An Intelligent Context Aware Recommender System For Real-Estate

Faiza Rehman[1], Hira Masood[1], Adnan Ul-Hasan[2], Raheel Nawaz[3], and Faisal Shafait[1,2]

[1] School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST), Islamabad, Pakistan
{faisal.shafait}@seecs.edu.pk
[2] Deep Learning Lab, National Center for Artificial Intelligence, Islamabad, Pakistan
[3] School of Computing, Mathematics and Digital Technology, Manchester
Metropolitan University, Manchester, U.K.
R.Nawaz@mmu.ac.uk

Abstract. Finding products and items in large online space that meet user needs is difficult. Time spent searching before finding a relevant item can be a significant time sink for users. As with other economic branches, growing Internet usage also changed user behavior in the real-estate market. Advancements in virtual reality offer virtual tours and interactive map and floor plans which make an online rental websites very popular among users. With the abundance of information, recommender systems become more important than ever to give the user relevant property suggestions and reduce search time. A sophisticated recommender in this domain can help reduce the need of a real-estate agent. Session-based user behavior and lack of user profiles leads to the use of traditional recommendation methods. In this research, we propose an approach for real-estate recommendation based on Gated Orthogonal Recurrent Unit (GORU) and Weighted Cosine Similarity. GORU captures the user search context and weighted cosine similarity improves the rank of pertinent property.We have used the data of an online public real estate web portal[4]. The data represents the original behavior of the user on an online portal. We have used Recall, User coverage and Mean Reciprocal Rank (MRR) metrics for the evaluation of our system against other state-of-the-art techniques. The proposed solution outperforms various baselines and state-of-the-art RNN based solutions.

Keywords: Recommender Systems · Deep learning · Real Estate · GORU · RNN.

1 Introduction

The recommender system falls in the information retrieval domain. The main purpose of the recommendation system is to improve the consumer experience

[4] AARZ.PK

and provide users with relevant items. Recommender system is a software utility that provides suggestions for the item to be used by the user [17]. Relying on these suggestions, users make various decisions including which movie to watch, which song to hear, which news to read, which house to purchase, etc. These systems have proved to be helpful when a user deals with an overwhelming amount of information searching online through pervasive large item space. The online product catalogs evolve continuously to include high-value products such as apartments and computers; hence, the task of locating the desired choice among a large set of options is intimidating for an average customer [3].

Like many other fields, the internet has also reshaped the behavior of a consumer and a supplier in the real estate domain. It is very difficult for both the supplier and customer to survey the real estate market physically, whereas an online real-estate portal provides an abundance of information with just a few clicks. Advancements in virtual reality provide the facility of a virtual tour. The introduction of interactive maps and floor plans makes online real estate portal users first choice for property search. By integrating a sophisticated recommender system, online search time can be reduced and reduce the need for a real-estate agent in property transactions. The importance of the real estate market is demonstrated by the fact that it tends to have a big impact on the economy of any country. In the US, the real estate industry accounted for $3,372,634 million or 17. 3% of the gross state product in 2017.

Despite the importance of a recommender system in real estate domain, it has been given relatively little attention in the research community. Most of the literature in our research comes from generic recommender system techniques. Most research in recommender system focused on models where users proper identifiers and profiles are available. In this setting matrix factorization based approaches and the neighborhood model dominated the literature [7]. However, in real systems the website rarely saves a users identifier. Even if tracking is possible, the users rarely visit the rental portal for the same purpose; instead they usually exhibit session based behavior. Lack of user profiles with session based behavior leads to the use of relatively simple methods that do not take into account the user profile, i.e., item to item similarity. These methods only take into account user's last click, ignoring the context of user search.

In this work, we propose a deep learning based recommender system specifically for the real estate domain for efficient online searching.The proposed approach recommends properties based upon user search context instead of just last clicked item. Section 2 overviews the related work on recommender systems. Section 3 describes the details of our methodology. Section 4 outlines the experimental details including the features used, experimental setups, results, and comparison with the baseline methods. Section 5 concludes the paper and provides points for future research.

2 Related Work

Despite the existence of many years of research,recommender systems in real-estate have remained little focused on in the research sphere. One of the first approaches which highlights the application of a recommender system in real estate was proposed by Shearin et al. [18]. The proposed system attempted to reproduce suggestions similar to a human real estate agent. The system was based upon an interactive learning procedure which was not further particularize in this study. It was based upon extensive user feedback, gathering user views about various aspects and attributes of properties. The information gathered through questionnaires was used to set the standard filter on the database and make a recommendation about relevant properties.

