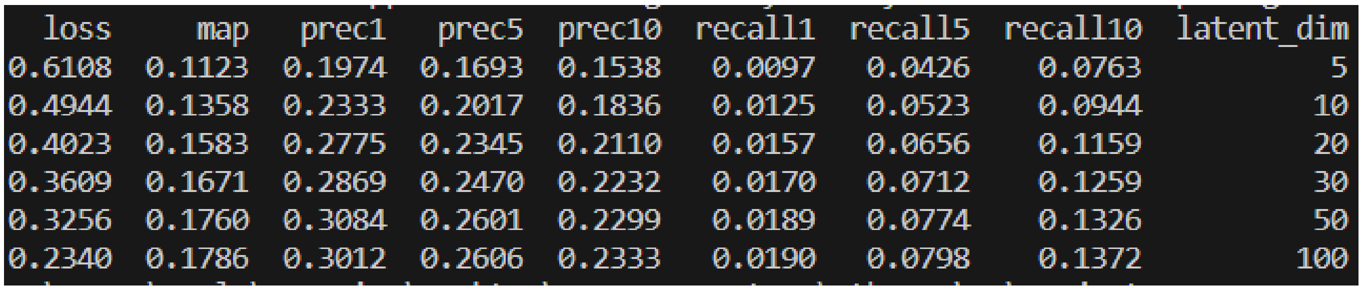
The paper proposes a personalized top-N sequential recommendation method that models each user as a sequence of items interacted in the past and aims to predict top-N ranked items that a user will likely interact in the "near future". This paper proposes a Convolutional Sequence Embedding Recommendation Model (Caser) to address this requirement. Caser embeds a sequence of recent items into an image and learns sequential patterns as local features of the image using convolutional iterations. Experiments on public data sets demonstrated that Caser consistently outperforms state-of-the-art sequential recommendation methods on a variety of common evaluation metrics.to embed sequential user-item interactions into a low-dimensional vector space.

We evaluated the proposed method on several real-world datasets and found that it replicated the author's recommendation methods in terms of accuracy and efficiency.

We further tested on other datasets, apart from paper as follows:

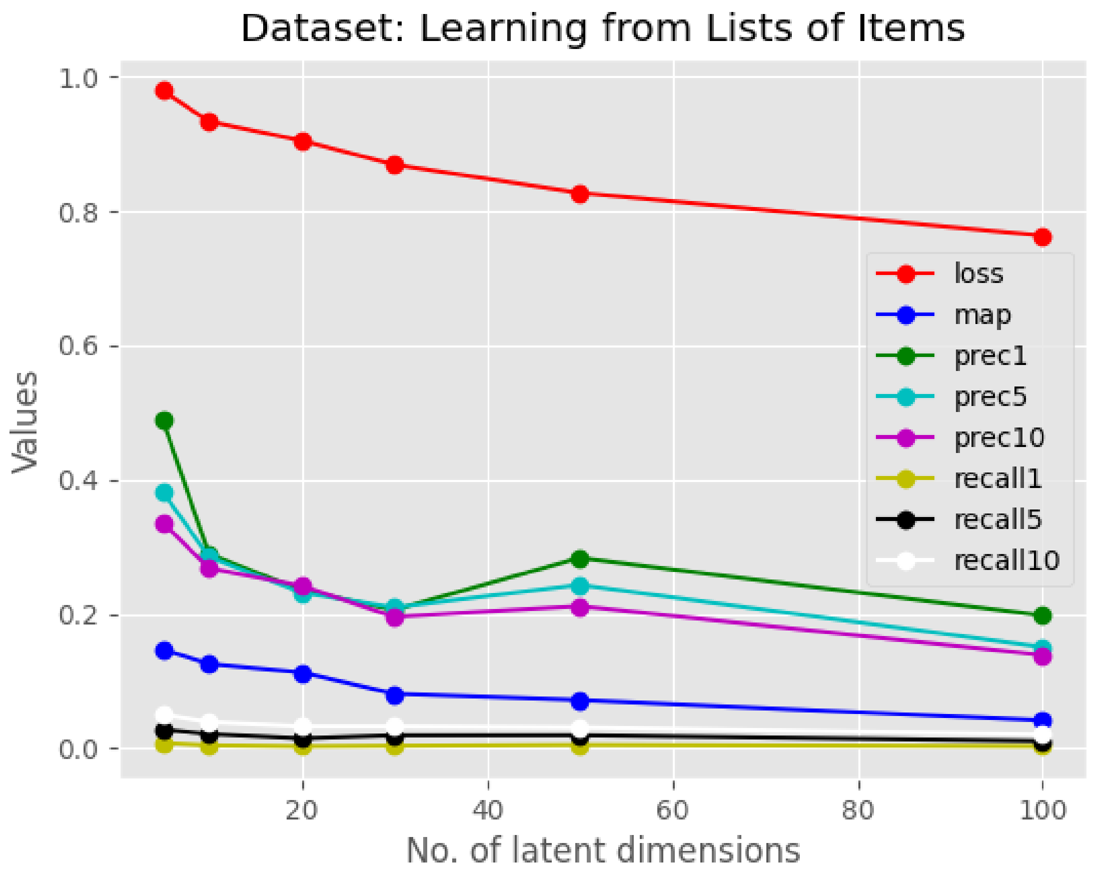
MovieLens 10M:

