Status Report: Future Recommendation

Mari • Jun - Aug, 2023

What's been done — Week5 June 27 - July 4

1. Methodology writing

- Shared embeddings, Sampled Softmax etc.

2. Studied further for reproduction

- Log Q probability correction
- Mixed negatives
- Dense All Items Prediction
- Improved my previous model from last week for window=3

LogQ (Candidate) Probability Correction

Why needed?

- Mixed negatives improve performance [1, 2]
- In-batch negative sampling uses positives as negatives ↠ Popular items occur frequently
- To correct sampling bias
- Some do not subtract log Q from positives [2], while some do [1, 3] ↠ Mine does not

Frequency of the item over the entire training set

1. Created a frequency count table for all items
2. Divide each by the number of all items $\mathcal{N}$
3. Take a natural log

- $\mathcal{N}$ is too big in practice/NLP ↠ Approximated, e.g., using Count-Min Sketch

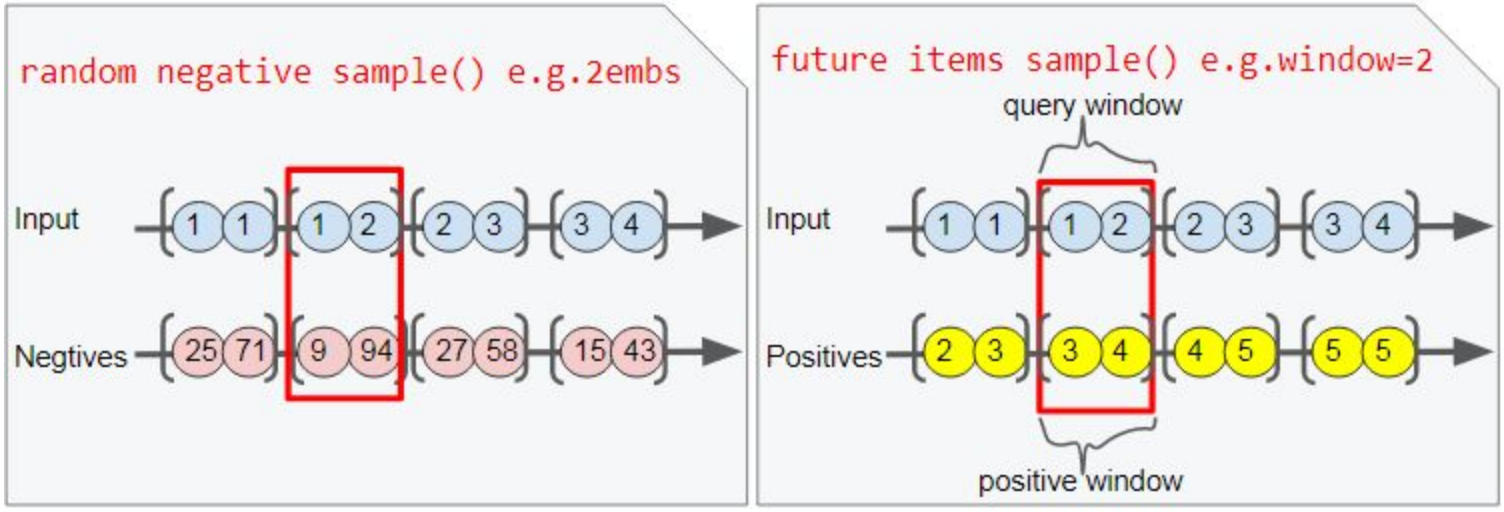
Mixed Negatives

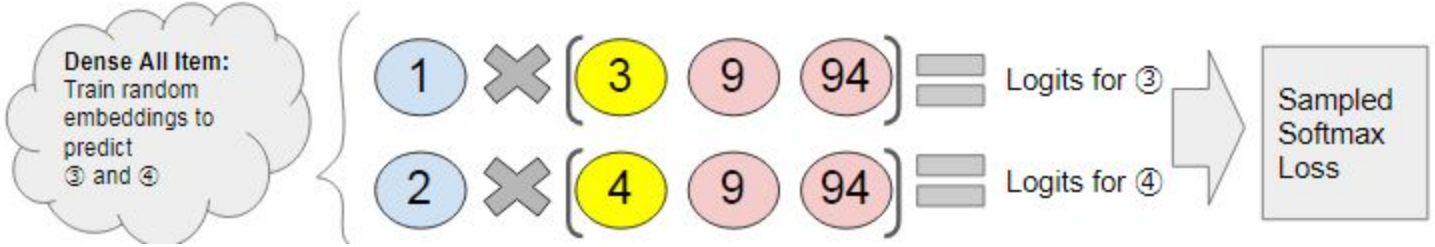
Mixing ratio depends on model and dataset

