Self-attentive Sequential Recommendation

Sung Joon Park, Michael Chang, Henry Carscadden

Problem Background

- Goal of any sequential recommendation system: capture the ‘context’ of users’ activities on the basis of actions they have performed recently
  - Challenge: How to capture useful patterns from a user’s action history?
- Traditional methods:
  - Markov Chains (MCs)
  - Recurrent Neural Networks (RNNs)
- Inspiration: *Transformer*
  - A new sequential model for machine translation tasks
  - Introduced a proposed attention mechanism called ‘self-attention’

Self-attentive Sequential Recommendation Model (SASRec)

- Application of the self-attention mechanism to sequential recommendation systems
- Goal: Fix the problems found in Markov Chains and Recurrent Neural Networks
    - SASRec can adapt depending on the dataset sparsity
- Self-attention mechanism is suitable for parallel acceleration
    - Allows SASRec model to be faster than CNN/RNN-based alternatives

Related Work

General Recommendations

- Matrix Factorization (ex: NeuMF)
- Item Similarity Models (ISM)

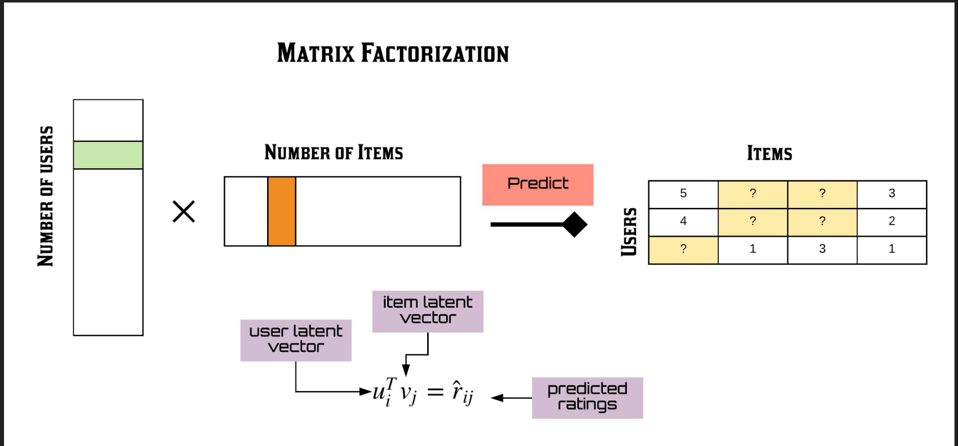
Temporal Recommendation

- Timestamps on users' activities

Sequential Recommendation

- FPMC (MF + item-item transitions)
- RNNs (Ex: GRU4Rec)

Attention Mechanisms



Dataset

4 datasets

- Amazon (Beauty, Games): sparse
- Steam: dense
- MovieLens: very dense

Data Usage:

- Usage of review/rating & timestamp
- Data Partitioning

Table II: Dataset statistics (after preprocessing)

Dataset	#users	#items	avg. actions /user	avg. actions /item	#actions
Amazon Beauty	52,024	57,289	7.6	6.9	0.4M
Amazon Games	31,013	23,715	9.3	12.1	0.3M
Steam	334,730	13,047	11.0	282.5	3.7M
MovieLens-1M	6,040	3,416	163.5	289.1	1.0M

Comparison Methods

General Recommendations

- PopRec
- Bayesian Personalized Ranking (BPR)

Sequential Recommendation (1st order: only last visited item)

