Transformer Based Session Recommendation System

Arnav Chakravarthy, Ajay Gunasekaran, Apeksha Rajkumari, Soumitro Choudhury,
Hussain Khuzem Lal, Nida Zeeshan Naveedur Rahman
CSE573, Semantic Web Mining, Spring '20
Group 1

***Abstract*—Previously, content-based and collaborative filtering recommendation methods have proven to be very efficient as demonstrated by multiple research papers. However these methods have 2 major drawbacks. The first being that they model the user's long term static preference while ignoring the user's short term transactional patterns. This could lead to a user's intent at a certain point of time to be submerged by his or her historical transactional behavior. Another major drawback is that it is difficult to keep track of a user's profile data due to privacy issues, where the user refuses to disclose their private information. In such cases these recommendation methods are not applicable. These problems led to the rise in popularity of Session-based Recommendation methods, where a user's session forms the basic unit of recommendation. A session could be all the items a user has purchased in a single transaction, or the songs the user has heard in the past one hour, or the web links the user has clicked on in the past half hour. The idea of session-based recommendation is to focus more on the dynamic and local information of session data, which conventional recommendation methods do not take into account. We aim to implement two session-based methods - one using LSTMs and the other using the state of the art, Transformers. By using Transformers, we use an attention mechanism which learns to focus on fewer and more relevant items in a sequence to make the next prediction. This will help give us more intuition and give us hands-on experience with these techniques.**

I. Introduction

The two primary goals [1] of a recommendation system are:

- Minimize the amount of time taken for a user to find the perfect recommendation based on his/her liking and choice (i.e, recommend the most suitable items first).
- Maximize the level of enjoyment experienced by the user once a recommendation is chosen (i.e, make sure the recommended item and the user's choice/liking is in accordance).

To achieve these goals, a number of models have been implemented and researched upon. All these models toy with different combinations of WHAT is recommended, HOW the recommendation is personalized depending on the user, WHEN the recommendation is made available and WHERE the recommendation is displayed on the user interface. The results of such combinations are then compared with one another to find the best model for a given dataset. Models used to design Recommender Systems are primarily classified into Shallow and Deep models which are further sub-categorized, as depicted in Figure 1.

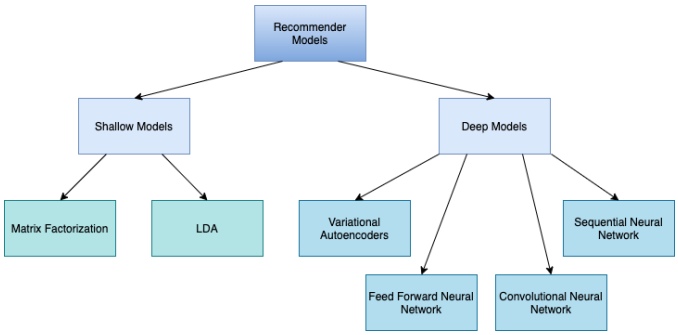

Fig. 1. Models used to build Recommender Systems

Latent Factor Models such as Matrix Factorization and Latent Dirichlet Allocation come under Shallow Models. Matrix Factorization uses a simple technique where given M number of users, N number of items and ratings of users to some items, the model recommends the top favorite items of a given user [2]. In Latent Dirichlet Allocation, the ratings matrix between users and items is used along with probabilistic topic modelling techniques to provide resultant recommendations that are not only strongly aligned with the user's interest but also those items that are currently popular in the user's community [3].

Deep Models comprise Variational Autoencoders, Feed-forward Neural Networks, Sequential Neural Networks and Convolutional Neural Networks. Variational Autoencoders use an interesting method of combining the generic user ratings along with feedback text provided by the users to encode users' preferences as a function of the textual feedback provided in a latent VAE space. By combining the ratings and textual review feedback, it is noted that the VAE model is able to provide improved recommendations [4]. Due to the amount of data present in today's world, Recommendation Systems are often faced with Big Data and Computational problems. Feedforward Neural Networks tackle these issues by using a Deep Semantic Analyzer and User Movie Attention (UMA) to produce a "Emotion". This emotion is then used to evaluate the taste of a user and thus provide corresponding recommendations [5]. Under Sequential Neural Networks, the task of providing a recommendation is viewed as a sequential process where Recurrent Neural Networks or other sequential models are used to model sequential data which is often sparse

in nature [6]. Causal Convolutional Networks are also used in Session based recommendation and have certain advantages over Recurrent Neural Networks such as parallelization and modelling long range dependencies better. The state of the art Transformers [9] can also be used for session based recommendation [8] where they make use of multihead self attention to form contextual embeddings for items along with the advantages provided by Causal Convolutional Neural Networks.