Graaff et al. [5] proposed a geosocial recommender system that used data from different data sources. The system was able to make recommendations about local businesses. It used data mainly from social media. The proposed approach applied a profile matching mechanism. The profile matching was based upon collaborative filtering to make suggestions. The authors claimed that the concept of a geoprofile could be shifted to the real estate domain. However, this approach was dependent on social media data to make suggestions.

Yuan et al. [22] proposed an approach to improve the efficiency and affordability of an online housing search. They proposed a user-oriented recommender system for real-estate portal using ontology and case-based reasoning relied which relied upon user search behavior. They aimed to find the semantic construction of housing unit information and designed all the sub-modules for building a recommender system. They used questionnaires and user search behavior for semantic relationship construction. The extracted ontology contained information about the knowledge collected in the user study and real estate domains. Finally it used case representation and case indexes to find the similarity between search queries and user cases.

A simulation-based recommender study using real estate data was published by Chulyadyo et al. [12]. They proposed a personalized Probabilistic Relational Model (PRM) based on users preferences in decision making. They also demonstrated that the same PRM could be used to achieve content-based, collaborative filtering and hybrid approaches.

The first ever machine learning based approach in real-estate recommender systems was proposed by Julian et al. [14]. In their research, they used MLP to make recommendations. The proposed solution makes recommendations based upon the ratings given by the user to a specific property. The MLP takes user id and property features as input and predicts a rating for the property that the user has not already rated. They also showed the various possibilities of embedding recommenders in the real-estate portal during the user search journey. The algorithm required explicit feedback from the user to make it work,and for every recommendation, the algorithm had to predict a rating for all the properties available in the system.

2.1 Recurrent Neural Network for Recommendation

RNN is a deep model that works particularly well when dealing with sequential or temporal data. RNNs have been successfully used in image and video captioning, time series prediction,and natural language processing. Long short-term memory (LSTM) [11] are a type of RNN that work particularly well. It includes additional gates that control the hidden state of the RNN. This helps with the vanishing gradient problem*(add a fa line or two about it) that comes up in standard RNN.

Gated Recurrent Unit (GRU) [4] is comparatively simpler thn LSTM. Recently, URNN and GORU have been proposed which help in reducing the gradient explosion problem. In this work, we have used GORU [13] for real-estate recommendation problem. GORU combines the ability of unitary RNN to remember the long-term dependencies with gated RNN's ability to forget irrelevant information.

RNNs were first used to model session data in [9]. This GRU based recurrent neural network is trained with the pairwise ranking loss function Top1.This network takes clicked item IDs as input and provides a recommendation after each click. Parallel RNNs were also used to incorporate item features, i.e., textual description and a thumbnail image of the item to make recommendations [10]. This technique provides a better recommendation compared to plain RNNs but has the drawback of increased training time.

In [19] the author used data augmentation technique to improve the performance of RNN in session-based recommendation. This technique has the drawback of increased training time because in this work a single session is split into sub-sessions. RNNs have also been used in user-item collaborative filtering [21] [6] where results are not up to par and hardly outperform matrix factorization method. In [16] authors proposed hierarchical RNN (HRNN) to make recommendations in the scenario when information about the users previous session is available.

In [15] the authors proposed a recommender system for Carpooling services which allows the drivers to share rides with other passengers. This reduces the passengers' fares and time, as well as traffic congestion and increases income for drivers. Another recommender system for movie recommendations was proposed in [2] which used the content based filtering approach with fuzzy logic and conformal prediction algorithm.

The goal of our study is to use novel sequence model GORU and URNN [1] for recommendation. Results show that GORU based models train faster with fewer epochs and hidden units. At the same time it performs better in terms of different measures as compared to GRU and URNN. It also outperforms other state of the art session based recommendation techniques by a significant margin.