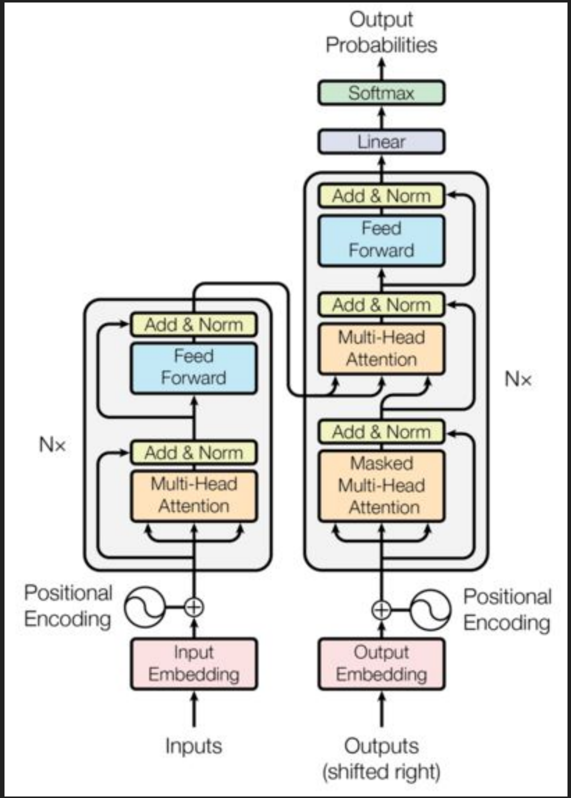
- Factorized Markov Chains (FMC)
- Factorized Personalized Markov Chains (FMC)
- Translation-based Recommendation (TransRec)

Deep-learning based sequential recommender systems (multiple previous items)

- GRU4Rec / GRU4Rec+
- Convolutional Sequence Embeddings

Overview of Model

1. Passes inputs through embedding layer
    a. Item and position embedding added
2. *b* blocks applied
    a. Embeddings passed to attention layer
    b. Output is processed
        i. Layer Normalization applied
        ii. Dropout applied
        iii. Residual connections applied
    c. Two feed-forward layers applied
3. Outputs decoded and highest probability item chosen



Input Embedding

- Embedding finds latent features describes inputs
  - Learns lower-dimensional representation that reproduces original input well
- Input embedding matrix (E) is the sum of two embeddings
  - Item embedding encodes item similarity
  - Positional embedding encodes the sequential nature of user's interactions
- Row *i* embeds item *i* to capture its relation to other items and its position

$$\widehat{\mathbf{E}} = \begin{bmatrix} \mathbf{M}_{s_1} + \mathbf{P}_1 \\ \mathbf{M}_{s_2} + \mathbf{P}_2 \\ \cdots \\ \mathbf{M}_{s_n} + \mathbf{P}_n \end{bmatrix}$$

SA (Self-Attention) Mechanism

- Three learned projections (W^K, W^Q, W^V) map input matrices to K (keys), Q (queries), and V (values)
- Each entry in the output of a SA layer is the dot product of a query vector and key vector scaled by a value vector
  - Recall for vectors a, b, a·b represents similarity of vectors
  - Each entry represents interaction between a key and query
  - Interactions forbidden between vector K_i and Q_j where j > i
    - Allowing such interactions outside of time horizon

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \tag{2}$$

$$\mathbf{S} = \text{SA}(\widehat{\mathbf{E}}) = \text{Attention}(\widehat{\mathbf{E}}\mathbf{W}^Q, \widehat{\mathbf{E}}\mathbf{W}^K, \widehat{\mathbf{E}}\mathbf{W}^V), \tag{3}$$

Intermediate Layers

- SA is passed to a two feedforward layers
    - Latent dimensionality *d* is maintained
    - Adds non-linearity to the network architecture
- Layer connection and regularization
    - Residual Connections
        - Passes connections between layers
            - Allows different levels of structure to interact
    - Dropout
        - Neurons randomly have activation set to 0
            - Reduces overfitting in deep networks
    - Normalization
        - Ensures layer output is Gaussian (0, 1)
            - Stabilizes training

$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}(\mathbf{S}_i \mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)},$$

Results

- 3 groups of recommendation baselines used:
  a. General recommendation methods
     - Pop Rec
     - Bayesian Personalized Ranking (BPR)
  b. Sequential recommendation methods
     - Factorized Markov Chains (FMC)
     - Factorized Personalized Markov Chains (FPMC)
     - Translation-based Recommendation (TransRec)
  c. Deep-learning based sequential recommendation systems
     - GRU4Rec
     - GRU4Rec^{+}
     - Convolutional Sequence Embeddings (Caser)
- 2 evaluation metrics:
  a. Hit Rate@10: counts the fraction of times that the ground-truth next item is among the top 10 items
  b. NDCG@10: a position-aware metric which assigns larger weights on higher positions

Results cont.