- In-batch versus: random ↠ Around 5 : 8 in [1] and 1 : 8 in [2]
- Mine (window=3, # of item to predict=3) worked best with 1 : 6 on MovieLense-1m
- It could be different by window size

Dense All Items

Introduce query-window to train multiple past embeddings





Results

Mixed negatives and Softmax with LogQ improve performance

- Need to tune mix ratio of negatives by window size

Average length: 140.12; Maximum window: 28

Model	NDCG@10_3	NDCG@10_5	NDCG@10_12	RECALL@10_3	RECALL@10_5	RECALL@10_12
Next item (BCE)	0.6067	0.5885	**0.5356**	0.8447	0.8263	0.7742
Future All items (BCE)	0.5717	0.5512	0.5033	0.8110	0.8031	0.7770
Future All items (Softmax)	0.6151	**0.5906**	0.5112	**0.8620**	**0.8436**	**0.7861**
Future All items (Softmax LogQ Mixed Neg)	**0.6268**	NA	NA	0.8548	NA	NA

Next Steps

Try to establish Dense All Items

- Also obtain the results for mixed negatives models with window size=5 and 12

The dense all action loss outperforms all action prediction on Recall@10 and global diversity. The key difference between these two losses is that in the all action loss, gradients from all positive examples for a user will be backpropagated through the same user embedding, resulting in a larger averaging effect, as compared to the dense all action loss, where gradients all pass through different transformer outputs, and are only averaged together after passing into the transformer.

Referrences

[1] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. Pinnerformer: Sequence modeling for user representation at pinterest, 2022.

[2] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In Companion Proceedings of the Web Conference.

[3] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, and Ed Chi (Eds.). 2019. Sampling-BiasCorrected Neural Modeling for Large Corpus Item Recommendations

What's been done — Week4 June 20 - 27

1. Methodology writing

- Data split design and metrics etc.