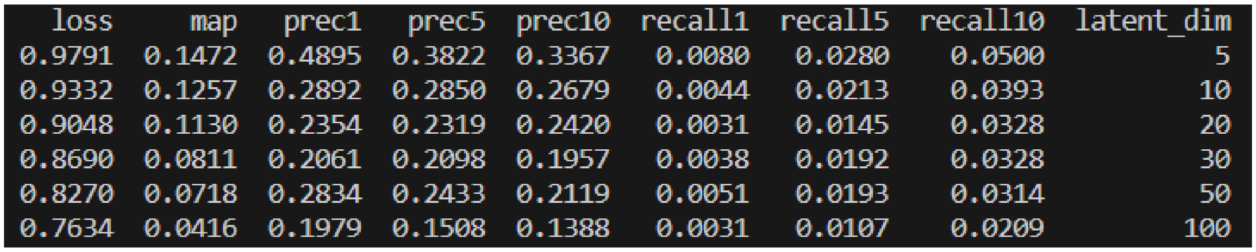




```
  loss      map    prec1    prec5   prec10  recall1  recall5  recall10  latent_dim
0.6108   0.1123   0.1974   0.1693   0.1538   0.0097   0.0426    0.0763           5
0.4944   0.1358   0.2333   0.2017   0.1836   0.0125   0.0523    0.0944          10
0.4023   0.1583   0.2775   0.2345   0.2110   0.0157   0.0656    0.1159          20
0.3609   0.1671   0.2869   0.2470   0.2232   0.0170   0.0712    0.1259          30
0.3256   0.1760   0.3084   0.2601   0.2299   0.0189   0.0774    0.1326          50
0.2340   0.1786   0.3012   0.2606   0.2333   0.0190   0.0798    0.1372         100
```