3 Methodology

Our proposed system consist of two main modules. The first is a candidate generation module which is based upon deep learning (i.e. Gated Orthogonal

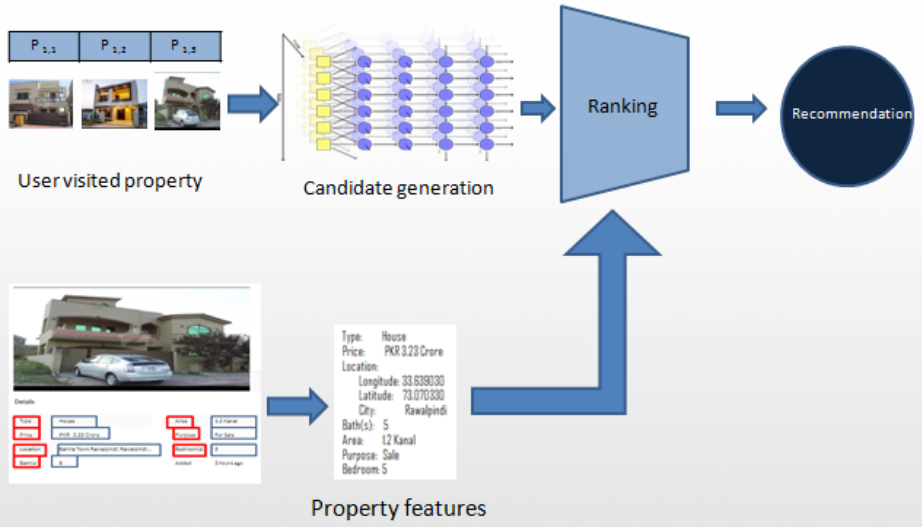


Fig. 1. An illustration of the proposed approach, *Candidate generation* module based upon deep learning takes one hot-encoding of property-ID and generate candidate properties, *Ranking* module takes candidate properties and apply Weighted Cosine Similarity between last click property and candidate properties.

Recurrent Unit) and the second is a ranking module.The candidate generation module generates properties based upon the user search context.The ranking module takes candidate properties as input and , using weighted cosine similarity ,improves the rank of the most relevant property. Figure 1 depicts the proposed solution.

3.1 Candidate Generation using GORU

This module is based on the RNN-based Gated Orthogonal Recurrent Unit (GORU). GORU models user's search behavior on the online real estate portal. For our study we used data from an a online real-estate portal. Our model takes in a 1 of N representation of user searched property ID and outputs the probability of the next possible items.

In our setup, we use a ranking loss function TOP1. For training, we use sequences of user clicked property IDs generated in session during searching online real estate website and model user search behavior. This module serves as a unit to generate the property IDs that are most likely to be the user most relevant property. Properties with a high probability serve as the candidates. Figure 2 shows the process of candidate generation.

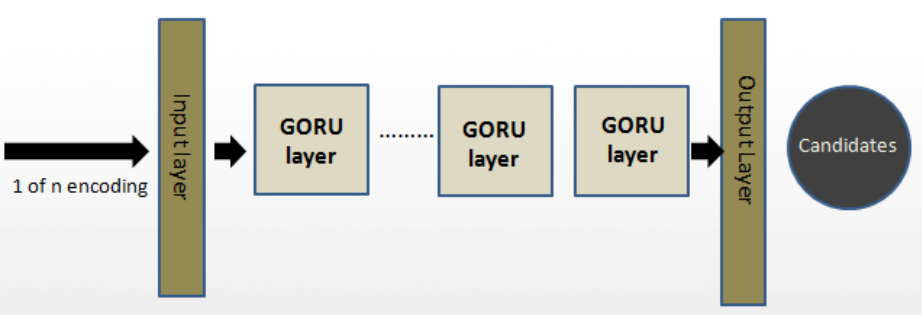


Fig. 2. Candidate Generation using GORU,*Input layer* takes 1 of N encoding of visited property ID and pass it to *GORU layers*, *Output layer* gives the probability corresponding to all the properties

Gated Orthogonal Recurrent Unit Recurrent neural network has been devised to model sequential data. The hidden state h_t of RNN makes them different from a feed-forward neural network.

$$h_t = g(Wx_t + Uh_{t-1}) \tag{1}$$

where g is the activation function. x_t is the input at time step t, h_{t-1} is previous hidden state.

GRU is gated model of RNN that deals with the vanishing gradient problem. Gates controls the changes of the hidden state The hidden state in GRU is the linear interpolation of previously hidden state and candidate hidden state.

