Session Based Recommender System using Recurrent Neural Network

Er. Sujan Poudel[a], Shyam Maharjan[b], Er. Binaya Subedi[a,b,*]

[a] heyitsmesujan@gmail.com

[a] Nepal Kasthamandap College, Kathmandu, 44600, Nepal

[b] Nepal Kasthamandap College, Kathmandu, 44600, Nepal

Abstract

Different service providers use various recommender systems to suggest items to users. Recently, research has focused on using Recurrent Neural Networks (RNNs) for recommendation systems due to their ability to utilize past session data for current recommendations. This paper proposes a new architecture that combines two RNNs: one storing past session data and another storing current user interactions, to enhance item recommendations in the current session. The model is pre-trained using item-interaction learning, trained on a GPU, and employs the Dropout technique to reduce overfitting. Parameters such as batch size, epoch size, and dropout layers were considered in designing the RNN. The results were compared and analyzed to identify the best architecture for accurate recommendations based on past sessions. The approach involves using a second RNN layer to learn from past sessions and predict current user interests, with the first RNN storing current sessions. Item-interaction embedding was used to determine the likelihood of co-occurring item interaction pairs. The model's performance was tested on three different datasets, showing improved recommendations based on past session interactions.

Keywords: recurrent neural network; session-based recommendation system

1. Introduction

There is a vast amount of information and products available on the web. Even within a single website the number of items can be overwhelming for users. This is true for news sites, streaming services, e-commerce sites, and many other sites on the Internet. Recommender systems can help create a better user experience by helping users find what they are looking for and are interested in. They can also help businesses in different ways, like showing targeted ads, help to offer a better product, and increase user engagement. Users are generally interested in finding what they are looking for as easy as possible, or to be shown products or content that interests them, but which they normally would not have discovered on their own. E.g. Spotify helps users discovering new music, tailored to the user, through their Discover Weekly 1 playlists. When a user buys something on the e-commerce site Amazon, the site display items that other users bought together with the chosen item.

In a session-based setting, the actions of the user within a session are correlated. This means that a recommender system can observe the user's actions and improve the recommendations as the system learns more about the user's interests. Recently, RNNs has been shown to work well in the session-based setting [1, 2, 3, 4]. RNNs are good at working with sequences of data, because they have an internal memory of what they have already seen, and the ability to update and discard information in their memory. Therefore, a RNN will make more accurate recommendations as it learns more about a user. This also means that a simple RNN will struggle to make good recommendations at the start of a session.

The advantage a RNN over many other recommendation/ prediction models is that it naturally considers the order of sequences. Many other models use the relaxed assumption that the order does not matter, or that items are only dependent on the last previous item in the sequence. Solutions that take the whole sequence into account are possible, but RNNs considers the order of sequences in a very natural way that few other models do.

In one of the paper [1] it is explained how a RNN can be used for session-based recommendations. Here a straightforward implementation of a session-based RNN recommender system is made. The RNN processes the sequence of items interacted with, and produce a list of recommended items. Items are represented as one-hot vectors at the input stage. The proposed RNN model achieved marked improvements over widely used approaches, on two datasets. The datasets contained sessions of item interactions in the e-commerce and movie domains. Later, [5] looked at how the model from [1] can be improved through data augmentation, pre-training, privileged information, and output embedding. All the suggested techniques gave improvements over the original RNN model from [1]. In [4] they investigate how contextual information can be used to improve a session-based RNN recommender system. That is, a RNN similar to the one proposed in [1], is improved by considering the temporal difference between events in a session, and the input context of those events. The input context is the external situation, such as the time of day or weather, of an event. In [6], it is shown how information about items can be used to improve a session-based RNN recommendation system. They used existing methods to create feature vectors from images or text related to items. Multiple modified variants of the architecture from [1], that are able to process the additional feature vector, are proposed.

The aforementioned papers only look at recommendations on a per session basis. That is, given a user with an interaction history consisting of multiple sessions, using information from past sessions to improve recommendations in the next session could be possible. This is explained in [2].A second RNN layer is added to the model in order to process previous sessions, use this to supply the original RNN layer with information about the current session, and thus be able to improve recommendations within the current session [3]. Example: for a news site, a user will probably be interested in reading news within the same news categories that he read in previous sessions or for an e-commerce site, if a user purchased hiking boots in his last session, he will probably not be interested in another pair in his next session. However, he might be interested in additional hiking equipment such as a primus stove. In this paper, it is investigated how the straightforward implementation of a session-based RNN recommender system can be extended to make use of previous user sessions, thereby improving recommendations within a session.