Our contributions is as follows:

- We build a recommendation model using both, Transformers as well as the LSTM.
- Provide a detailed comparative study between both these approaches.
- Use visualizations to gain better understanding on what the models are learning and make them more explainable
- Build a demo website in order for anyone to use our models for recommendations.

II. Problem Statement

In the age of data influx, where users are overcrowded with choices, recommender systems have been proven to be of immense help for decision-making. A study by McKinsey showed that 75% of Netflix views come from recommendations based on strong algorithms. A recommender system works like an information filter system that ensures increased consumer satisfaction as it narrows down the choices, which might often overwhelm the consumers.

While designing a recommender system, there are a few problems that have to be considered. The conventional content-based recommender systems and collaborative filtering recommender systems have certain drawbacks, which call for a more advanced system. They give more importance to the user's long-term static preference and ignore the short-term transactional patterns. The user's preference shift through time is thus neglected and the user's intent at the time is overshadowed by their past patterns. These systems break down each basic transactional unit into multiple user-interaction pairs and this breaks the transactional behavior and the shift in user's preference goes undetected. Also, the conventional recommender systems require user information, which is not always available due to privacy issues. Once these problems have been eliminated, recommender systems make the best way to deal with the issue of information overload.

III. Related Works

Session-based recommendation systems are still a relatively new topic in the recommendation domain and have only started attracting attention in recent years. The earliest recommendation systems made use of data mining techniques such as rule mining and sequential pattern mining [12]. Weiyang Lin et al. presented a recommendation technique in 2001 [13] based on association rule mining. Rules were mined for a specific target user, and the technique employed associations between users and between items to make recommendations. In 2006, C. Romero Morales et al. developed a tool [14] to recommend links to students based on sequential pattern mining of log files. A web recommendation system based on closed sequential patterns was proposed in 2010 by Utpala Niranjan et al. [15] The proposed system first uses the PrefixSpan algorithm to mine sequential web access patterns from preprocessed web server log data. It then constructs a pattern tree from the discovered closed sequential web access patterns and provides recommendations based on the constructed tree. The performance of the system was evaluated by precision, applicability and hit ratio.

The early 2010s saw a rise in development of statistics and machine learning techniques. This in turn led to research on recommendation systems based on Markov chain models, recurrent neural network models, etc. One drawback of sequential pattern-based approaches is there tendency to filter out infrequent items and patterns, resulting in information loss. A Markov chain-based approach takes all patterns into consideration and thus greatly decreases information loss. A method proposed by Steffen Rendle et al. [16] combined Markov chains with matrix factorization - another popular approach to recommendation systems. It is based on an underlying Markov chain where transitions are user specific. Thus, a transition matrix is learned for each user essentially resulting in a transition cube. The transition cube is then factorized by a pairwise interaction model. The method is shown to outperform both common matrix factorization and unpersonalized Markov chain model. Neural model-based approaches learn interactions over items within or between sessions and then make recommendations based on these interactions. In 2016, Balazs Hidasi et al. proposed a session-based recommendation system [17] using a GRU-based recurrent neural network. To better fit this task, the basic GRU was modified by introducing session parallel mini-batches, mini-batch based output sampling and ranking loss function. It was shown that this method significantly outperforms common baselines used for session-based recommendations. In this paper, we implement and compare two such session-based recommendation systems – LSTMs and Transformers.

IV. Dataset

We have made use of the MovieLens-1M Dataset which contains the following details:

- 1 million ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.
- Simple demographic info for the users (age, gender, occupation, zip)

We have simply considered only the user-movie interaction details, excluding the ratings, since we want the recommendation to be implicit as it has a higher similarity to real-life scenarios, where the user does not necessarily rate each item he/she interacts with.

We set the maximum sequence length to be 200 movie interactions of each user, sorted by ascending order of the timestamp to form our sequences. If the user has more than

200 interactions we consider the most recent 200 interactions, and if it is lesser than 200, we pad the sequence towards the left.

For our validation label, we use second most recent interaction ($|S^u|$-1) , for testing we use the most recent interaction ($|S^u|$) and the remaining interactions are used for training, where $|S|$ represents a user sequence.

V. System Architecture & Algorithms

In our approach, we aim to implement two session-based methods - one using LSTMs and the other using multi-headed self attention Transformers. While LSTMs are most commonly used for sequential data, they are found to be time consuming to train as well as do not perform very well in case of long term predictions. Transformers, on the other hand, are able to model long term dependencies much better. Their parallelizable architecture enables them to be trained much faster.