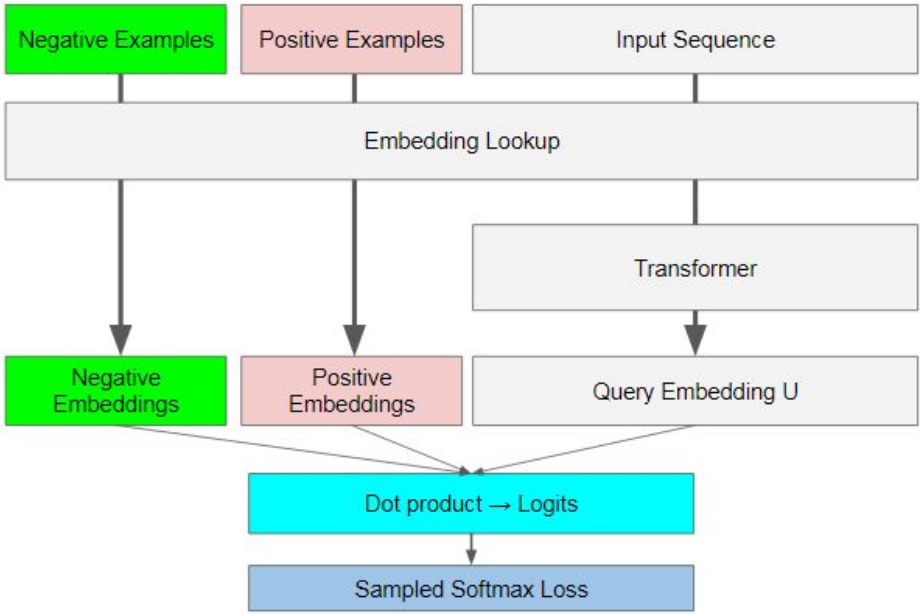


Fig 1. Network training process with the changes made

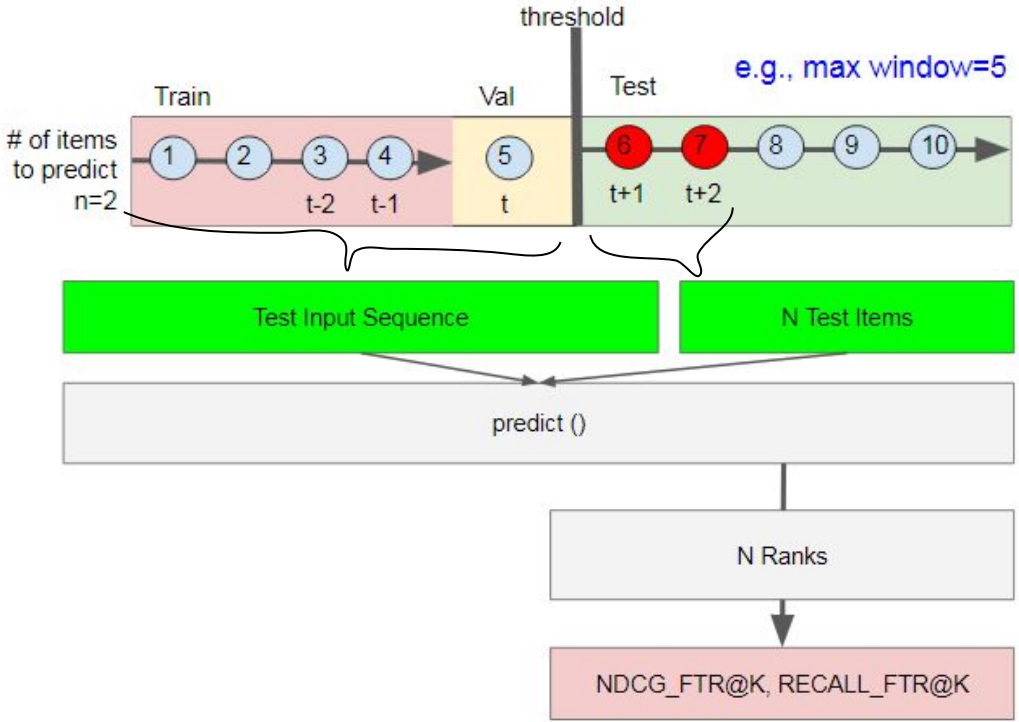
2. Studied Embedding Learning

- Improved Future Items Predictor
  - Data split
  - Future positive sampling
  - Multiple random negative sampling
  - Shared positive & negative embeddings
  - Sampled softmax loss
- Partially reproduced PINNERFORMER results in All Items Prediction on MovieLense-1M

Data Split Design and Metrics

1. Use Leave N out split

- Val 1 : Test N (ignore val as it does nothing)



2. Use NDCG@K and RECALL@K

$$\mathrm{NDCG@K(U)} = \frac{1}{n}\sum_{i=t+1}^{n} \mathrm{NDCG@K}_i$$

$$\mathrm{NDCG@K}_{FTR} = \frac{1}{|\mathcal{U}|}\sum_{U\in\mathcal{U}} \mathrm{NDCG@K(U)} \quad (1)$$

