xLSTM architecture for recommendation

Vivekanand Ramakrishnan (Working Version)

Abstract

At its core, a recommendation engine uses computer algorithms to predict and suggest new items of interest to users based on their past user interactions, buying behaviors and contextual data. On the deep learning front, xLSTM (extended LSTM) and Transformers have been eveolved with latest architectures and plays a very important role in Large Language Models. Since recommenders have became an essential part of our daily digital experiences, in this paper we will be leveraging XLSTM architecture based recommenders and will be comparing the results with other modern architectures like Autotransformers, Recurring Neural Networks (RNN) and other Matrix Factorization Methods. xLSTM incorporates architectural enhancements like attention mechanism, gating improvements and bidirectional capabilities. It will be impactful due to several unique aspects when to compared to the tradional successful methods.

1 Existing Architecture For Recommenders

1.1 Transformers Based:

Uses the self-attention mechanism from transformer architectures (like in GPT or BERT) to model the user-item interactions. It also captures the complex sequential patterns in user behavior (e.g., purchase history or clicks) and finally makes personalized recommendations by understanding the contextual relationships between items and users. Unlike, RNN which process information sequencially (one step at a time), transformer process information in parallel utilizing the self attentio mechanism which results in faster computation precerving long term dependencies.

A transformer-based recommender system uses an embedding layer to convert the users and items into dense vector representations. It applies self-attention layers to capture complex dependencies and sequential relationships in user-item interactions, enhanced by positional encodings to preserve and keep the order of actions. The final output layer computes scores or rankings to predict the most relevant items for each user.

1.2 Matrix Factorization Based:

It decompose a user-item interaction matrix into two smaller matrices: one representing users and the other representing items. These matrices capture latent factors (hidden patterns) that explain the user preferences and item characteristics. By reconstructing the original matrix, the system predicts how much a user might like an unseen item. Some techniques include SIngular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF) and Probabilistic Matrix Factorization (PMF).

1.3 Other Approaches:

Below are the commonly classfied two major types:

1. Collaborative Filtering (User to User), and

2. Content Based Filtering (Product to Product).

Preprint. Under review.

Table 1: List of Data Sources

Part		
Name	Description	Size (GB)
Amazon Books	Long Term Interest Modelling	1.4 GB
Spotify Million Playlist	Playlist continuation and music discovery	1.3 GB
Movie Lenz 20M	Movie Recommendation with temporal patterns	146 MB

1.4 Other Existing Simple Methods to Identify User similarities:

Using 1. Correlations, 2. Cosine Similarities, 3. Jaccard Similarities, 4. Euclidean Distance, 5. Hamming Distance, 6. Manhatten Distance, 7. Bhattachryya Distance, 8. Neural Network Embeddings, 9. Kullback Leibler divergence, 10. Embeddings and Latent Features, 11. Sequence-Based Similarity, 12. Deep Collaborative Filtering with Embeddings (via Neural Networks), 13. Transformer Models for Sequential Recommendations (e.g. BERT4Rec), 14. In life science application (some techniques used for DNA Sequencing to find the similarities, which could also be leveraged for RS), and 15. Other Hybrid Approches.

2 Proposed xLSTM Based Approach Recommender

XLSTM Architecture: Architectural enhancements like attention mechanism, gating improvements and bidirectional capabilities, can accelerate the recommender performance even in sequence processing recommendation tasks.

2.1 Data Sources

Below are the list of data sources which we will be using to leverage various models with respect to xLSTM (Refer Table 1 Above). Datasets collected from Recbole (https://recbole.io) and other libraries.

2.2 List of Models and its configurations

BPR Model Configurations:

```
config_dict = {
    'model': 'BPR',
    'dataset': 'ml-100k',
    'data_path': 'dataset/',
    'epochs': 10,
    'train_batch_size': 512,
    'eval_batch_size': 512,
    'train_neg_sample_args': {'uniform': 1},
    'learning_rate': 0.001,
    'topk': [5, 10],
    'metrics': ['Recall', 'Precision', 'NDCG'],
    'valid_metric': 'recall@10',
    'log_level': 'none',
    'checkpoint_dir': './saved_models/'
}
```

Various other models including xLSTM to be added.