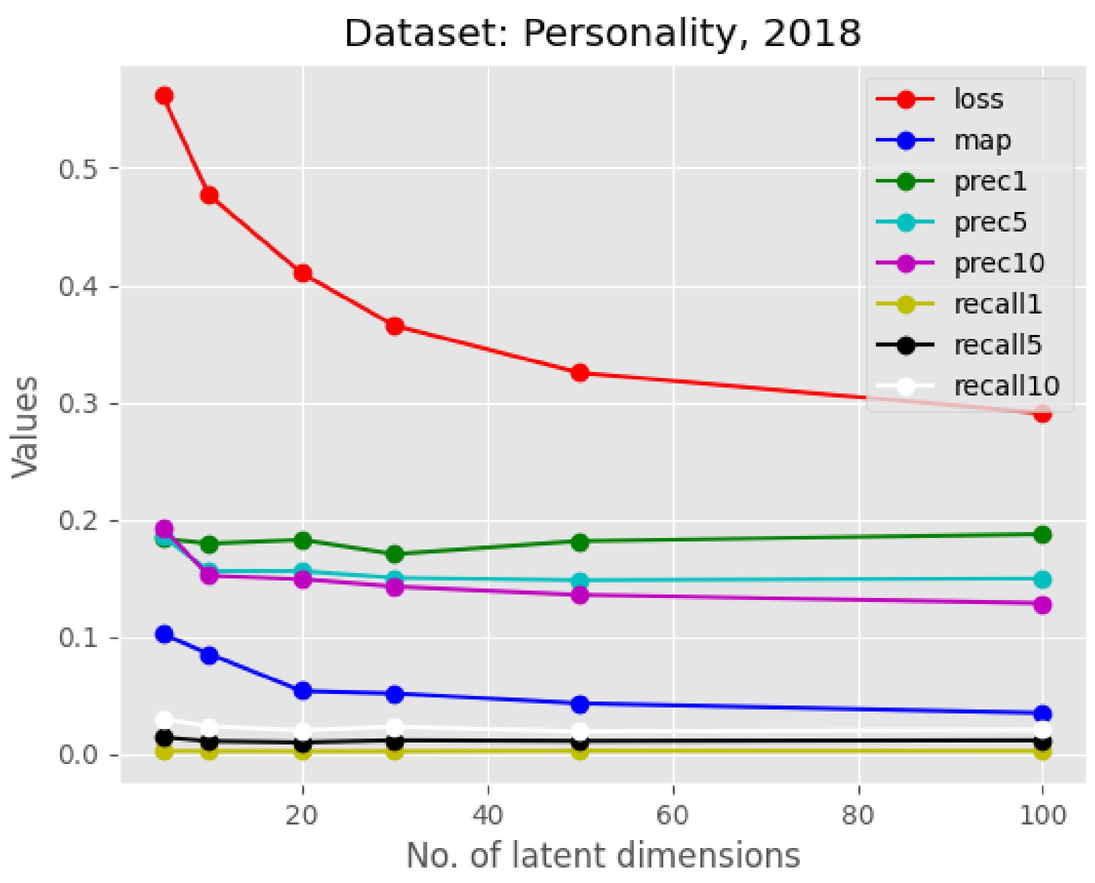
Learning from sets of items 2019:

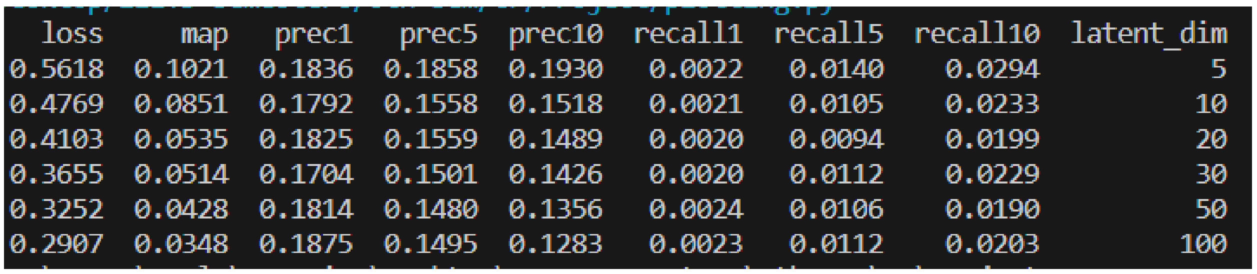




```
  loss     map    prec1   prec5  prec10  recall1  recall5  recall10  latent_dim
0.9791  0.1472  0.4895  0.3822  0.3367   0.0080   0.0280    0.0500           5
0.9332  0.1257  0.2892  0.2850  0.2679   0.0044   0.0213    0.0393          10
0.9048  0.1130  0.2354  0.2319  0.2420   0.0031   0.0145    0.0328          20
0.8690  0.0811  0.2061  0.2098  0.1957   0.0038   0.0192    0.0328          30
0.8270  0.0718  0.2834  0.2433  0.2119   0.0051   0.0193    0.0314          50
0.7634  0.0416  0.1979  0.1508  0.1388   0.0031   0.0107    0.0209         100
```