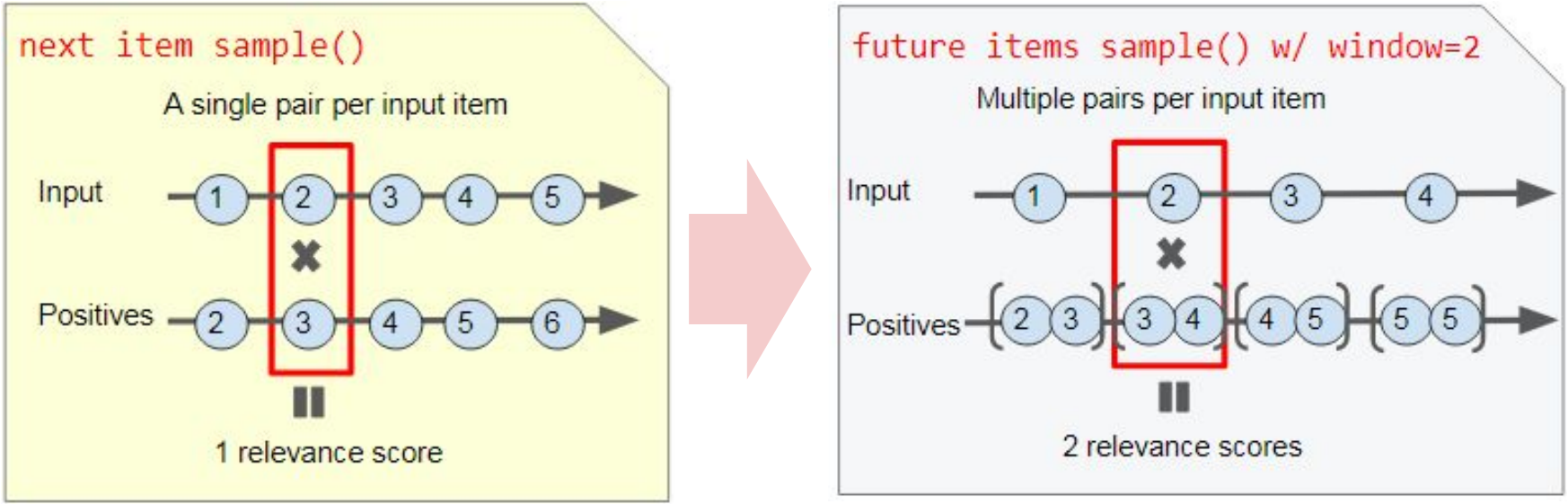
$$\mathrm{RECALL@K(U)} = \frac{1}{n}\sum_{i=t+1}^{n} 1\{rank_U(item) \le K\}$$

$$\mathrm{RECALL@K}_{FTR} = \frac{1}{|\mathcal{U}|}\sum_{U\in\mathcal{U}} \mathrm{RECALL@K(U)} \quad (2)$$

Future Positive Pairs Selection

Select multiple (query, item) pairs

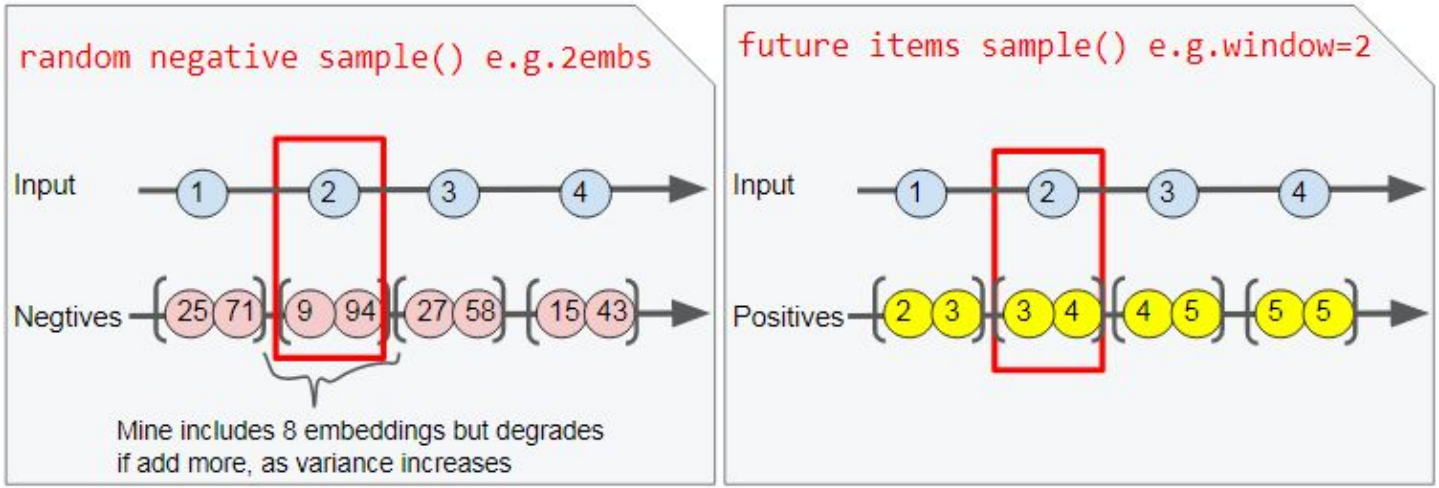
- Select multiple positive items for each input item