Table III: Recommendation performance. The best performing method in each row is boldfaced, and the second best method in each row is underlined. Improvements over non-neural and neural approaches are shown in the last two columns respectively.

Dataset	Metric	(a) PopRec	(b) BPR	(c) FMC	(d) FPMC	(e) TransRec	(f) GRU4Rec	(g) GRU4Rec^{+}	(h) Caser	(i) SASRec	Improvement vs. (a)-(e)	Improvement vs. (f)-(h)
Beauty	Hit@10	0.4003	0.3775	0.3771	0.4310	<u>0.4607</u>	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	<u>0.3020</u>	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
Games	Hit@10	0.4724	0.4853	0.6358	0.6802	<u>0.6838</u>	0.2938	0.6599	0.5282	**0.7410**	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	<u>0.4759</u>	0.3214	**0.5360**	14.5%	12.6%
Steam	Hit@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	<u>0.8018</u>	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	<u>0.5595</u>	0.5381	**0.6306**	21.4%	12.7%
ML-1M	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	<u>0.7886</u>	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	<u>0.5538</u>	**0.5905**	14.1%	6.6%

Training Efficiency & Scalability

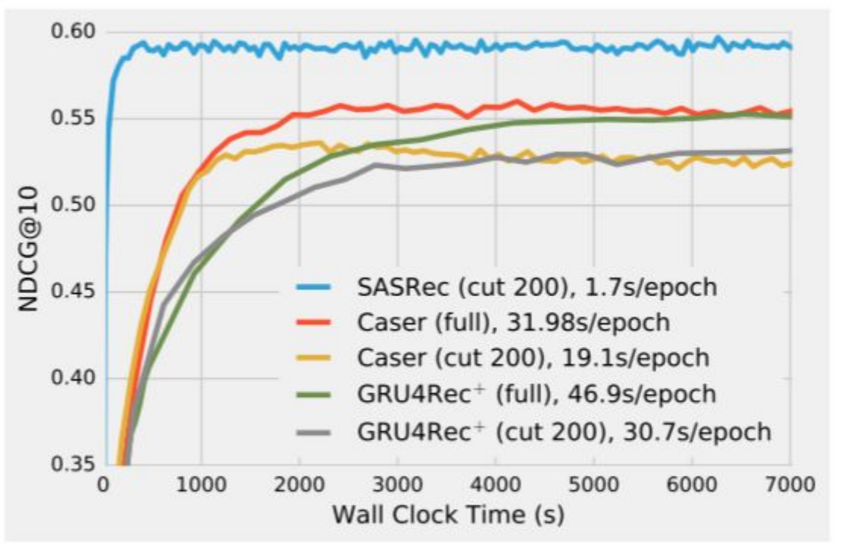


Table V: Scalability: performance and training time with different maximum length n on *ML-1M*.

n	10	50	100	200	300	400	500	600
Time(s)	75	101	157	341	613	965	1406	1895
NDCG@10	0.480	0.557	0.571	0.587	0.593	0.594	0.596	0.595

Future Work

- SASRec model was able to adaptively consider items for prediction
- Empirical results on sparse and dense datasets show that SASRec can outperform many baselines
- Future work:
  - Extending the model using rich context information
    - E.g. dwell time, action types, locations, devices, etc.
  - Handle much longer sequences
    - E.g. clicks

References

- Kang, W., & McAuley, J. (2018, August 20). Self-Attentive sequential Recommendation. Retrieved March 20, 2021, from https://arxiv.org/abs/1808.09781
- Le, J. (2020, June 28). Recommendation system series Part 4: The 7 variants of MF for collaborative filtering. Retrieved March 25, 2021, from https://towardsdatascience.com/recsys-series-part-4-the-7-variants-of-matrix-factorization-for-collaborative-filtering-368754e4fab5