Personality 2018:





```
  loss     map    prec1   prec5   prec10  recall1  recall5  recall10  latent_dim
0.5618  0.1021  0.1836  0.1858  0.1930   0.0022   0.0140    0.0294           5
0.4769  0.0851  0.1792  0.1558  0.1518   0.0021   0.0105    0.0233          10
0.4103  0.0535  0.1825  0.1559  0.1489   0.0020   0.0094    0.0199          20
0.3655  0.0514  0.1704  0.1501  0.1426   0.0020   0.0112    0.0229          30
0.3252  0.0428  0.1814  0.1480  0.1356   0.0024   0.0106    0.0190          50
0.2907  0.0348  0.1875  0.1495  0.1283   0.0023   0.0112    0.0203         100
```

SHAN: Sequential Recommender System based on Hierarchical Attention Network

- Model for sequential predictions taking the user's long and short-term interests into account.
- Uses a hierarchical attention mechanism to construct the user's latent factor vector.
- Employs a latent-factor model for calculating preferences.

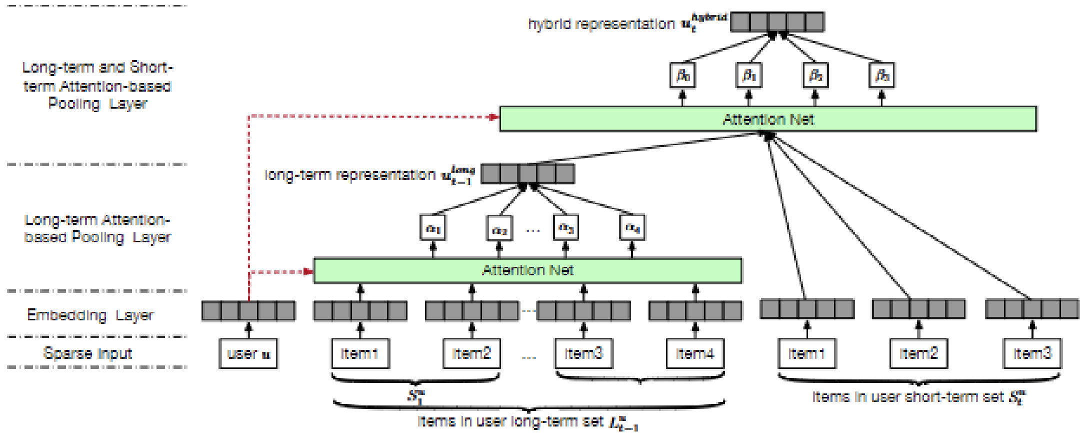

Long-term and Short-term Attention-based Pooling Layer
Long-term Attention-based Pooling Layer
Embedding Layer
Sparse Input
hybrid representation u_t^{hybrid}
β_0
β_1
β_2
β_3
Attention Net
long-term representation u_{t-1}^{long}
α_1
α_2
α_3
α_4
Attention Net
user u
item1
item2
item3
item4
item1
item2
item3
S_1^u
Items in user long-term set L_{t-1}^u
Items in user short-term set S_t^u

Figure 1: The architecture of our model. A hybrid user representation is learned by a sequential hierarchical attention network, which combines both the long- and short-term user preference.

(source: SHAN by Ying et al.)

Approach:

1. Embed user and item label-encoded representations into vectors.
2. Used an attention mechanism at the lower levels for long-term sequential data.
3. Used an independent attention mechanism at upper level for short-term sequential data.
4. Weighted combination of the two representations to capture the entire user-representation.
5. Latent factor model : take a matrix product of net user-representation with item representations to get preference of the user.

The ReLU activation function was used as the activation for the attention layers.

Loss Function:

1. The loss function optimisation involved sampling for a random item and maximising the log-sigmoid value of the difference between the target preference and the random item's preference:

$$\arg\min_{\Theta} \sum_{(u, L^u_{t-1}, S^u_t, j, k) \in D} - \ln \sigma(R_{ujt} - R_{ukt}) + \lambda_{uv} ||\Theta_{uv}||^2 + \lambda_a ||\Theta_a||^2, \quad (9)$$

   Here, R_{ujt} and R_{ukr} denote the target's preference and the random item's preference, σ is the sigmoid activation function, λ's are the regularisation parameters for the weights **Θ** for the embeddings and the attention-layers.
2. To prevent overfitting, early-stopping was done.
3.