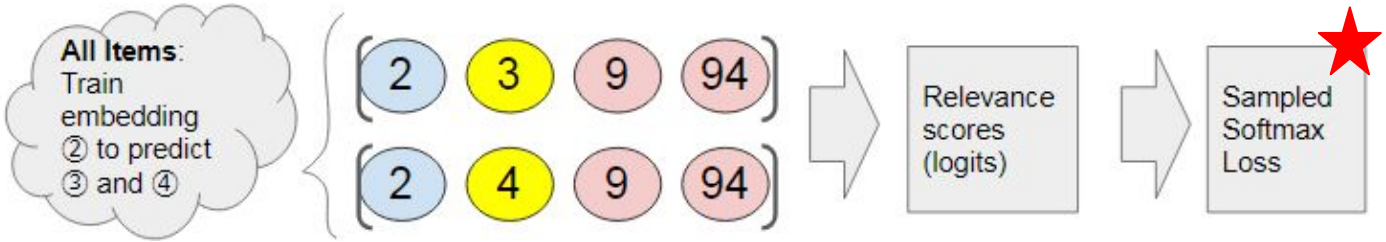


Shared Positive & Negative Embeddings

$$\mathcal{L}(u_i, p_i) = -\log\left(\frac{e^{s(u_i,p_i)}}{e^{s(u_i,p_i)} + \sum_{j=1}^{N} e^{s(u_i,n_j)}}\right)$$

Learn embeddings as a multi-class classification problem

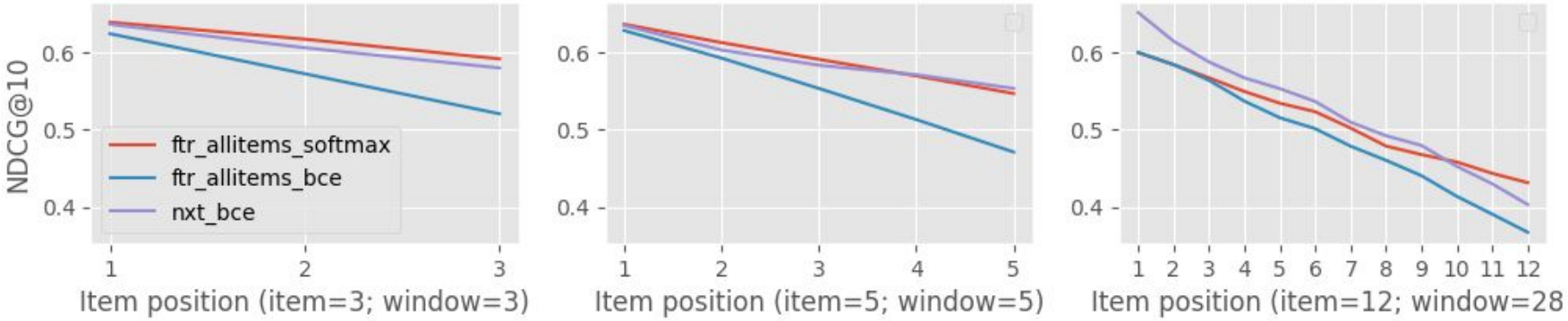




Results

Longer window improves performance

- Shorter window is enough for a few items ↛ Predict more items need a longer window
- Mixed negatives (in-batch & random) with logQ is an option to shorten the window



Average length: 140.12; Maximum window: 28

Model	NDCG@10_3	NDCG@10_5	NDCG@10_12	RECALL@10_3	RECALL@10_5	RECALL@10_12
Next item (BCE)	0.6067	0.5885	**0.5356**	0.8447	0.8263	0.7742
Future All items (BCE)	0.5717	0.5512	0.5033	0.8110	0.8031	0.7770
Future All items (Softmax)	**0.6151**	**0.5906**	0.5112	**0.8620**	**0.8436**	**0.7861**