The reason LSTMs cannot model long term dependencies is because the path length between long range dependencies in the network tend to cause huge loss of information, since it is not able to remember earlier tokens. Also, LSTMs have to be trained sequentially which greatly increases their training time. Transformers, on the other hand, have a constant path length between long term dependencies, and use multi headed self attention to pay attention to only "relevant" tokens (using a masking) to predict the next token. This also helps avoid co-reference.

A. LSTM Based Recommendation

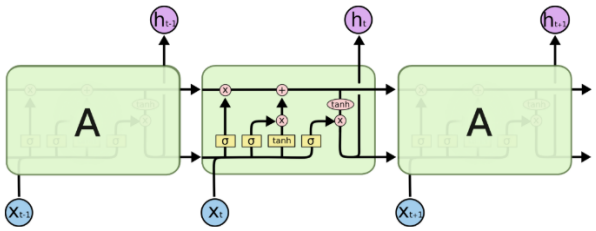

Fig. 2. LSTM Architecture

As we can see in LSTMs (Fig 2):

- Every token is encoded using its token embedding and the previous hidden state.
- This develops a dependence on previous tokens to embed the next token, hence it is not parallelizable.
- They contain multiple gates such as the Input, Forget and Output gate which help to model long term dependencies better than traditional Recurrent Neural Networks.
- Due to their sequential nature it becomes difficult to model very long sequences as the hidden states become more biased towards recent items in the sequence which overpower earlier items.
- Each hidden state at every timestep can also be directed towards a feedforward neural network followed by a softmax layer which can be used for classification tasks.

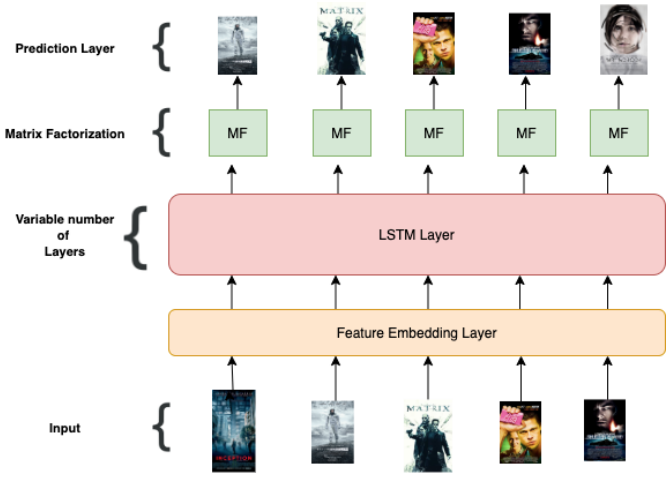

Fig. 3. LSTM Model

Our LSTM based recommendation Model is depicted in Fig 3:

- Make use of an LSTM layer which will output the user representations at each timestep
- Matrix Factorization will output a relevancy score with respect to each item
- The prediction layer will pick the items with the highest relevancy score

B. Transformer Based Recommendation

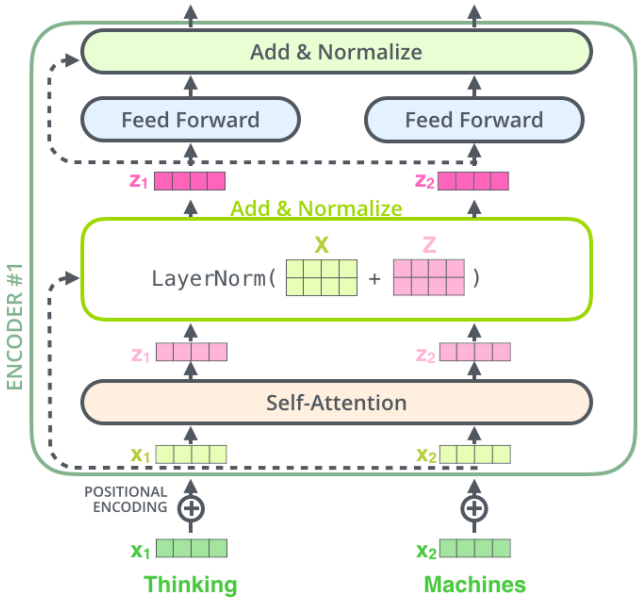

Fig. 4. Transformer Architecture

Transformers [9], was proposed very recently and achieved state of the art results on neural machine translation. Unlike

previous sequential models which rely on recurrence or convolution, the transformer relies purely on an attention mechanism known as self attention. This makes it possible for it to form contextualized embeddings of tokens, and the parallelizable architecture enables the training time to be much lesser than that of recurrence models.