There are at least two kinds of recommendations. One type is novel recommendations. These help the users to discover new content according to interest. It is a way to help the users explore, by guiding them to content that they will find interesting, which maybe they would not have found by self. An example of this is Netflix and Spotify which try to expose the user to new movies/series and music, respectively. The goal of these kinds of recommendations is often to get the user to consume more content than he normally would, but also to help the user discover content.

Another type is predictive recommendations. These help the users by showing them what they are looking for. The goal is often to provide an improved user experience, by saving the user from the work of finding the desired content. An example of this could be an e-commerce site that helps users finds what they are looking for, saving them some trouble of searching and browsing.

It is hard to evaluate the recommendations of a system that produces novel recommendations. This is because it would require actual users to evaluate the recommendations, either directly or indirectly.

Predictive recommendations, on the other hand, can be evaluated without human interaction, if the system has access to a dataset. The true actions of the user are in the dataset, so the recommendation problem becomes a sequence prediction problem. Because of this, the work in this paper focus on creating a recommender system that tries to predict the actions of users based on their past session activities.

2. Research Methodology

In this work, three datasets—Reddit, Instacart, and Last.fm—are utilized to train Recurrent Neural Networks (RNNs) due to their varied data structures. The Reddit dataset includes user activity on Reddit, with interactions manually split into sessions using a 60-minute inactivity threshold [13]. The Last.fm dataset records user listening habits, with sessions split using a 30-minute threshold to manage memory consumption by reducing the item set size [14]. The Instacart dataset contains user shopping cart logs, pre-sorted into sessions, requiring no further splitting [15]. These datasets, detailed in Table 1, provide diverse training data essential for achieving good RNN performance.

Table 1: Structures of Dataset

Dataset	Subreddit	Instacart	Last.fm
File type	CSV	CSV	TSV
Size	484 MB	106 MB	1.3 GB
Total items	40000	60000	100000
No. of users	25000	30000	1100

3. Results, Analysis And Discussion

3.1 Experimental Setup

Table 2 : Configurations used for II-RNN models.

	Reddit	Instacart	Last.fm
Embedding size	50	80	**100**
Learning rate	0.001	0.001	**0.001**
Dropout rate	0	0.2	**0.8**
Maximum recent session representation	15	15	**15**
Batch size	100	100	**100**
No. of GRU layers in RNN	**1**	**1**	**1**

3.2 Results in different baselines

Comparison of results obtained from different baselines and models developed in this paper are shown in the table below. The results are the best scores to show how the models and baselines achieved on Recall@k and MRR@k for all three datasets. The results shows that the II-RNN model performs well compared to all other scores. Scores shown in the table below are compared to the I-RNN model.

Reddit Dataset:

Table 3 : Results from different baselines compared to II-RNN on Reddit

	Recall@5	Recall@10	Recall@20	MRR@5	MRR@10	MRR@20
I-RNN	0.3482	0.4238	0.5033	0.2599	0.2699	**0.2754**
II-RNN	0.4747	0.5614	0.6448	0.3513	0.3629	**0.3687**
Item KNN	0.2173	0.3034	0.3889	0.1175	0.1289	**0.135**
Most recent	0.2152	0.2205	0.2209	0.0969	0.0977	**0.0977**
Most Popular	0.1322	0.1946	0.2647	0.085	0.0932	**0.0982**
BPR-MF	**0.0816**	**0.1189**	**0.1774**	**0.0583**	**0.0631**	**0.0671**

The table shows the score that is obtained by different models and baselines on Reddit dataset. The chart above shows the comparison of score that the models obtained on Recall@k and MRR@k where k=5, 10, 20. From above chart it is seen that the II-model used in this paper work outperforms all the baselines that are used for testing. Here, II-RNN scores 0.4747 for Recall@5, 0.5614 for Recall@10 and 0.6448 for Recall@20. Similarly, II-RNN scores 0.3513 for MRR@5, 0.3629 for MRR@10 and 0.3687 for MRR@20. Table 4 shows the highest scores performed by all the baselines and I-RNN while performing the test.