Data preprocessing:

We dropped users and items having less than 20 appearances in the entire dataset and constructed sequences having a minimum size of 11 items.
Note: This differs from the vanilla implementation of the paper, wherein they shortlist the last 9-months' data for creating the models. This reduced the training examples and increased the number of users and items considered for evaluation.
The data splits for training, validation and testing were done created in 7:1:2 ratio selected in a random fashion.
The period of time chosen was 1 day (1440 minutes).

Results:

Gowalla dataset:

Note: the dataset used for training and validation were larger than what they used in the paper. The deviations could have happened due to the differences in sizes of the datasets and training examples.

There were 9,908 users and 41,012 items in the dataset.

The train-test-val splits are as follows:

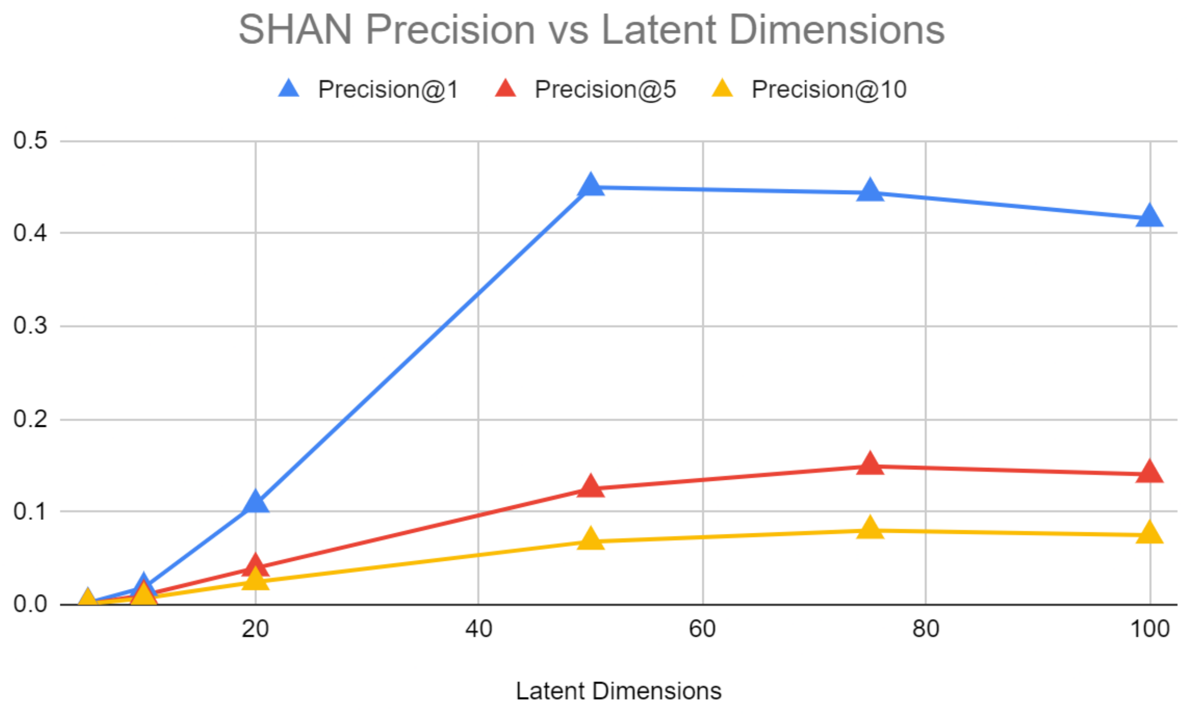
1. Train: 10,808 sequences
2. Val: 349 sequences