*1) **Embedding**:* We make use of 2 types of embeddings for the Transformer - Item Embedding, $\mathbf{M} \in \mathbb{R}^{|\mathrm{I}| \times \mathrm{d}}$ as well as Positional Embedding, $\mathbf{P} \in \mathbb{R}^{|\mathrm{p}| \times \mathrm{d}}$, where $|I|$ represents the items (movies in our case), $|p|$ represents the maximum sequence length and d represents the latent dimension we embed our items and positions into. The positional embedding can be fixed or learned. We tried using both methods and found that the learned embeddings give us a better accuracy.
The final embedding can be represented as:

$$\mathbf{I}_s = \begin{bmatrix} \mathbf{M}_{s1} + P_1 \\ \mathbf{M}_{s2} + P_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{M}_{sn} + P_n \end{bmatrix} .(1)$$

*2) **Self Attention Layer**:* :

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{QK^T}{\sqrt{d}})V \quad (2)$$

Here Q,K,V represent the Query, Key and Value. Inuitively the QK^T computes the attention scores and we apply the softmax function over this value (after scaling) to get the attention weights. These represent how much attention to give each item when computing its output. These attention weights are multiplied with the Value in order to give the weighted sum of all the values. Hence this helps to pay attention to important items and drown out irrelevant items.

*3) **Position-Wise FeedForward**:* The self attention is a linear block, and we need to employ some non linearity in the model. Hence we make use of a position wise feed forward layer followed by a RELU activation function. The position wise is also used to take into account the interactions between the latent dimensions outputted at each timestep. Since it is position wise, the linear transformations are the same across positions but differ from layer to layer.

*4) **Matrix Factorization**:* This is the penultimate layer of our model (before the loss). Here we simply do a matrix multiplication between the output of the position wise feedforward layer at a timestep and the embedding of the item at that timestep. It produces a relevancy score which intuitively means how relevant is a item to a user, given the user's interaction history. For a positive example, the item embedding at a timestep would be the ground truth item embedding and for a negative example it would be the embedding of a randomly sampled item. The way we construct a positive example is by just 'shifting' the input sequence right by a timestep.

$$Matrix - Factorization(S_i) = F_i \times {I_i}^T \quad (3)$$

Here:

- $Matrix - Factorization(S_i)$ represents the relevancy score outputted at the ith position. S
- F_i represents the vector output of the FeedForward layer at position i.
- I_i represents the embedding of the item label at position i whose relevancy score we want to compute

*5) **Residual Connections**:* Adding more and more layers to a multi layer neural network increase performance, but only in theory. In practice adding more layers could lead to deterioration in performance. This was only the case until Residual networks were proposed [10]. Residual connections, employ the adding of lower layer features to the higher level features. Hence in the worst case, even if we employ high regularization in our model, it would be learning the identity function in deeper layers, which will not result in a decrease in performance.

*6) **Layer Normalization**:* Unlike Batch Normalization, which normalizes inputs across batches, Layer Normalization [11] normalizes the input across the features, hence enabling the inputs to be independent of other inputs in the same batch.

*7) **Dropout**:* Dropout regularization is a regularization technique which has proved to be very useful in alleviating the overfitting problems in deep neural networks. The idea of dropout is to randomly turn of some neurons with a probability p during each batch, so that the model does not give too much weightage towards one weight parameter and distributes the learning across all the parameters in the weight matrix. This is intuitively similar to an ensemble of a very large number of models.

Finally, upon implementing both the above mentioned models, we aim towards reporting a comparative study about the advantages and disadvantages of both models.

Following this, a self attention transformer model is built using the following approach and its working is illustrated in Fig 5.

- Makes use of the Transformer Encoder.
- Masked self attention will be used, enabling every item to pay attention to only the previous items.
- Matrix Factorization is done by doing a matrix multiplication betweeen the output of the position wise feedforward layer at a certain timestep as the user representation at that timestep and the positive/negative item (passed through an embedding layer) as the item representation, outputting a relevancy score with respect to each item

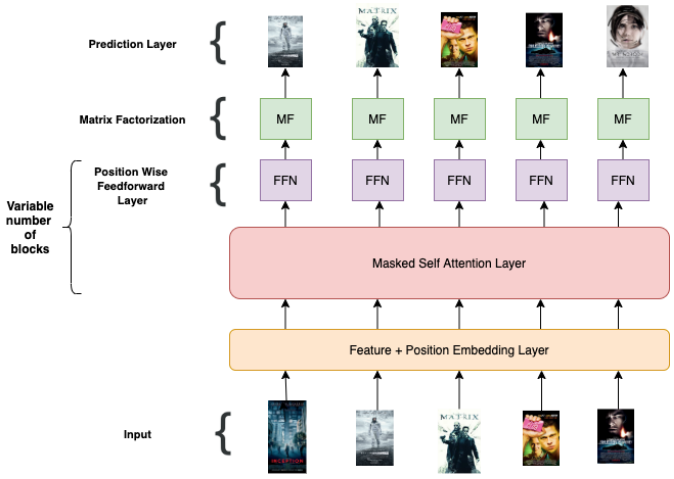

Fig. 5. Transformer Model

- The prediction layer will pick the items with the highest relevancy score

C. Loss Function

Our Loss Function is a simple binary cross entropy loss where we aim to maximize relevancy scores of positive items and minimizing that of negative items.