Instacart dataset:

Table 5: Results from different baselines compared to II-RNN on Instacart

	Recall@5	Recall@10	Recall@20	MRR@5	MRR@10	MRR@20
I-RNN	0.1165	0.1546	0.2049	0.0818	0.0868	**0.0903**
II-RNN	0.1412	0.1865	0.2452	0.101	0.1069	**0.111**
Item KNN	0.1123	0.1566	0.2055	0.0643	0.701	**0.0736**
Most recent	0.0001	0.0003	0.0005	0.0	0.0001	**0.0001**
Most Popular	0.0709	0.0979	0.1278	0.042	0.0456	**0.0478**
Bpr-mf	**0.0178**	**0.0343**	**0.0609**	**0.0073**	**0.0095**	**0.0113**

The table shows the score that is obtained by different models and baselines on Instacart dataset. The chart above shows the comparison of score that the models obtained on Recall@k and MRR@k where k=5, 10, 20. From above chart it is seen that the II-model used in this paper work outperforms all the baselines that are used for testing. Here, II-RNN scores 0.1412 at Recall@5, 0.1865 for Recall@10 and 0.2452 for Recall@20. Similarly, II-RNN scores 0.101 at MRR@5, 0.1069 for MRR@10 and 0.111 for MRR@20. Table 5 shows the highest scores performed by all the baselines and I-RNN while performing the test.

Last.fm dataset:

Table 6 : Results from different baselines compared to II-RNN on Last.fm

	Recall@5	**Recall@10**	**Recall@20**	**MRR@5**	**MRR@10**	**MRR@20**
I-RNN	0.1357	0.1844	0.2475	0.0863	0.0928	**0.0971**
II-RNN	0.1479	0.2048	0.2789	0.0931	0.1006	**0.1055**
Item KNN	0.0848	0.119	0.1597	0.05	0.0545	**0.0574**
Most recent	0.1061	0.1306	0.1378	0.0422	0.0456	**0.0462**
Most Popular	0.0569	0.0693	0.0863	0.0471	0.0487	**0.0499**
Bpr-mf	**0.0496**	**0.0703**	**0.1096**	**0.0315**	**0.0379**	**0.0413**

Table shows the score that is obtained by different models and baselines on Last.fm dataset for ith Epoch where i=19. The chart above shows the comparison of score that the models obtained on Recall@k and MRR@k where k=5, 10, 20. From above chart it is seen that the II-model used in this paper work outperforms all the baselines that are used for testing. Here, II-RNN scores 0.1479 for Recall@5, 0.2048 for Recall@10 and 0.2789 for Recall@20. Similarly, II-RNN scores 0.931 for MRR@5, 0.1006 for MRR@10 and 0.1055 for MRR@20. Table 6 shows the highest scores performed by all the baselines and I-RNN while performing the test.

3.3 Discussion of results in different baselines:

3.3.1 BPR-MF

BPR-MF is a strong recommender, and therefore is popular to use in real applications. II-RNN model trained in this paper was tested against the BPR-MF. And the result was outperformed by II-RNN model and II-RNN model scored well in comparison to the BPR-MF for all three datasets which can be seen from the results in Table 4, Table 5 and Table 6. This is because BPR-MF does not directly apply to the session-based setting, since it is not able to dynamically change its recommendations throughout a session.

3.3.2 Item-KNN

Item-KNN is a simple, and strong baseline. It is commonly used in practice as an item- to-item recommender [16]. In our three datasets Item-KNN performed the best among all the baselines and is represented in the following tables.

3.3.3 Most Popular and Most Recent

Most recent baseline performed well in Reddit and Last.fm dataset in comparison to Instacart. This is due to the high repetitiveness in Reddit and Last.fm dataset.

Most Popular baseline performed well in Instacart due to low repetitiveness of its dataset. The model created in this paper performed well on different baselines that has been used for testing. As stated in different papers, this was the expected case. Item- kNN and most recent baselines were the strongest baselines on the Reddit and Last.fm dataset, but were outperformed by the II-RNN.

The performance of II-RNN model developed was found to give better results on the datasets that has been used with the appropriate dropout and pre-training the datasets with Item-interaction Embedding. Ignoring these factors, performance of II-RNN was not well than the I-RNN. So the parameters in the model are still to be improved to increase the performance of the model.

