# DeepRank: Adapting Neural Tensor Networks for Ranking the Recommendations

Raaiha Humayun Kabir[1], Bisma Pervaiz[1], Tayyeba Muhammad Khan[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
{rkabir.bese15seecs, bpervaiz.bese15seecs, tkhan.bese15seecs, faisal.shafait}@seecs.edu.pk

[2] Deep Learning Laboratory, National Center of Artificial Intelligence, Islamabad, Pakistan

[3] School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, U.K.
R.Nawaz@mmu.ac.uk

**Abstract.** Online real estate property portals are gaining great attraction from masses due to ease in finding properties for rental or sale/purchase. With a few clicks, a real estate portal can display relevant information to a user by ranking the searched items according to user's specifications. It is highly significant that the ranking results display the most relevant search results to the user. Therefore, an efficient ranking algorithm that takes user's context is crucial for enhancing user experience in finding real estate properties online. This paper proposes an expressive Neural Tensor Network to rank the properties when searched for based on the similarity between the two property entities. Previous similarity techniques do not take into account the numerous complex features used to define a property. We showed that the performance can be enhanced if the property entities are represented as an average of their constituting features before finding the similarity between them. The proposed method takes into account each feature dynamically and ranks properties according to similarity with an accuracy of 86.6 percent.

**Keywords:** Neural Tensor Networks · similarity · recommender system · real estate · ranking.

## 1 Introduction

The purpose of recommendation systems is to filter information in such a way that it is presented to the querying user in an order according to his/her preference. Many well-known recommendation systems today like Amazon, Netflix and YouTube take input from the user and show the results to the querying user based on that input. However, they do not take into account that user preferences cannot be correctly judged by a single query. Recommendations generated based on a single input from the user are hence not very reliable.

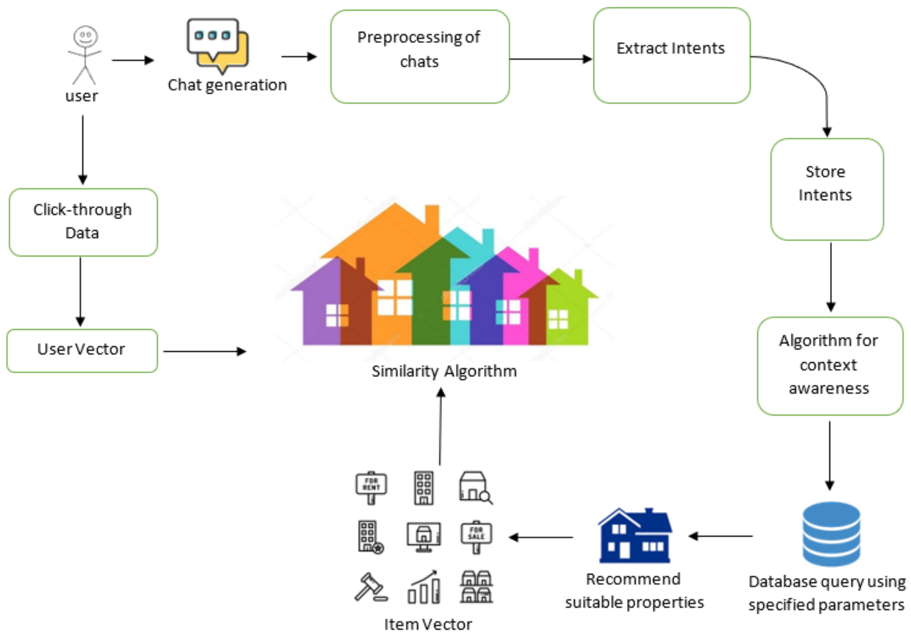

Fig. 1: Visual representation of the system used to store context and preferences, to recommend properties and to display ranked recommendations.

In this paper, we aim to improve the recommendations generated by taking into account a users context. The user's context is defined by their browsing data on a particular website and the chat history retrieved from the Chatbot, which is the end product where recommendations are to be shown. The data of a users browsing history will help to rank the recommendations fetched based on the users query.

We used real estate data set for our research where when a user wants a property with certain specifications, similar properties from the database are ranked using the browsing data of that user. Data from the real estate domain is very complex as a single property can have multiple features that describe it fully. Moreover, many personal variables can influence a person's choice regarding a property. For example, one person might be looking for a house near to a certain school for his/her convenience whereas another person might want an apartment with 4 bedrooms. Hence determining what each user may like becomes a difficult task because of personal variants and because of how vast the property types can be. We chose this domain to solve the problem mentioned above so that users can be shown the properties most important to them during a particular period.