$$-\sum_{S^u \in \mathcal{S}} \sum_{t \in [1,2,\ldots,n]} \left[\log(\sigma(r_{o_t,t})) + \sum_{j \notin S^u} \log(1 - \sigma(r_{j,t})) \right]$$

Fig. 6. Loss Function

where:

- o_t - Positive item
- j - Negative item
- S^u - sequence of movies grouped by user and sorted by timestamp
- Validation: The 2nd most recent item used for validation ($|S^u|$-1)
- Testing: The most recent item used for testing ($|S^u|$)
- Training: The remaining items are used for training (1:($|S^u|$-2))

D. Network Training

We utilize the Adam Optimizer, which employs adaptive moment generation in order to learn our model parameters and propagate the gradients through our network. We set our hyper-parameters as:

- dropout=0.2
- d=50 (dimension)
- number of layers = 2
- number of heads = 1
- learning rate = 0.001

We utilized AWS Cloud GPUs to train our model.

VI. Evaluation

Illustated below in figure 8, figure 9 and figure 10 are comparative line graphs depicting the loss function for the LSTM and the Transformer Models using the key shown in Figure 7:

Fig. 7. Legend

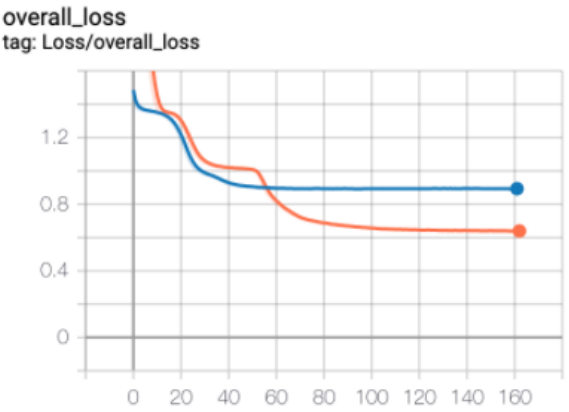

Fig. 8. Overall Loss

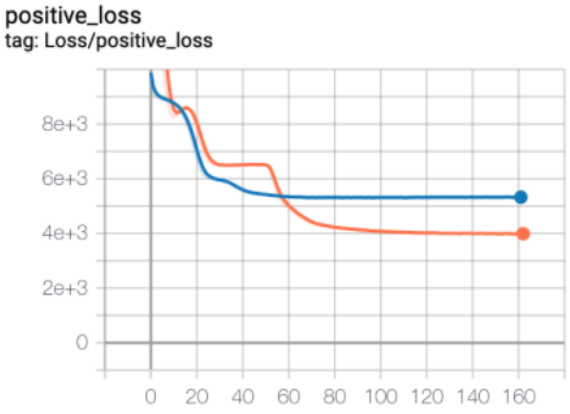

Fig. 9. Positive Loss

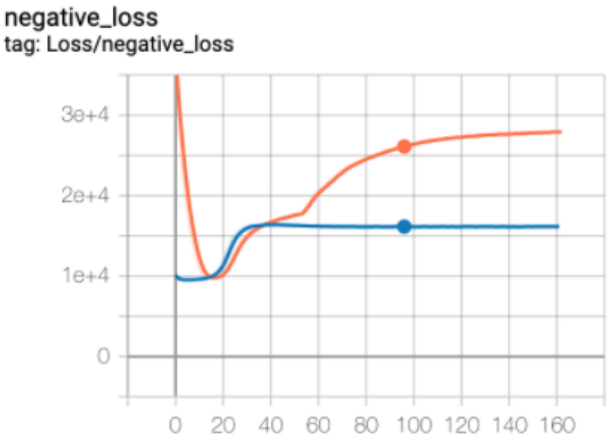

Fig. 10. Negative Loss

The Hit@10 and NDCG@10 evaluation metrics are used to compare the performance of the LSTM and Transformer Model using the testing and validation dataset and the performance is visualized below in figure 11, figure 12, figure 13 and figure 14 using figure 7 as key.