Next Steps

Improve my future items predictor

- Try tuning on different datasets - Amazon Beauty (sparse) etc.
- Figure out computation of logQ correction term to mix in-batch negatives
- Think about Dense All Items prediction in my approach

What's been done — Week3 June 15 - 22

1. Back ground reading

- Introduction and Related work sections almost completed

2. Experiments

- Tried different data split
- Experimented with Next item, All items, Dense all items predictions on MovieLense-1M
- Tried 2 approaches
  1. Data split only (next item predictor)
  2. Data split + Positove sampling (future items predictor)

Data split

Use val 1 : test N split

- e.g., train [1,2,3,4,5], val [6], test [7,8,9]
- val 1 : test N works better than val N : test N
- Validation size has no effect on network training because training happens before evaluate_val() and evaluate()
- Thus, longer validation window just makes training sequence length shorter (not good)

Positive Sampling

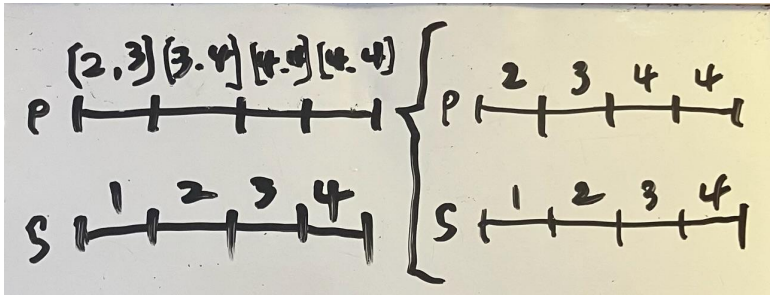
```
def sample():

    user = np.random.randint(1, usernum + 1)
    while len(user_train[user]) <= 1: user = np.random.randint(1, usernum + 1)

    seq = np.zeros([maxlen], dtype=np.int32)
    pos = np.zeros([maxlen], dtype=np.int32)
    neg = np.zeros([maxlen], dtype=np.int32)
    nxt = user_train[user][-1]
    idx = maxlen - 1

    ts = set(user_train[user])
    for i in reversed(user_train[user][:-1]):
        seq[idx] = i
        pos[idx] = nxt
        if nxt != 0: neg[idx] = random_neq(1, itemnum + 1, ts)
        nxt = i
        idx -= 1
        if idx == -1: break

    return (user, seq, pos, neg)
```

E. Network Training

Recall that we convert each user sequence (excluding the last action) $(\mathcal{S}_1^u, \mathcal{S}_2^u, \ldots, \mathcal{S}_{|\mathcal{S}^u|-1}^u)$ to a fixed length sequence $s = \{s_1, s_2, \ldots, s_n\}$ via truncation or padding items. We define o_t as the expected output at time step t:

$$o_t = \begin{cases} \texttt{<pad>} & \text{if } s_t \text{ is a padding item} \\ s_{t+1} & 1 \le t < n \\ \mathcal{S}_{|\mathcal{S}^u|}^u & t = n \end{cases},$$

where <pad> indicates a padding item. Our model takes a sequence s as input, the corresponding sequence o as expected



Future items predictor (window size =2)

Next item predictor (default SASRec)

Results

Window size effect

- Next-item performance improves as window size increases, peaking at 12
- Performance degrades as the item gets farther from the e1 embedding
- Shorter the window size, steeper the performance drop
- Aim to uplift those lines