Figure 1 is a simplified version of the model we used to rank the recommendations. A user will generate certain chats by conversing with the chatbot (a platform where a user can input their preferences like the number of rooms, size of the house, etc.) and their preferences will be saved. The chats will be preprocessed to extract the intents, which will be stored in the database to improve the Chatbot's conversation. The chatbot will generate certain recommendations for the user. These recommendations will be ranked using the user's context. Recommendations will be ranked based on what the user's visited pages (on the same website) contain, and by finding out the similarity between the generated recommendations and the visited properties. The similarity model is based on Neural Tensor Networks which instead of predicting the relations [17] is used to rank the recommendations. To make this possible we have represented each property as an entity (a description of the property's feature). To rank different properties we calculate a similarity score of two properties. Property with the highest score gets the highest rank.

The first contribution of this paper is the adaptation of the Neural Tensor Network (NTN), to find out the similarity between two entities and then rank them, instead of using it to model relational information. We modified the Neural Tensor Networks so that instead of determining the relationship between two properties, it could calculate the similarity score by assessing commonalities between two properties.

The second contribution is that since the property entities can be displayed as the average of each of the word/feature vector in the property, it can be scaled up to a great amount to incorporate a million features, which can further increase the strength of the model resulting in better output.

The third contribution is of maintaining context. The results are ranked according to the similarity scores between the recommended properties and the ones visited by the user previously. It can automatically prioritize the features of a property that are important for the user in a particular period. Hence, it takes into account both the user's manually entered preferences and inferred preferences.

## 2 Related Work

A lot of work has been done on finding good recommendations for a user who is querying the system. Several recommendation systems have applications related to e-commerce. Online shopping sites like Alibaba, Amazon, and eBay recommend to the user their desired product. We will discuss some of the existing work in the field of recommendation systems done by these organizations.

Lee et al [9, 1, 6] provided us with deep insights into how a user's context can be defined and what are the problems associated with these systems. When building a context-aware system for users performance has to maintained since multiple users may be accessing the service simultaneously. Privacy protection of the user's data is another problem that should be taken care of since context-aware systems monitor users activities.

In Shawar et al [14] the chatbot described is designed to generate a recommendation for hotels. They were able to answer queries of the user using RBM and clarifying the knowledge by asking more questions from the user but another problem emerged in this approach, that was related to context maintenance. Every time the user left the conversation their previous history would be lost.

Fuzzy et al [4] used a Hybrid Content-Based Fuzzy Conformal Recommender System for finding items online without the hassle to go through piles of data available online. However, they have not used the user's context hence the recommendations are not highly personalized.

The technique used by Qadir et al [12] provides the aggregated score of the driver to the requesting passenger in case of carpooling services because of the conflicting needs of the driver and the passenger. However, the preferences of the passengers, like the gender of the driver has not been taken into account and the priority to each ride request is assigned based on spatial closeness to the current location of the vehicle.

Anwaar et al [3] tackled collaborative filtering cold start cases of user rating records but other methods of feedback are not discussed. Ying et al [18] talked about collaborative deep ranking using the Bayesian framework model but, the only scenario with sparse implicit feedback is discussed. Liu et al [11] however, showed the comparison between ranking based recommender systems and rating-based recommender systems and also showed the effectiveness of ranking based recommender systems. Guan et al [8] talked about ranking the recommendations of tags on social networks; they have achieved this ranking using a graph-based algorithm that takes into account a user's interests. However, the algorithm only uses keywords for generating the result and the rest of the information is ignored.

Singhal et al [16] talked about the latest deep learning techniques used in current recommendation systems. This helped us to obtain a summary of the work that has been done up till now for recommendation systems using deep learning in different domains. It explains the results of using deep learning in recommender systems across different domains and whether using deep learning in recommender systems has shown any noteworthy improvement over the conventional systems for generating recommendations.

Feng et al [7] introduced a new personalized ranking metric to make the point-of-interest recommendations for social networks. This was useful to us since we also had to find a suitable metric to check our ranked recommendations.

The most relevant approach is described in Socher et al [17]. They introduce Neural Tensor Networks which are essentially neural networks but instead each layer is replaced by many tensor layers. Using these Neural Tensor Networks, they showed how two unknown entities can be related by finding the relationship between them. Before this paper, the entities were represented as single entity vector but with Neural Tensor Networks, the entities can be represented as the average of the words in it and hence the performance increases. The cost function uses triplets in the form of *(Entity 1, Relationship, Entity 2)*; if the two entities have the correct relationship between them, their confidence score is higher than when either of the two entities is replaced by a random entity. The algorithm in this paper has been used to find a relation between two entities; it handles multiple relations between entities but has not been adapted to find out the similarity between multiple entity pairs, and then ranking all the entity pairs on the basis of how similar are the two entities in a particular pair, mainly in the domain of real estate recommendations.