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{2}$$

where z is the update gate computed as

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{3}$$

Where σ is the activation function.The candidate hidden state $\hat{h_t}$ is computed in a similar manner

$$\hat{h_t} = tanh(Wx_t + U(r_t \odot h_{t-1})) \tag{4}$$

and finally, the reset gate rt is given by

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{5}$$

A complex-valued matrix O is unitary when it satisfies $OO^T = I$ where I is the identity matrix. and real-valued unitary matrix is called orthogonal. Any vector multiplied by orthogonal matrix satisfies $OO^T = I$. A complex-valued matrix U is unitary when it satisfies $OO^T = I$.

$$||Ox|| = ||x|| \tag{6}$$

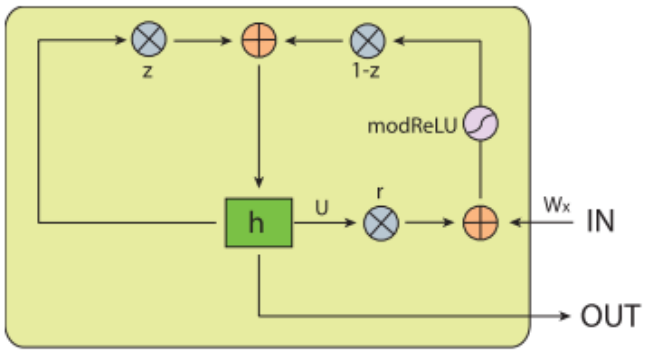


Fig. 3. An illustration of GORU [13],h is the hidden state.z and r are update and reset gate , here U is the orthogonal matrix.here activation function in $modRELU$

Due to the above property, the U matrix is able to preserve the norm of matrix passing through it and allows the gradient to propagate for a longer time. In our scenario, to use GORU we change the hidden state matrix in orthogonal matrix and activation function to modRELU given as

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h_t} \tag{7}$$

$$\hat{h_t} = modReLU(W_x x_t + r_t \odot (Oh_{t-1}) + b_h) \tag{8}$$

where $\hat{h_t}$ is candidate hidden state.

Figure 3 represents the working of GORU. We use 1-of-n encoding with session parallel mini-batches. We also tried other sequence learning algorithms i.e. GRU, URNN and LSTM. Results show that GORU outperforms the others. We also tried different loss functions, TOP1 and BPR show competitive results and outperform the others.

1 of N Encoding The input of the network is the current state of the session while the output is the item of the next event in the session. For input, we use a 1 of N encoding representation of the item. The length of the input vector equals the number of items and only the coordinate corresponding to the current item is one, the others are zeros. We also experimented with adding an additional embedding layer, but the 1-of-N encoding always outperforms.

Session Parallel Mini Batches RNNs mostly use in-sequence mini-batches for natural language processing tasks. For example, it is common practice to use a sliding window over sentences and place these windowed segments adjoining each other to form mini-batches. This strategy does not suit our task, because (1) The length of sessions can be very diverse, even more than that of sentences:

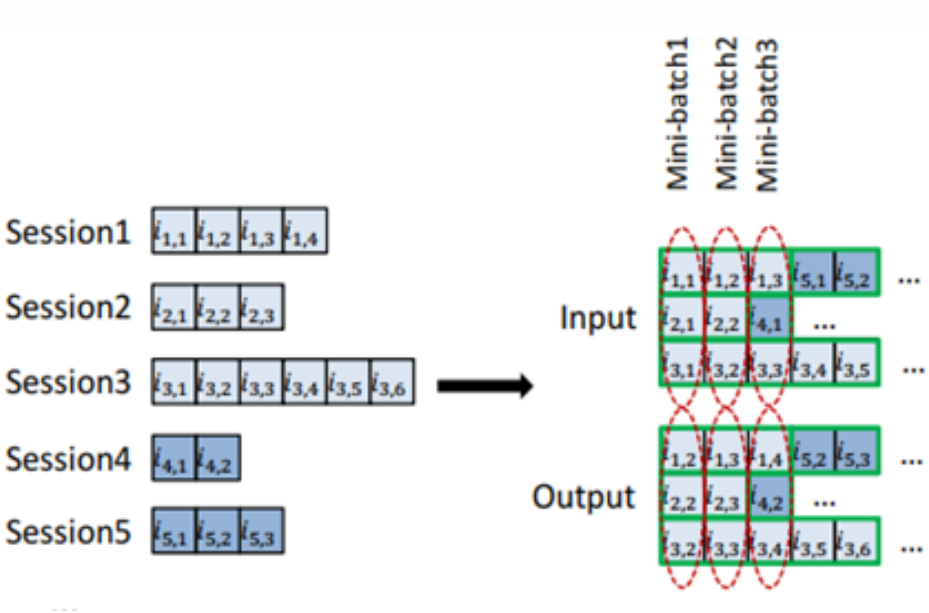


Fig. 4. Session Parallel Mini Batches

some sessions consist of only 2 events, while others may contain hundreds of events. (2) Our goal is to capture how a session progress over time, so breaking it down into segments would make no sense.