4. Results

Results obtained from model comparison:

Subreddit:

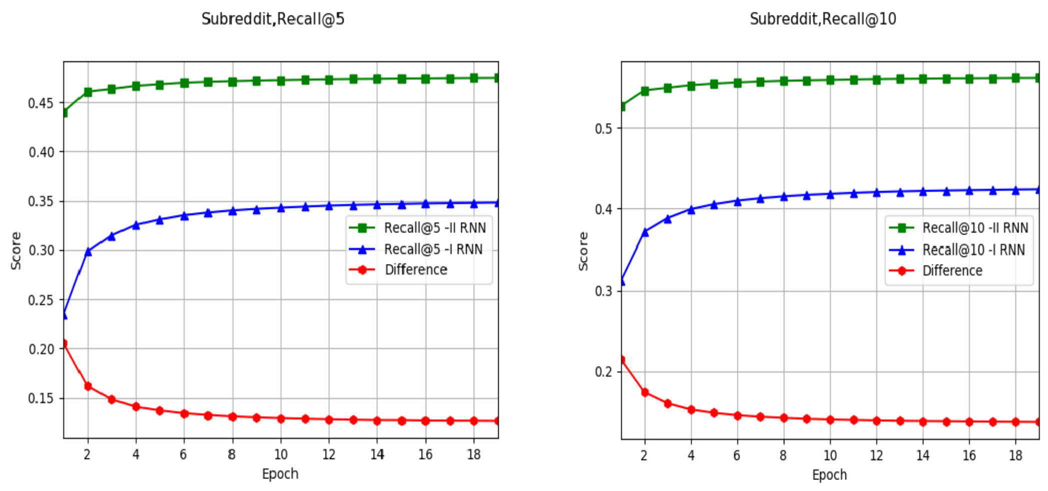


Figure 1: Comparison of results for two RNNs on the Subreddit, at Recall@k metric.

In figure 1, the subreddit dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the Recall@k metrics and the results showed that the model suggested in this paper gives better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.5614 while for I-RNN the score is 0.4238 for Recall @10 and II-RNN obtained 0.4747 while I-RNN obtained score 0.3482 for Recall@5 .The results showed in Table 6 are plotted in the above graph to compare the results for Recall@5 and Recall @10.

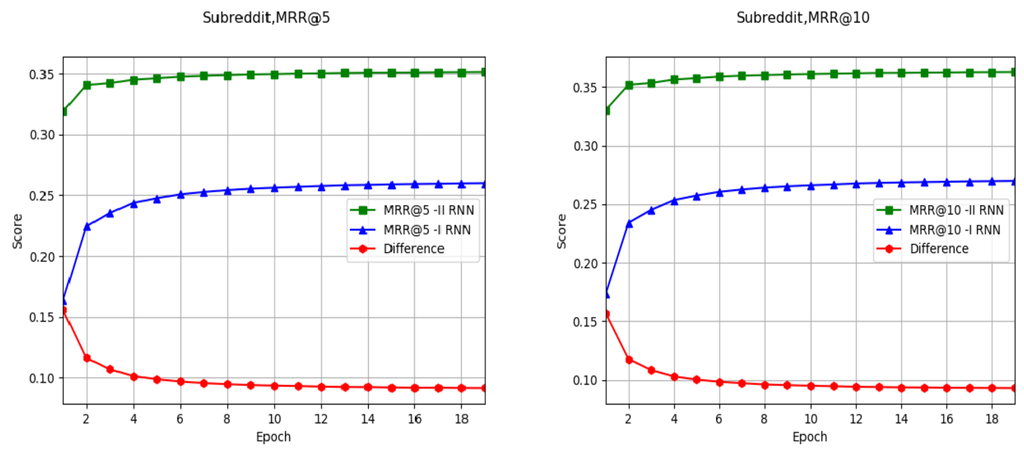


Figure 2: Comparison of results for two RNNs on the Subreddit, at MRR@k metric.

In figure 2, the subreddit dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the MRR@k metrics and the results showed that the model suggested in this paper performs better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.5614 while for I-RNN the score is 0.4238 for MRR @10 and II-RNN obtained 0.4747 while I-RNN obtained score 0.3482 in MRR@5.The results showed in Table 6 are plotted in the graph to compare the results.