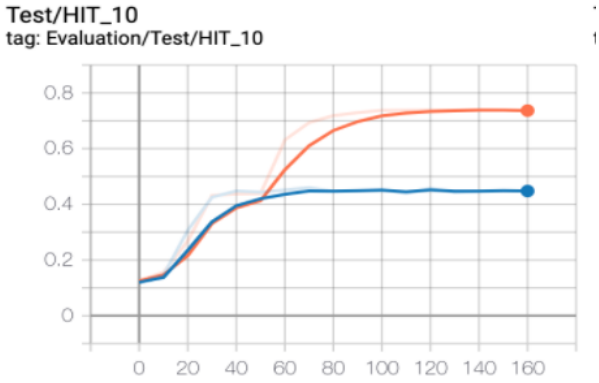

Fig. 11. Hit@10 - Test Dataset

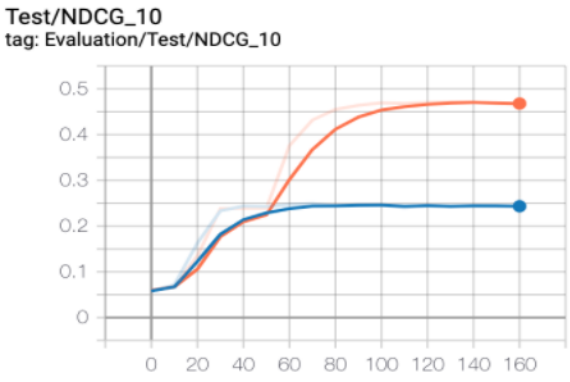

Fig. 12. NDCG@10 - Test Dataset

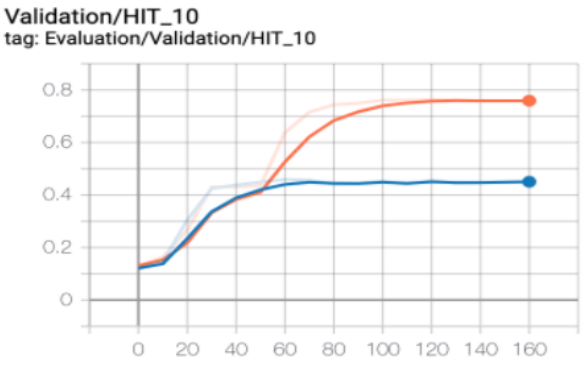

Fig. 13. Hit@10 - Validation Dataset

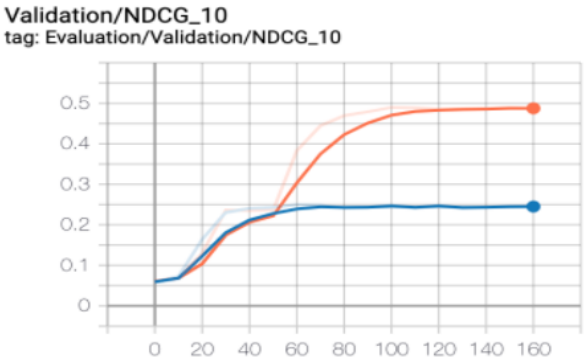

Fig. 14. NDCG@10 - Validation Dataset

We can clearly see that the transformer is performing significantly better than the LSTM on both the validation as well as the test set. We attribute this to the fact that the LSTM is not able to model long term dependencies as well as the Transformer, since the more recent movies are overpowering the earlier movies in the user sequence. However, the Transformer is able to use its attention mechanism to pay attention to only the important movies and drown out irrelevant movies, hence being able to get a more contextualized representation for the user. Hence for further demonstrations we would be paying more attention to only the Transformer based recommendation system.

Finally, the performance of the Transformer recommendation model built is compared with other state of the art models as depicted in figure 15. It can be seen that Our Model performs better than all other models under the Hit@10 metric with a score of 0.755 (except for FPMC), and out-performs all models except for FPMC and GRU4Rec+ under NDCG@10 metric with a score of 0.49.

Dataset	Metric	Pop Rec	BPR	FMC	FPMC	Trans Rec	GRU4 Rec	GRU4 Rec+	Our Model
ML-1M	Hit@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	**0.755**
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	**0.49**

Fig. 15. Our Model's Performance

VII. User Interface Designs

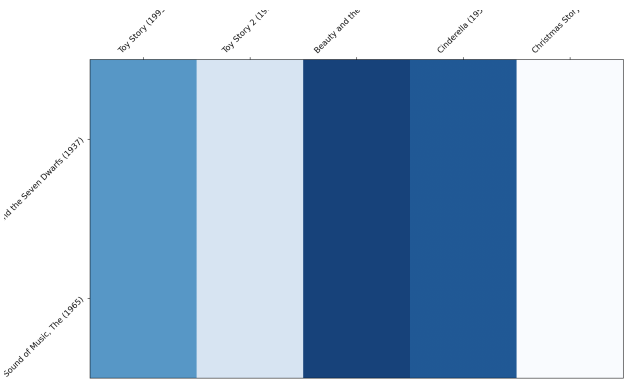

Fig. 16. Self Attention Visualization

As we can see in Fig 16, the movies on the x-axis represent our inputs and the movies on the y-axis represent the predictions our Transformer recommendation model has made. Our inputs were **[Toy Story, Toy Story 2, Beauty and the Beast, Cinderella, Christmas Story]** *(x-axis)* and our model predicted **[Snow White and the Seven Dwarfs, Sound of Music]** *(y-axis)*