For this purpose, we use session-parallel mini-batches, depicted in Figure 4 . First, we create an order for the sessions Then, we put the first occurrence of the first X sessions to form the input of the first mini-batch (the required output is the second occurrence of our active sessions). The second mini-batch consists of the second occurrence and so on. If any of the sessions terminate than the next available session is put in its place. Sessions are considered independent,so we reset the suitable hidden state when this session switch occurs.

Ranking Loss Function The key to recommender systems is the relevance-based ranking of recommended items. Although this task can also be considered a classification task, learning-to-rank approaches generally outperform other approaches. Ranking can be point-wise, pairwise and list-wise.

Point-wise ranking independently estimates the score or the rank of items, and the loss is defined in such a way that the rank of relevant items is low. The *pairwise* ranking compares the score or the rank of pairs of a positive and a negative item and the loss is defined so that the rank of the relevant or positive item should be lower than that of the negative item. *List-wise* ranking uses the scores and ranks of all items and compares them to the perfect ordering. As this last method requires sorting, it is computationally expensive and not often

used. In our case, we use a pair-wise loss function named TOP1 proposed by Hidasi et.al [9] specifically for recommendation tasks. It is the regularized approximation of the relative rank of the pertinent property. The relative rank of the pertinent property is given by $L_s = \frac{1}{N_S} . \sum_{j=1}^{N_S} I \{\hat{r}_{s,j} > \hat{r}_{s,i}\}$. $I \{.\}$ is approximated with a sigmoid function. Optimizing for this would modify parameters such that the score for i would be high. But in some cases, certain relevant items also act as a negative example and the score tends to become increasingly higher. To circumvent this, we want to impose the scores of the negative sample to be near zero. Which is the organic expectation towards the scores of negative items, Thus regularization term has added to the function. This regularization term must be in the same range as the relative rank and acts similarly to it. Finally, the loss function is as follows: $L_s = \frac{1}{N_S} . \sum_{j=1}^{N_S} \sigma \{\hat{r}_{s,j} > \hat{r}_{s,i}\} + \sigma \hat{r^2}_{s,j}$

3.2 Ranking

In this module, we took generated candidates as inputs and measured the similarity of all candidate to a users last clicked item.The similarity measure we use in our study is weighted cosine similarity. In Tonara et al. [20]'s survey, they identify 7 major deciding factors in real estate sales and purchase. These criteria are price, number of rooms, number of bathrooms, location, building area, land area, and house certificate. As on the portal websites, the information about the house certificate is not available, we excluded this and we identified two further deciding factors, Property type (i.e. house, plot, flat etc) and property purpose (i.e. whether it is available for rent or sale). The location is subdivided into City, longitude, and latitude. Each property is represented as an array of attributes. We used min-max normalization for features normalization.

$$P1 = (\text{PropertyType}, \text{PropertyPurpose}, \text{City}, \text{Longitude}, \text{Latitude}, \text{Area}, \text{NumRooms}, \text{NumBaths}) \tag{9}$$

Assign weights according to the relative importance of features.

$$P1 = (w_1 \times \text{PropertyType}, w_2 \times \text{PropertyPurpose}, w_3 \times \text{City}, w_4 \times \text{Longitude}, w_5 \times \text{Latitude}, w_6 \times \text{Area}, w_7 \times \text{NumRooms}, w_8 \times \text{NumBaths}) \tag{10}$$

For some candidate property C:

$$C1 = (\text{PropertyType}, \text{PropertyPurpose}, \text{City}, \text{Longitude}, \text{Latitude}, \text{Area}, \text{NumRooms}, \text{NumBaths}) \tag{11}$$

Assign weights according to the relative importance of features.

$$C1 = (w_1 \times \text{PropertyType}, w_2 \times \text{PropertyPurpose}, w_3 \times \text{City}, w_4 \times \text{Longitude}, w_5 \times \text{Latitude}, w_6 \times \text{Area}, w_7 \times \text{NumRooms}, w_8 \times \text{NumBaths}) \tag{12}$$