Instacart:

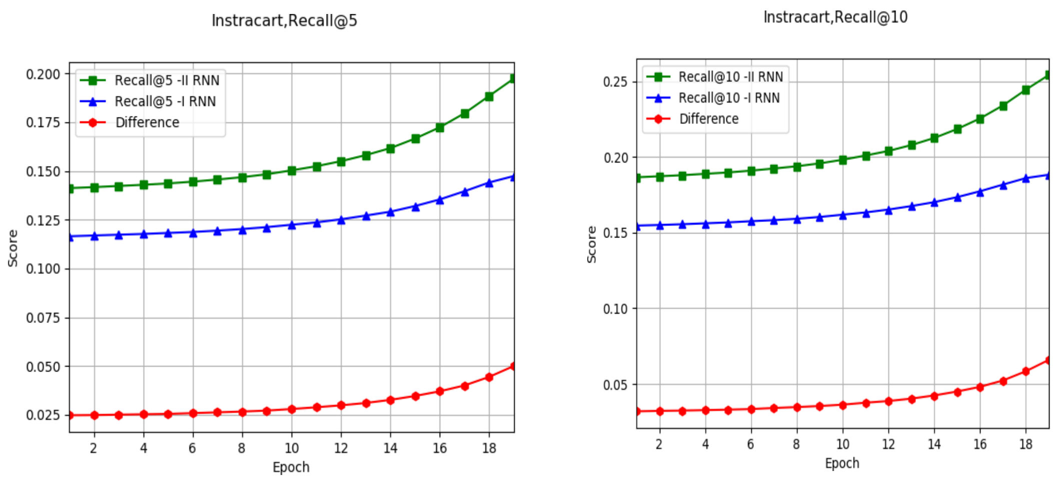


Figure 3: Comparison of results for two RNNs on the Instacart, at Recall@k metric.

In figure 3, the Instacart dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the Recall@k metrics and the results showed that the model suggested in this paper performs better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.2541 while for I-RNN the score is 0.1882 for Recall @10 and II-RNN obtained 0.1974 while I-RNN obtained score 0.1474 in Recall@5.The results showed in Table 6 are plotted in the graph to compare the results.

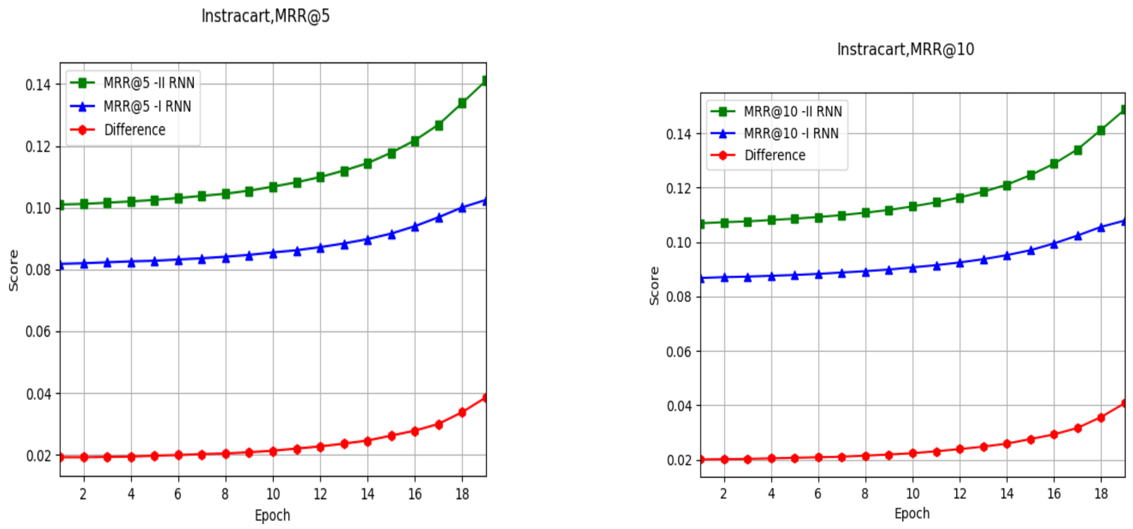
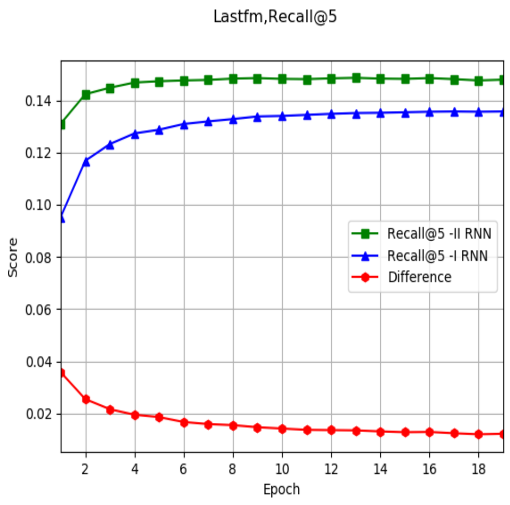