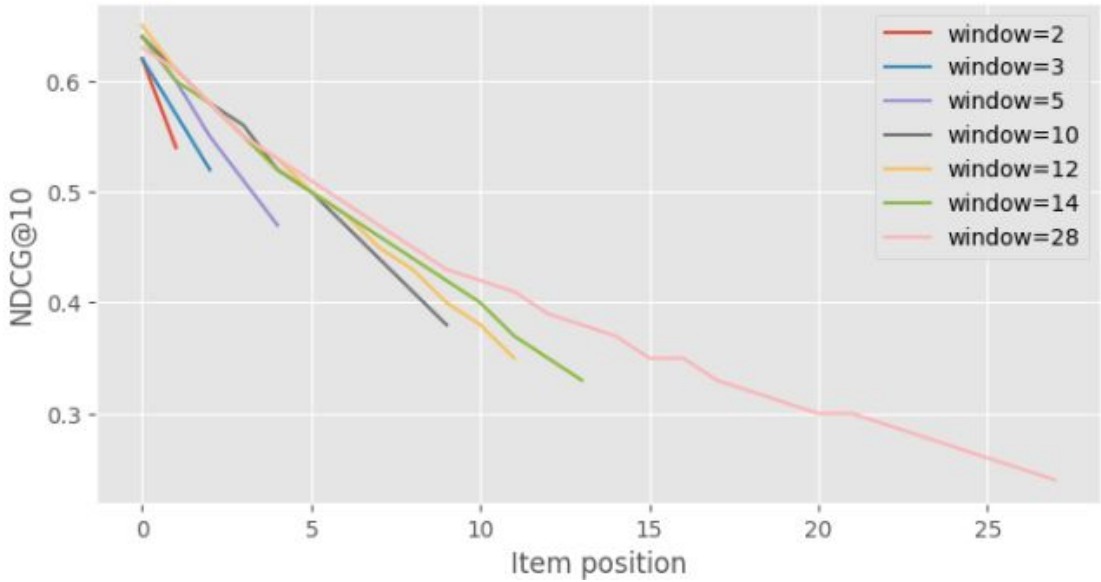


Fig 1. Effect of window size on All items prediction for each item position

Results

Window size=3

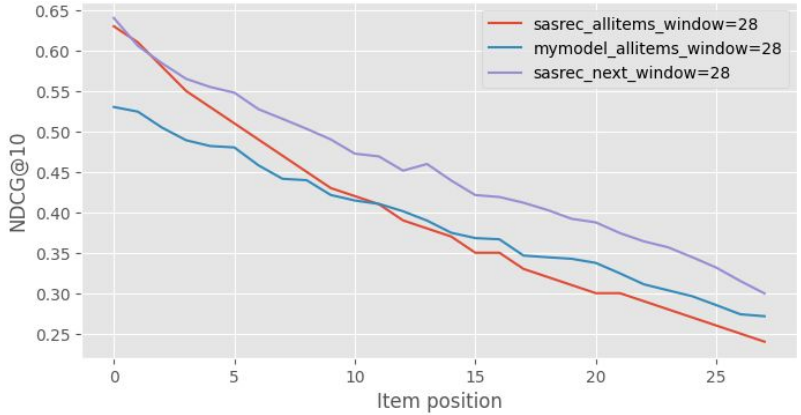
- My model is slightly better than SASRec

Training objective (window=3)	NDCG@10_ftr	HIT@10_ftr
Dense all (SASRec+DataSplit)	0.5727	0.8108
All items (SASRec+DataSplit+PosSamp)	0.5717	0.8110
Next item (SASRec)	0.5709	0.8045
All items (SASRec+DataSplit)	0.5703	0.8080

Window size=12

- SASRec is the best
- But my model performs better on further items - will think about to improve closer items as well

Training objective (window=12)	NDCG@10_ftr	HIT@10_ftr
Next item (SASRec)	0.5227	0.7574
All items (SASRec+DataSplit)	0.4983	0.7571
Dense all (SASRec+DataSplit)	0.4977	0.7581
All items (SASRec+DataSplit+PosSamp)	0.4874	0.7581



Next Steps

1. Try to improve closer items prediction for my model?

2. Think about future items predictor for Dense all items

- embedding layer needs to be changed

3. Start working on Methodology section

What's been done - Week1 June 1 - 8

1. Back ground reading

- Read related papers from TiSASRec, SASRec, PinnerFormer etc.
- Working on Introduction and Related Work sections

2. Experiment Preparation

- Reproduced SASRec and TiSASRec with Pytorch
- Codebase exploration
- Approached All Action Predictions by predicting [item+1, item+2] with e1 fixed - Can we do it with just changing sampling method or need to change model structure (e,g. output a list)?

Next Steps

1. Background reading

Continue reading and working on Related Work section

2. Code study

Approaches of future item prediction in mind