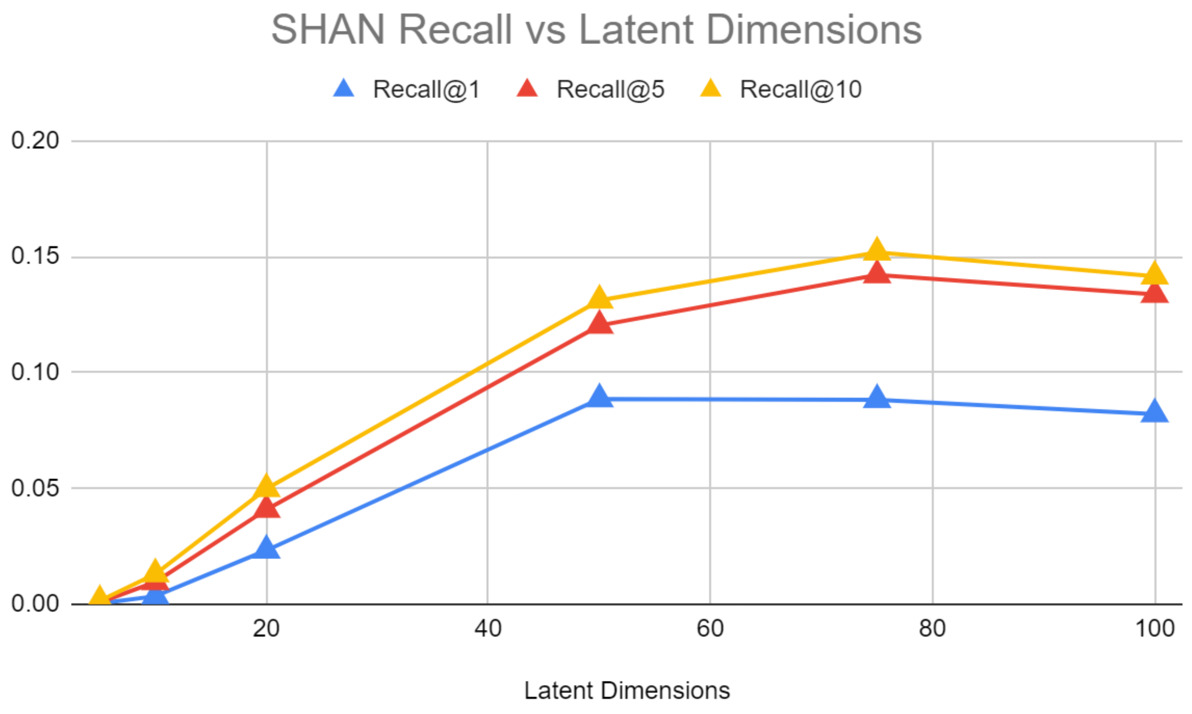
3. Test: 1,525 sequences

The splits were done randomly.

Early stopping was used to prevent overfitting and the best model was saved. The last entry of the sequence was used as a target, as per the paper.

Latent Dimensions	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
5	0.00197	0.00079	0.00092	0.00036	0.00077	0.00155
10	0.01836	0.01023	0.00682	0.00348	0.00965	0.01306
20	0.1082	0.03948	0.02459	0.02327	0.04088	0.04984
50	0.44984	0.12472	0.06807	0.08858	0.12032	0.13117
75	0.44393	0.14911	0.08	0.08826	0.14209	0.15189
100	0.41639	0.14046	0.07482	0.08212	0.13366	0.14157


SHAN Precision vs Latent Dimensions
Precision@1
Precision@5
Precision@10
0.5
0.4
0.3
0.2
0.1
0.0
20
40
60
80
100
Latent Dimensions


SHAN Recall vs Latent Dimensions
Recall@1
Recall@5
Recall@10
0.20
0.15
0.10
0.05
0.00
20
40
60
80
100
Latent Dimensions

Amazon Books dataset:

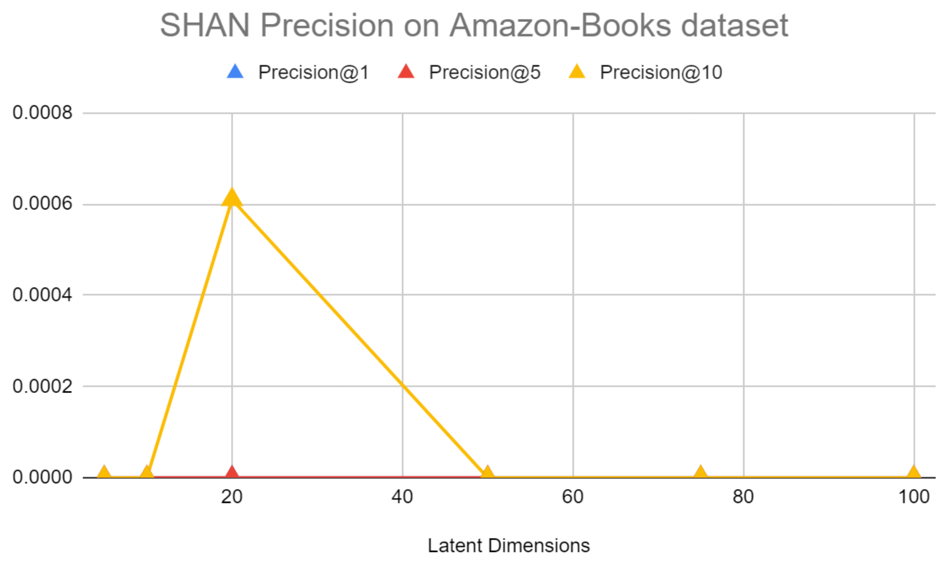
There were 6,109 users, 64,754 items.

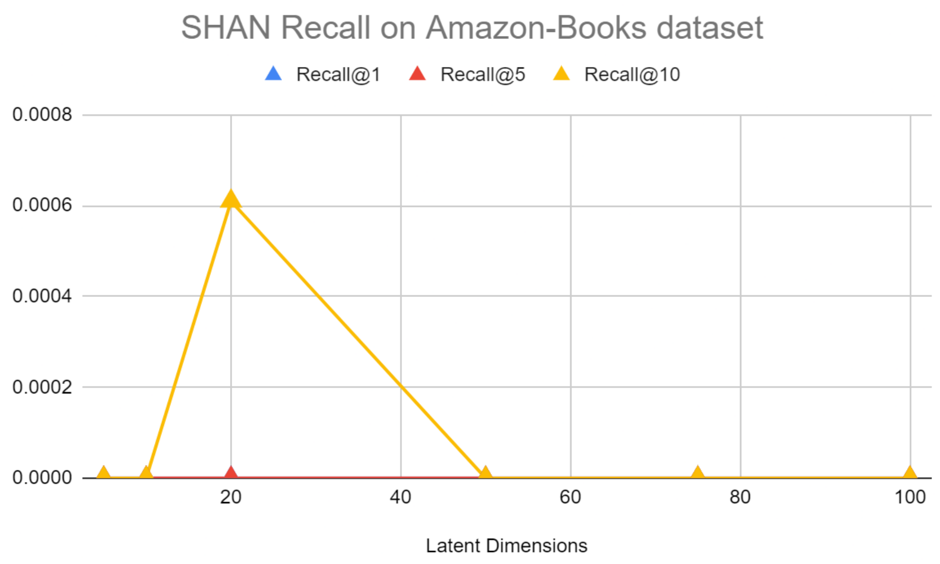
The train-test-val splits are as follows:

1. Train: 6,747 sequences
2. Val: 32 sequences
3. Test: 164 sequences

Latent Dimensions	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
5	0	0	0	0	0	0
10	0	0	0	0	0	0
20	0	0	0.00061	0	0	0.00061
50	0	0	0	0	0	0
75	0	0	0	0	0	0
100	0	0	0	0	0	0

The SHAN model was unable to produce good results on the Amazon-Books dataset. This could be a result of the less number of sequences used for training which prevented the model from learning relevant representations for prediction.