Figure 4: Comparison of results for two RNNs on the Instacart, at MRR@k metric.

In figure 4, the Instacart dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the MRR@k metrics and the results showed that the model suggested in this paper performs better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.1487 while for I-RNN the score is 0.1079 for MRR @10 and II-RNN obtained 0.1411 while I-RNN obtained score 0.1025 in MRR@5.The results showed in Table 6 are plotted in the graph to compare the results.

Last.fm:



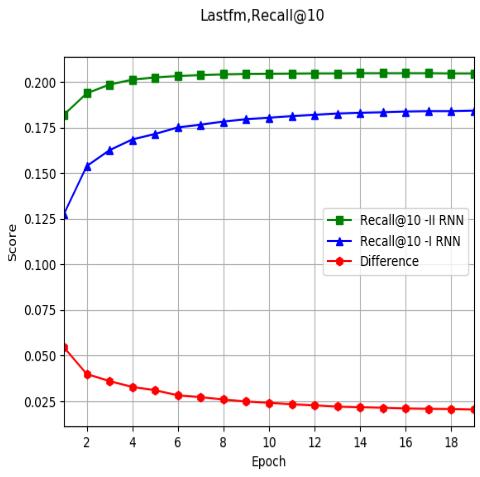
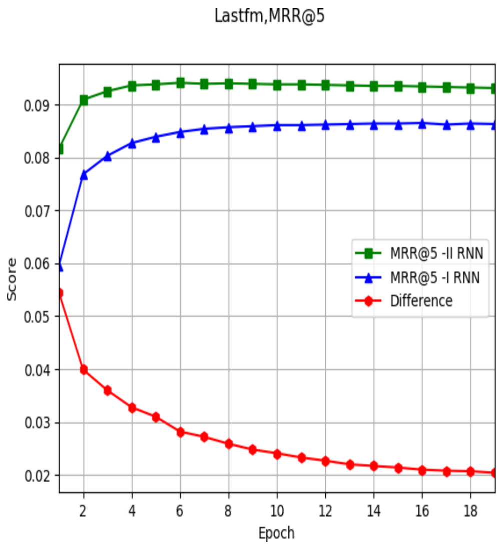


Figure 5: Comparison of results for two RNNs on the Last.fm, at Recall@k metric.

In figure 5, the last.fm dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the Recall@k metrics and the results showed that the model suggested in this paper performs better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.0952 while for I-RNN the score is 0.1277 for Recall @10 and II-RNN obtained 0.131 while I-RNN obtained score 0.3482 in Recall@5.The results showed in Table 6 are plotted in the graph to compare the results.



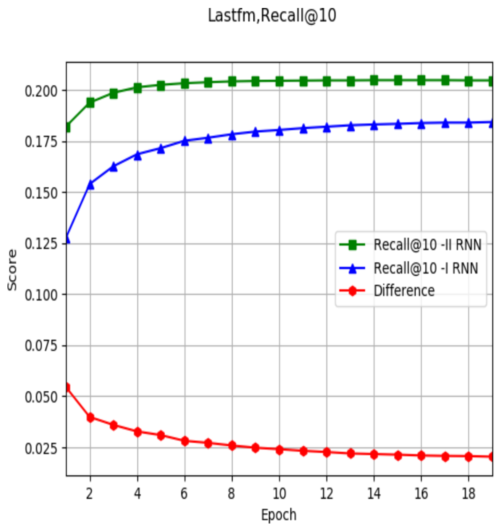


Figure 6: Comparison of results for two RNNs on the Last.fm, at MRR@10 metric.