With the help of Neural Tensor Networks, not only can we solve the problem of ranking complex domain entities, but it can also enhance the deep learning modern recommendation systems [16] to elevate user experience.

## 3 Methodology

This section describes the proposed methodology in detail. In the subsequent subsections, we first discuss the distance model that has been traditionally used to find the similarity between the two entities. Then we introduce how we adapted the Neural Tensor Networks (NTN) to work on complex real estate data. The NTN evaluates the property entities and finds out the similarity score to rank the properties based on the user's context.

### 3.1 The Distance Model

The distance-based model calculates the distance between two entities and based on a threshold it defines whether two entities are similar or not. We initially used the distance model approach to rank the recommendations provided to the user by the chatbot. The problem with using the distance model is that it does not take into account the user's context in any way. This approach only calculates the distance between the user's preferences vector and the properties recommended by the bot. Hence, the distance model is unable to use the context to infer the user's current priorities. Therefore, the process of ranking the recommended properties according to the user's context is not possible with the distance model.

### 3.2 Neural Tensor Networks (NTN)

We use Neural Tensor Networks (NTN) for ranking different properties based on the property's features and the user's context. The NTN is used to calculate the similarity between two properties, by finding the common features amongst them and then evaluating their score. For example, we fetch 4 properties *(B, C, D, E)* from the database that are similar to property *A*; the NTN will calculate the similarity between the properties. Assuming that the similarity score for *"PropertyA similar PropertyD"* is higher than *"PropertyA similar PropertyB"*, then we can conclude that *D* will precede *B* in the ranking list. Hence the result is a ranked list of property recommendations that can be displayed. To use Neural Tensor Networks, we need to define a relationship between the two entities in question. The only relation useful for this purpose was of "similar" to find out whether two entities are similar to each other or not. Figure 2 is a visual representation of the neural tensor layer that is used to find the similarity between two property entities. The diagram contains two slices of tensor layers.

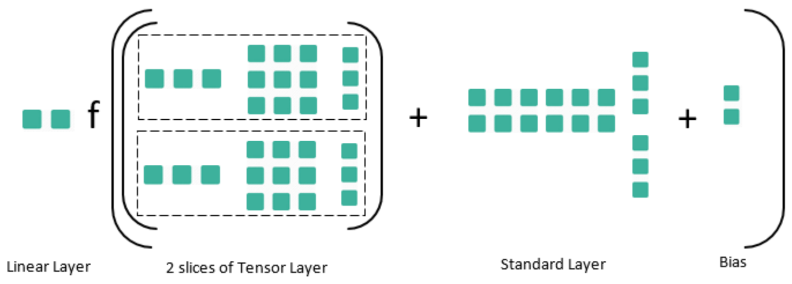

Fig. 2: Representation of a Neural Tensor Layer, for visualizing equation (1), used to compute similarity.

As described in [17], Neural Tensor Networks are used because they replace each linear layer of a neural network with a bi-linear tensor layer and hence can relate two vectors across multiple dimensions. The relation we use is similarity; therefore, we can compute the similarity score between two entities by:

$$g(e1, S, e2) = u_S^T f(e_1^T W_S^{[1:k]} e2 + V_S \begin{vmatrix} e1 \\ e2 \end{vmatrix} + b_S) \tag{1}$$

This equation from [17] was used to determine how similar two properties are to each other in terms of their features. The function f = tanh applies a standard non-linearity on each element. A tensor is represented by $W_S^{[1:k]}$ whereas its product is represented as $e_1^T W_S^{[1:k]} e2$ which produces a vector that belongs to $\mathbb{R}^k$ and every entry is computed by one slice of tensor $1 : k$.

$$e_1^T W_S^{[1:k]} e_2 \tag{2}$$

Equation (2) presents the bilinear tensor product. Each slice of the tensor is used to determine the similarity between two entities. The model, for example, will learn that two types of properties are similar if their price is within each others range, or their size is within the same range. Hence the tensor layers help to recognize how close each feature of the property is and whether both are similar or not. The activation function used for training is *tanh*, which acts as a scaled sigmoid function. It is nonlinear in nature, so it can easily be used to stack layers. Its range is bound to (-1, 1) hence, it prevents the problem of activation's blowing up. The gradient of *tanh* is stronger than sigmoid that means that the derivatives are steeper.