The heatmap in the Fig. 16 depicts how much "attention" the model is paying to the inputs when making its predictions. The darker the colour with respect to the input depicts more attention being paid to that input. We can see that Beauty and the Beast is being paid the most attention, followed by Cinderella and so on.

These indeed make good predictions as these movies target mostly young children.

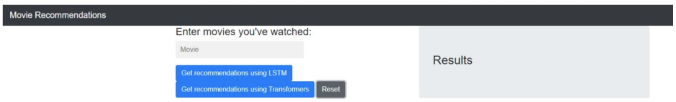

Fig. 17. LSTM and Transformers Options

The website provides the options to get movie recommendations using the LSTM model and the Transformer model. Clean design of the website minimizes distractions.

User can enter movies watched with the help of an autocomplete dropdown. From figure 19, we can see that on providing inputs: Toy Story,Toy Story 2, Beauty and the Beast,Cinderella, A Christmas Story the transformer model recommends similar children's movies like Sound of Music, Driving Miss Daisy, Snow White and the seven little dwarfs, among others

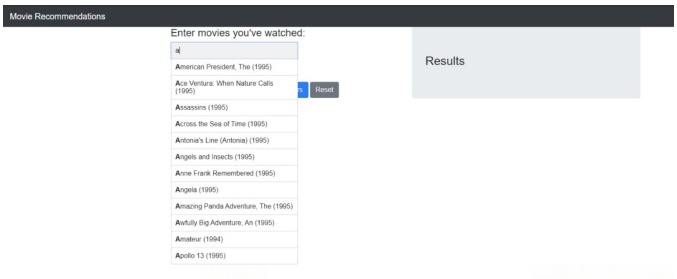

Fig. 18. Entering Movies Watched

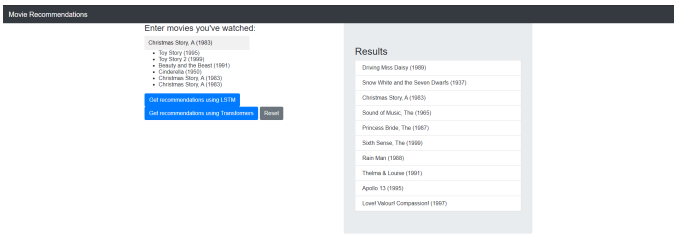

Fig. 19. Results using Transformers

VIII. Division of Work & Team Member's Contributions

A. Division of Work (Figure 20)

Task	Description	Deadline	Assigned to
Literature Review	Read various papers on methods used to build recommender systems	02/7/2020	All
Dataset Selection	Chose the widely used Movielens dataset and Million Song Dataset	02/14/2020	All
Preprocessing	Converted data into a suitable format which can be fed to the LSTM and Transformer Model	02/28/2020	All
Method 1: LSTM Session based Recommendation	Refer to the Methods and Algorithms Section	03/20/2020	Soumitro, Apeksha, Hussain
Method 2: Transformer based Session Recommendation	Refer to the Methods and Algorithms Section	03/20/2020	Nida, Arnav, Ajay
Comparison/ Evaluation	Comparing the two models using Mean Reciprocal Rank and Mean Average Precision	04/06/2020	All
Report	List down our methods and findings	04/29/2020	All

Fig. 20. Division of Work

B. Individual Team Member's Contribution

1) *Ajay Gunasekaran*:

- Explored various means and methods to build a recommendation system and studied the pros and cons of each model
- Understood the workings of the transformer model and experimented with different evaluation metrics for the transformer model.
- Performed the model evaluation for the Transformer mode through Hit@10 evaluation metric
- Helped with the visualisation of the self attention of the Transformer.
- Helped with the documentation of the findings.

*2) **Apeksha Rajkumari**:*

- Studied research done on LSTM models to design an algorithm well fitted for movie recommendation
- Worked on the LSTM model implementation with the aim to get the best possible accuracy
- Worked on setting up the local server for the website to host it from the computer

*3) **Arnav Chakravarthy**:*

- Programmed the Transformer model from scratch
- Wrote GPU enabled code to run both the transformer and LSTM model
- Helped in the evaluation using the NDCG@10 and Hit@10 metrics
- Used Tensorboard to visualize our model's performance such as the loss functions and evaluation metrics
- Built a 3D Visualization for visualizing the learned item embeddings space by projecting the high dimensional features onto a 3D space using feature extraction techniques such as T-SNE and PCA.
- Made visualizations for the self attention in order to make the model more explainable and for the reader to understand how much attention does our model give each previous item in the sequence in order to make a recommendation.