In figure 6, the last.fm dataset is used for the comparison of two models one that is suggested in this paper which considers past session another one that does not considers past session data. The comparison is based on the MRR@k metrics and the results showed that the model suggested in this paper performs better score then the previous one, the results shown are up to 19 epochs, as the results seems to be consistent after that. The highest scored obtained for II-RNN on epoch 19 is 0.0884 while for I-RNN the score is 0.0635 for MRR @10 and II-RNN obtained 0.0816 while I-RNN obtained score 0.0595 in MRR@5.The results showed in Table 6 are plotted in the graph to compare the results.

5. Conclusion

This paper focuses on extending current RNN architecture to handle session-related problems on recommendation system, current RNN model here only considers current session of user. Though there are different techniques for recommendation, recurrent neural network is used here to solve the problem as it can handle large sets of data. Using item-interaction embedding to pre-train the RNN model shows that the likelihood of item interaction pair to co-occur greatly increases which gave the increased score on session considered recommendation.

The model was pre-trained using item-interaction learning, trained on Graphical Processing Unit (GPU) and Dropout technique is used for reducing over-fitting issues. Final results were compared with different baselines like Item-KNN, Most Popular, and Most Recent. Also, the Results between two RNN architectures were compared and analysed to verify that the developed architecture in this paper can recommend the items more perfectly based on past sessions. The model developed was tested on three datasets that contains different structures of data and the improved score was obtained while recommending the items to the user in the start of sessions based on their past interactions. The scores obtained in different performance metrics like MRR@k and recall@k used for the testing shows that the RNN architecture suggested in this work can recommend more relevant item to the user than the previous system. This is because the architecture used takes consideration on users past session activities and use them to recommend in current session which was the issue in previous system.

REFERENCES

[1] Balázs Hidasi et al., “Session-based Recommendations with Recurrent Neural Networks,” CoRR abs/1511.06939, 2016.

[2] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, Paolo Cremonesi, “Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks,” doi: 10.1145/3109859.3109896, 2017.

[3] M. Ruocco, “Inter-Session Modeling for Session-Based Recommendation,” In: CoRR arXiv:1706.07506 , 2017.

[4] YuyuZhang et al., “Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks,” In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. AAAI’14. Québec City, Québec, Canada: AAAI Press, 2014.

[5] Qiang Liu et al., “Context-aware Sequential Recommendation,” In: CoRR abs/1609.05787, 2016.

[6] Yong Kiam Tan, Xinxing Xu and Yong Liu, “Improved Recurrent Neural Networks for Session-based Recommendations,” CoRR abs/1606.08117, 2016.

[7] XiaotingZhao, DianeHu, LiangjieHong, “Learning Item-Interaction Embeddings for User Recommendations,” arXiv:1812.04407, 2018.

[8] Balázs Hidasi et al., “Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations,” In: Proceedings of the 10th ACM Conference on Recommender Systems. RecSys'. isbn:isbn: 978-1-4503-4035-9.doi: 10. 1145/2959100.2959167, 2016.

[9] “Matrix Factorization: A Simple Tutorial and Implementation in Python,” [Online]. Available: http://www.quuxlabs.com/blog/2010/09/matrix- factorization-a-simple-tutorial-and-implementation-in-python/. [Accessed 01, 2019].

[10] Ian Goodfellow, Yoshua Bengio and Aaron Courville, “Deep Learning. url:http://www.deeplearningbook. org.”.

[11] [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks- and-lstm-4b601dd822a5. [Accessed 02, 2019].

[12] [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms- and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed 02, 2019].

[13] "Subreddit Interactions," [Online]. Available: https://www.kaggle.com/colemaclean/subreddit-interactions. [Accessed 01, 2019].

[14] "The Last.fm Dataset," [Online]. Available: http://millionsongdataset.com/lastfm/. [Accessed 02 2019].

[15] "The Instacart Online Grocery Shopping Dataset 2017," [Online]. Available: https://www.instacart.com/datasets/grocery-shopping-2017. [Accessed 01, 2019].

[16] C. C. Aggarwal., "Recommender Systems: The Textbook. 1st. Springer Publishing Company, Incorporated," isbn: 3319296574, 9783319296579., 2016.

[17] Steffen Rendle et al., "BPR: Bayesian Personalized Ranking from Implicit Feedback," In: CoRR abs/1205.2618, 2012.