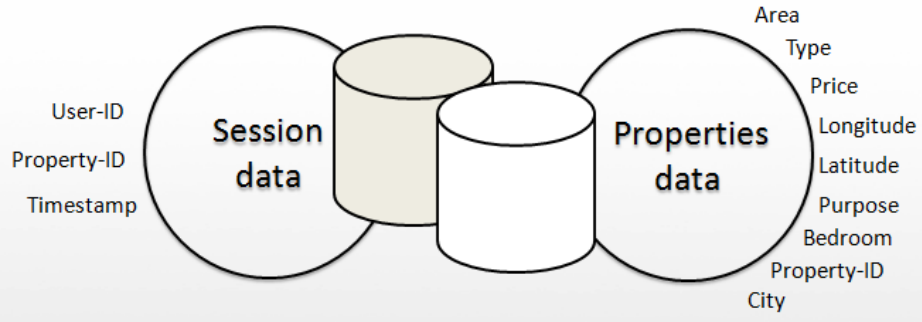


Fig. 5. *Session data* contains information about user behavior in session.*Property data* contain information about the particular real estate property.

$$\cos(\mathbf{P1}, \mathbf{C1}) = \frac{\mathbf{P1C1}}{\|\mathbf{P1}\|\|\mathbf{C1}\|} = \frac{\sum_{i=1}^{n} \mathbf{P1}_i \mathbf{C1}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{P1}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{C1}_i)^2}} \tag{13}$$

The cosine similarity measure is used to find the similarity between generated candidates and last clicked property. Candidate properties with high similarity values are used as recommendations. The result shows that this module improves the ranking of the relevant property.

4 Experimental Evaluation

To thoroughly evaluate our algorithm, we have performed a number of experiments. This section describes the data-set used and the protocol followed,as well as outlining the key results obtained.

4.1 Data Description

We have obtained the data of an online real-estate portal. It consists of two types of data,The Google analytics data of the online portal,which contains session-based user data and user browsing history, (this session based data grabs the user behavior in the form of click stream)and the property attributes stored in the database of the online portal.

Figure 5 shows the data we have used in our research study. The session based data demonstrates the user's session based behavior. Property data contains information about property attributes that the users are concerned about while purchasing property. These include primary attributes such as the number of rooms, area, and the location of the property, as well as derived attributes such as distance to school, distance to shopping center, etc. These attributes are fundamental to establishing similarity between properties and identifying a particular user's interest.

The following table (Table 1) shows session data that we have used for model training and testing purpose.

Table 1. The specification of the data set used in our study

Data	Number of Events	Number of Session
Train Data	26,121	6,390
Test Data	8,238	2,608

4.2 Baselines

The following are the most used approaches in session-based recommendation systems.

POP: Popularity predictor that always recommends the most popular item. Although this approach is quite simple, it is widely used in practical systems and in certain domains it acts as a strong baseline.

S-POP: Recommend most popular item in the current session. In our setting, we choose most popular item in the current batch.

KNN: It is a item to item collaborative filtering approach in which an item similar to the current items is recommended. Cosine similarity measure is used to find the similarity between the items session vectors.

BPR-MF: BPR-MF is a matrix factorization method. To make it work in a session based setting, we used the same approach as described in [9].

GRU4rec:. It is the first system that applies RNN in recommender system [9]. In this system, GRU has been used to generate the recommendations. This paper serves as the basis for our candidate generation module as well.

RNN with TOP-K Gain: This applies RNN with a new rank function [8]. They introduced a new class of loss functions that, together with an improved sampling strategy, provide top-k gains for RNNs.

4.3 Evaluation Metrics

Recall: Recall is also known as the sensitivity. It is the fraction of relevant items retrieved through the recommender over the total number of relevant items.

$$\text{Recall} = \frac{\text{relevant\_items} \cap \text{recommended\_items}}{\text{relevant\_items}} \tag{14}$$

MRR: In information retrieval and recommender systems the rank of the relevant item is also important, i.e., at what rank the system is retrieving the relevant item. Mean reciprocal rank (MRR) is the average of the inverse of the rank given to the first relevant item.

$$MRR = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{rank_i} \tag{15}$$

Here $rank_i$ is the rank given to the first recommended relevant item. S is the number of sessions.

User_ coverage: User coverage is the fraction of number of users getting a correct recommendation over total number of users getting recommendations.

$$\text{User\_coverage} = \frac{U_r}{U} \tag{16}$$

Here U_r is the number of user getting relevant recommendations. U is the total number of users. In our scenario the number of users is the same as the number of sessions.