*4) **Soumitro Choudhury**:*

- Studied multiple research papers on LSTM and provided insights on possible movie recommendation algorithms with LSTM
- Worked on the LSTM model and helped in the fine tuning of model parameters
- Worked on setting up the local server for the website to enable hosting from the computer

*5) **Hussain Khuzem Lal**:*

- Worked on the LSTM model implementation and helped in the fine tuning of the parameters for the model
- Designed and developed the front-end of the website
- Added both models, LSTM and Transformer, to the back-end of the website, thus enabling it to make predictions on the click of a button

*6) **Nida Zeeshan Naveedur Rahman**:*

- Studied about various machine learning models used to build Recommender Systems and provided insights on pros and cons of certain models
- Understood the organization of the Movielens dataset and assisted with preprocessing the dataset into sequences in a form that could serve as input for the Transformer model
- Studied the work performed in [18] and assisted with fine tuning parameters of the Transformer Model
- Performed Hit@10 testing on the resultant predictions of the Transformer Model to measure performance
- Articulated and reported all findings in the report

IX. Conclusions

We were able to successfully build two recommendation model-https://www.overleaf.com/project/5e9f66e02d51560001142af2s of LSTM and Transformer for the MovieLens dataset. The models were built with careful consideration of the recent items than items in the previous parts of the sequence. As expected, the built Transformer model was able to achieve evaluations scores that outperformed our LSTM model, as well as most of the state of the art models. We are able to successfully visualize the self-attention that contributes towards the prediction of our models and our models' evaluation and loss functions through Tensorboard. We were able to successfully build a 3D visualization of our learned embedding, present in a high dimensional space onto a 3D space through feature extraction techniques such as T-SNE and PCA. We were able to provide a web visualization for both our models through the usage of Flask.

References

[1] https://www.slideshare.net/AnoopDeoras/shallow-and-deep-latent-models-for-recommender-system

[2] https://towardsdatascience.com/introduction-to-latent-matrix-factorization-recommender-systems-8dfc63b94875

[3] Bhowmick, Abhishek, Udbhav Prasad, and Satwik Kottur. "Movie Recommendation based on Collaborative Topic Modeling." (2014).

[4] Karamanolakis, Giannis, Kevin Raji Cherian, Ananth Ravi Narayan, Jie Yuan, Da Tang, and Tony Jebara. "Item recommendation with variational autoencoders and heterogeneous priors." In Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, pp. 10-14. 2018.

[5] Ibrahim, Muhammad, Imran Sarwar Bajwa, Riaz Ul-Amin, and Bakhtiar Kasi. "A neural network-inspired approach for improved and true movie recommendations." Computational intelligence and neuroscience 2019 (2019).

[6] Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. "Session-based recommendations with recurrent neural networks." arXiv preprint arXiv:1511.06939 (2015).

[7] https://towardsdatascience.com/creating-a-movie-recommender-using-convolutional-neural-networks-be93e66464a7

[8] Wang-Cheng Kang, Julian McAuley. "Self-Attentive Sequential Recommendation" arXiv preprint arXiv:1808.09781 (2018)

[9] Vaswani et al "Attention is all you need" arXiv preprint arXiv:1706.03762 (2017)

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016.

[11] L. J. Ba, R. Kiros, and G. E. Hinton, "Layer normalization," CoRR, vol.abs/1607.06450, 2016.

[12] Wang, Shoujin Cao, Longbing Wang, Yan. (2019). A Survey on Session-based Recommender Systems.

[13] Weiyang Lin, Sergio A Alvarez, and Carolina Ruiz. 2002. Efficient adaptive-support association rule mining for recommender systems. Data mining and knowledge discovery 6, 1 (2002), 83–105.

[14] C Romero Morales, AR Porras Pérez, S Ventura Soto, C Hervás Martınez, and A Zafra. 2006. Using sequential pattern mining for links recommendation in adaptive hypermedia educational systems. Current Developments in Technology-Assisted Education 2 (2006), 1016–1020.

[15] Utpala Niranjan, RBV Subramanyam, and V Khanaa. 2010. Developing a web recommendation system based on closed sequential patterns. In International Conference on Advances in Information and Communication Technologies. Springer, 171–179.

[16] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th international conference on World wide web. ACM, 811–820.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015).

[18] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In Advances in neural information processing systems, pp. 5998-6008. 2017.