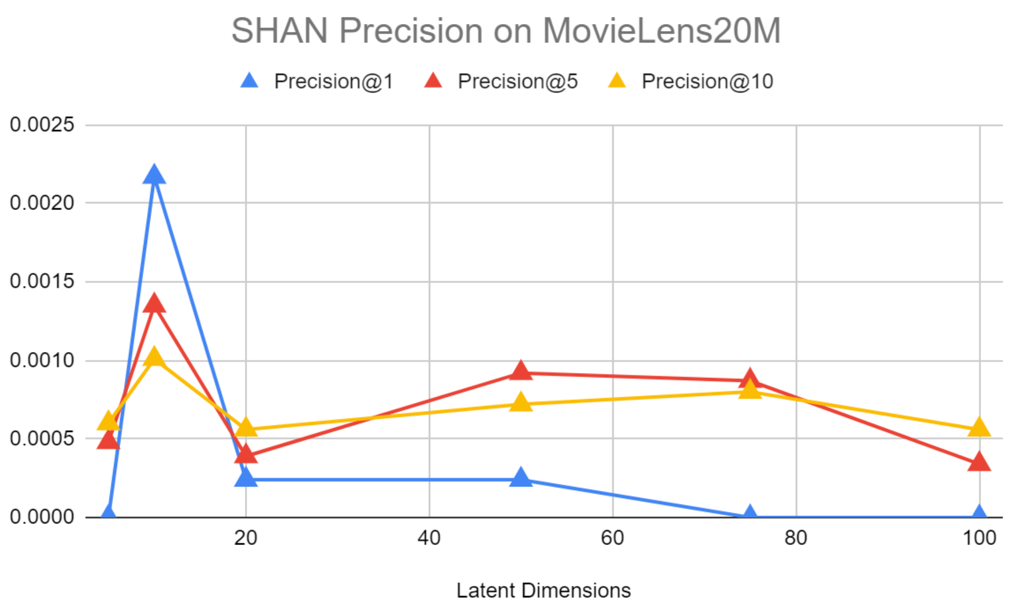
MovieLens20M dataset:

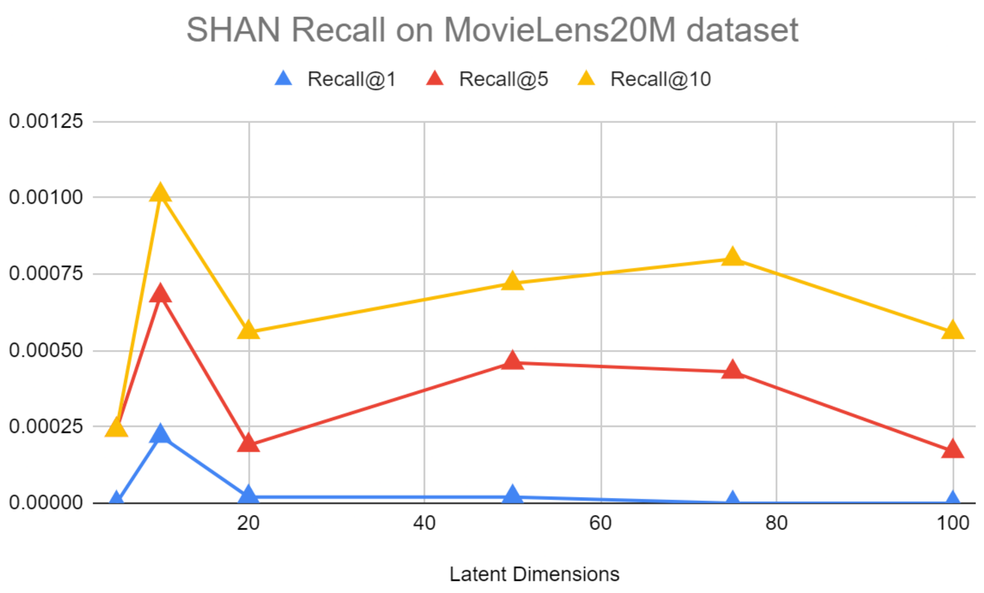
There were 101,196 users, 9,630 items.

The train-test-val splits are as follows:

1. Train: 101,700 sequences
2. Val: 351 sequences
3. Test: 4,143 sequences

Latent Dimensions	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10
5	0	0.00048	0.0006	0	0.00024	0.00024
10	0.00217	0.00135	0.00101	0.00022	0.00068	0.00101
20	0.00024	0.00039	0.00056	2.00E-05	0.00019	0.00056
50	0.00024	0.00092	0.00072	2.00E-05	0.00046	0.00072
75	0	0.00087	0.0008	0	0.00043	0.0008
100	0	0.00034	0.00056	0	0.00017	0.00056





Conclusion:

1. We observe that the SHAN model performs poorly on Amazon-Books and the Gowalla dataset.
2.