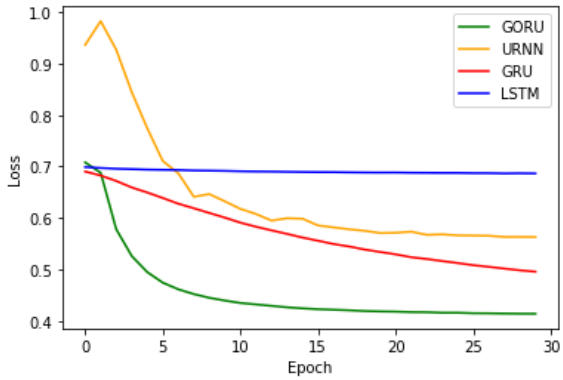
4.4 Results and Discussion

We optimized the hyperparameters by running multiple experiments and randomly selecting parameter points. Then the parameters were further optimized by tuning each parameter separately. The optimization was done on a separate validation set, then evaluated on test data. We have experimented with different sequence models i.e. LSTM, GRU, URNN, and GORU. For our experiments, we have used GeForce Titan X GPU. However, even it can be trained in a reasonable time even on a CPU.

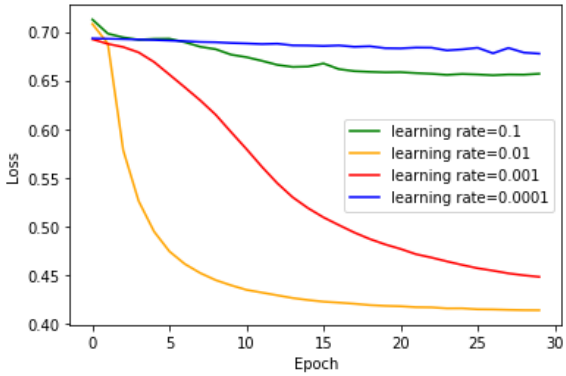
Figure 6(a) shows the loss vs epoch graph for different sequence models. Note that GORU shows faster convergence than the other sequence models,as well as achieving the lowest loss after 20 epochs. The loss function of GRU showed a steady decline but the convergance was slow. Therefore, we chose GORU as the model of choice in our study. Figure 6(b) shows the effect of using different learning rates when training GORU. According to the results, a learning rate of 0.01 performs the best. It is interesting to note that both a high learning rate of 0.1 and a low learning rate of 0.0001 resulted in poor convergence. We also tried different optimizers i.e. RMSprop, Adagrad, Adadelta and Adam. Figure 6(c) shows the convergence for different optimizers. In our case Adam performed better than others, while AdaDelta could not converge. We have experimented with different loss function i.e. TOP1, BPR, TOP1-MAX, CCE, Blackout. In terms of evaluation metrics TOP1 and BPR both give competitive results. Blackout performed the worst but at the same time increases the diversity factor.

Our proposed Hybrid GORU approach outperforms various baselines including RNN based solutions . Table 2 shows the results of the best performing network. Error analysis shows that this network performs particularly well for long sessions and the items that have higher support count i.e. the properties that are most visited. The GORU based approach has a significant gain over previous approaches included GRU based approaches. By increasing the number of hidden units within a single layer, results can be further improved with respect to all the evaluation metrics whereas adding layers decreases the performance of the network. In terms of the loss function, BPR and TOP1 both give competitive results and outperforming other loss function i.e. CCE and Blackout [6].

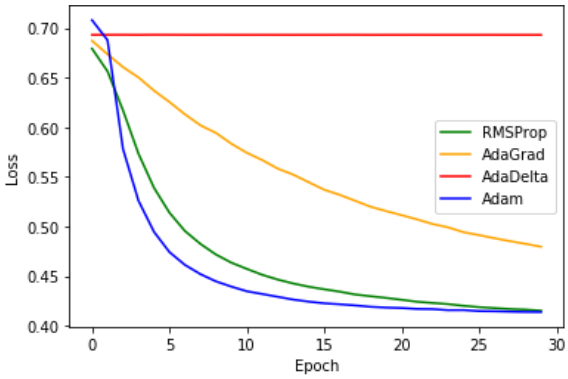
The proposed approach is popularity based,therefore it doesn't work well for rarely visited items. Moreover, the database of real estate portals tends to update frequently. This change requires the retraining of model frequently to integrate new properties in the recommendation.



(a) A comparison of different *sequence models*



(b) A comparison of different *learning rates*



(c) A comparison of different *optimizers*

Fig. 6. Loss vs Epoch graphs for different parameters

Table 2. A comparison of the proposed method with different baseline methods. The results show a clear superiority of the presented method as it significantly outperforms the baseline methods on all evaluation metrics.