2.3 Evaluation Metrics:

To evaluate the model accuracy Recall 5, 10, Precision, Normalized Discounted Combined Gain (NDCG) will be used mainly.

1. Recall: How many relevant items recommended/Total No. of relevant items available

```
Interpretation: The model was able to retrieve recall@10 of the relevant items in its top 10 recommendations for each user in the test set.
Start time: 2024-11-04 02:28:16
End time: 2024-11-04 02:30:09
Total run time: 113.30 seconds
```

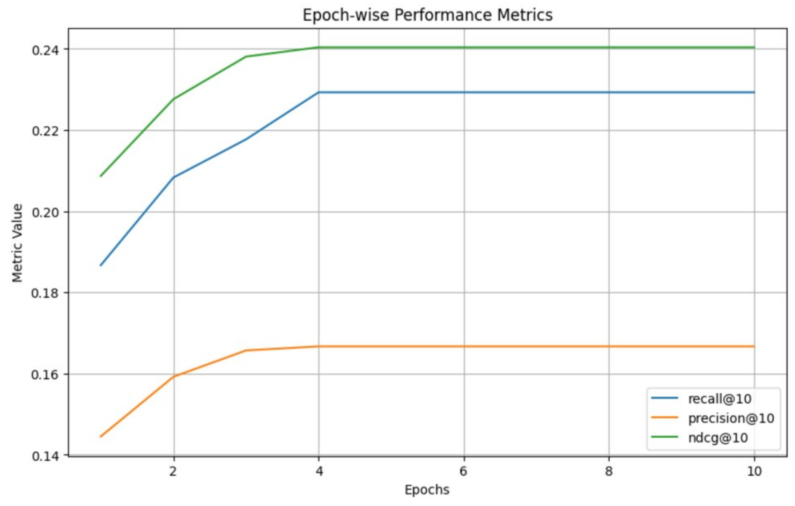

Figure 1: Basic Results from BPR Model

It measures the relevance.

2. Precision: How many relevant items recommended//Total No. of items recommended

It measures the accuracy.

3. Normalized Discounted Combined Gain (NDGC): For Ranking/Ordering.

1st set of output from BPR Model:

3 Conclusion:

Various recommender algorithm has been tested with noval approach with xLSTM along with other hybrid models. Performance has been studied and accuracy has been significantly improved. The codes and datasets used in this project has been documented here in this github (https://github.com/Vivekanandr/xLSTM-Recommender) page. Finally, conclude (to be added) that the xLSTM core benefits like attention mechanism and gating improvements helps to improve the accuracy further. In future, various xLSTM models can be leveraged further for various industry use cases and emerging applications.

References

[1] xLSTM: Extended Long Short-Term Memory: https://arxiv.org/pdf/2405.04517

[2] xLSTM-Mixer: Multivariate Time Series Forecasting by Mixing via Scalar Memories: https://doi.org/10.48550/arXiv.2410.16928

[3] Amazon Science: https://github.com/amazon-science

[4] xLSTM Time : Long-term Time Series Forecasting With xLSTM: https://doi.org/10.48550/arXiv.2407.10240

[5] Quaternion Transformer4Rec: Quaternion numbers-based Transformer for recommendation: https://github.com/vanzytay/QuaternionTransformers

[6] Recommender Systems: A Primer: https://doi.org/10.48550/arXiv.2302.02579

[7] Exploring the Impact of Large Language Models on Recommender Systems: An Extensive Review: https://arxiv.org/pdf/2402.18590

[8] Recommender Systems with Generative Retrieval: https://openreview.net/pdf?id=BJ0fQUU32w

[9] Attention Is All You Need: https://arxiv.org/abs/1706.03762

[10] Recbole: https://recbole.io

[11] Group Lens: https://grouplens.org/datasets/movielens/100k/

[12] OpenAI: https://openai.com/

[13] Hugging Face: https://huggingface.co/docs/hub/en/models-the-hub

[14] Kreutz, C.K., Schenkel, R. Scientific paper recommendation systems: a literature review of recent publications. Int J Digit Libr 23, 335–369 (2022). https://doi.org/10.1007/s00799-022-00339-w

[15] Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities https://doi.org/10.3390/app10217748

[16] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5

[17] A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice https://doi.org/10.48550/arXiv.2407.13699