Method	User- Coverage	Recall @20	MRR@ 20
POP	0.9	0.7	0.1
S-POP	31.6	35.8	26.2
KNN	40.9	35.4	14.2
BPR-MF	29.4	32.0	25.2
GRU4rec [9]	34.6	30.9	17.3
TOP-K Gain [8]	43.1	37.5	19.1
Hybrid GORU	**58.5**	**59.9**	**29.3**

5 Conclusion

The personalized real estate recommender system can help the user to find the relevant property in significantly less search time. Although our ranking module requires manually static weights ,it can be used to incorporate the business needs e.g. increase diversity in the recommendations, prioritize the candidate properties whose advertisements have been submitted earlier than others. Our work can serve as a basis to explore the real estate domain further with deep learning techniques.
Advanced features in online real-estate portals i.e. virtual tours , interactive maps, 3d tours etc. attract increasingly large numbers of visitors. To accommodate this large amount of online visitors, implementation of this recommender system for distributed platforms would be an effort worth making both for its academic value and market potential.

Acknowledgment

This research was partly supported by HEC Grant TDF-029.

References

1. Arjovsky, M., Shah, A., Bengio, Y.: Unitary evolution recurrent neural networks. In: International Conference on Machine Learning. pp. 1120–1128 (2016)
2. Ayyaz, S., Qamar, U., Nawaz, R.: Hcf-crs: A hybrid content based fuzzy conformal recommender system for providing recommendations with confidence. PLOS ONE **13**(10), 1–30 (10 2018). https://doi.org/10.1371/journal.pone.0204849
3. Chen, L., Pu, P.: Preference-based organization interfaces: Aiding user critiques in recommender systems. In: User Modeling (2007)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)

5. De Graaff, V., van Keulen, M., de By, R.A.: Towards geosocial recommender systems. In: Proceedings of the 4th International Workshop on Web Intelligence & Communities. p. 8. ACM (2012)
6. Devooght, R., Bersini, H.: Collaborative filtering with recurrent neural networks. CoRR **abs/1608.07400** (2016)
7. F. Anwaar, N. Iltaf, H.A.R.N.: Hrs-ce: A hybrid framework to integrate content embeddings in recommender systems for cold start items. Journal of Computational Science **29**, 9 – 18 (2018)
8. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: CIKM (2018)
9. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
10. Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D.: Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 241–248. ACM (2016)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
12. Hodoň, M., Eichler, G., Erfurth, C., Fahrnberger, G.: Innovations for Community Services. Springer (2018)
13. Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljacic, M., Bengio, Y.: Gated orthogonal recurrent units: On learning to forget. Neural computation **31**(4), 765–783 (2019)
14. Knoll, J., Groß, R., Schwanke, A., Rinn, B., Schreyer, M.: Applying recommender approaches to the real estate e-commerce market. In: International Conference on Innovations for Community Services. pp. 111–126. Springer (2018)
15. Qadir, H., Khalid, O., Khan, M.U.S., Khan, A.U.R., Nawaz, R.: An optimal ride sharing recommendation framework for carpooling services. IEEE Access **6**, 62296–62313 (2018). https://doi.org/10.1109/ACCESS.2018.2876595
16. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the Eleventh ACM Conference on Recommender Systems. pp. 130–137. ACM (2017)
17. Ricci, F., Rokach, L., Shapira, B.: Chapter 1 introduction to recommender systems handbook. In: Recommender Systems Handbook (2010)
18. Shearin, S., Lieberman, H.: Intelligent profiling by example. In: Proceedings of the 6th international conference on Intelligent user interfaces. pp. 145–151. ACM (2001)
19. Tan, Y.K., Xu, X., Liu, Y.: Improved recurrent neural networks for session-based recommendations. In: DLRS@RecSys (2016)
20. Tonara, D.B., Widyawono, A.A., Ciputra, U.: Recommender System in Property Business a Case Study from Surabaya , Indonesia. SPECIAL ISSUE- International Journal of the Computer, the Internet and Management **23**(May), 30–31 (2013)
21. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: Proceedings of the tenth ACM international conference on web search and data mining. pp. 495–503. ACM (2017)
22. Yuan, X., Lee, J.H., Kim, S.J., Kim, Y.H.: Toward a user-oriented recommendation system for real estate websites. Information Systems **38**(2), 231–243 (2013)