### 3.3 Minimizing the cost

The cost function used to minimize the cost is the Tensor Net cost function. The main idea behind it is to replace one entity in each training line with a random entity so that the score of the correct sample can be greater as compared to the one where one entity was randomly replaced. This helps the model to learn the pattern and determine the similarity between two entities. The cost function is as stated below and its validity has been proved by Socher et al [17] and is also described with detail. The NTN parameters used [17] are

$$\Omega = u, W, V, b, E \tag{3}$$

$$J(\Omega) = \Sigma_{i=1}^{N} \Sigma_{c=1}^{C} max(0, 1 - g(T^{(i)}) + g(T_c^{(i)})) + \lambda \parallel \Omega \parallel_2^2 \tag{4}$$

In the above equation, N is the number of triplets used in training. The score of a correctly identified triplet is greater than the incorrect one up to a margin of 1. We take C incorrect samples for each correct triplet and use L2 regularization of all the parameters, weighted by $\lambda$.

## 4 Experimentation

Experiments were done on the real estate data, by finding out which properties were similar to each other and then ranking them based on a particular user's context

### 4.1 Dataset

The real estate data consists of original properties in Pakistan that have been obtained from the real estate website[4] in Pakistan. Our data consists of properties that are available on the website, browsing history of a user on website and chats generated by the user when he/she converses with the chatbot to describe his/her preferences.

Each property is identified by a property identifier and similarly. Each user also has a user identifier. We have limited the property type to houses for this research work, but it can easily be extended to other property types like shops and commercial offices. Areas of barren land with no construction were not made a part of this research. Similarly, we restricted the data of the properties by passing it through a city filter so, our database only consists of properties in Islamabad, Pakistan. For each user, their chat history and preferences given to the chatbot are recorded in the database alongside the properties that were recommended to the user as a result of some previous conversations with the bot. In addition to this data, the click-through data, which consists of the user's clicking on properties of interest, is also maintained in the database. The click-through data of a user is managed unless new data is generated as a result of the user's activity.

The data set has been divided into three parts; the training data set, development data set (validation data set) and testing data set. The training data set

[4] www.aarz.pk

includes only positive examples (properties similar to each other); with a total of 103,100 properties, of which 50 percent is obtained from the original data set of actual properties in Islamabad, and 50 percent of which is synthetically generated by changing the values of the features to generate properties that can serve as an example of similar properties for the training the model. The development data set contains both positive and negative examples (properties that are not similar to each other) with a label of 1 for the positive examples and a -1 for the negative example. The development data set has 4,000 properties in total where each positive example has its counterexample. Thus, there were 2,000 positive and 2,000 negative examples. The testing data set contains 608 properties with positive and negative examples. Furthermore, all of the properties to be tested were extracted from the original data set.

### 4.2 Data Representation

The data is represented in the form of triplets. Each triplet has two entities and a relation. In our model, we used a single relation that is of similarity and the entities represent the two properties under consideration.
General representation of a triplet is:

```
Entity1 Relation Entity2
```

Representation specific to our use case is:

```
PropertyA Similar PropertyB
```

Each property consists of its features. The general representation of a property is:

```
City Sector Size Bedrooms Bathrooms Kitchen PowerRoom
           Price PropertyType Purpose
```

For example, if there is an apartment in the database that is situated in Sector G-10 Islamabad, has 3 bedrooms, 2 bathrooms, 1 kitchen, available for rent at Rs. 50,000 per month, it will be represented as:

```
Islamabad G-10 0 3 2 1 0 50000 Apartment Rent
```

### 4.3 Features of Properties

A real estate can either be land or a built unit, so we extracted features for both of these types, and concluded that the features of a plot are a subset of the features of a house. The features that were extracted are tabulated in Table 1.

Due to the data availability issue, there was no data available for many of the features from each category listed in Table 1, in several of the properties. Hence, we selected only those features for which there was complete and consistent data available. The features with the missing data could not be assigned a weight in the training, therefore excluding them was inevitable to prevent the distortion

| Land Features | House Specific Features | Surrounding Environment Features |
|---|---|---|
| City | Bedrooms | Adjacent to water |
| Locality (Area) | Bathrooms | Sidewalks or walking paths |
| Area (Size) | Kitchen | Distance from airport |
| Price | Power Room | Distance from park |
| Property type | Storey | Distance from Mosque |
| Purpose (buy or rent) | Taxes | Distance from local commute |
| | Garage/parking | Distance from market |
| | Basement | Kid friendly environment |
| | Basement Bathroom | Dog friendly environment |
| | Basement ceiling height | |
| | Basement kitchen | |
| | Basketball court | |
| | Bedrooms on basement | |
| | Bedrooms on main level | |
| | Central heating/cooling system | |
| | Furnished | |
| | Fence Yard | |
| | Grill | |
| | Guestroom | |
| | Home theatre | |
| | Library | |
| | Pool | |
| | Rec room | |
| | Security system | |

Table 1: Features that define the properties.

of the results. The model and the similarity algorithm can further be improved by obtaining data for all features listed above, which can then also be used to cater to diverse groups of people with diverse needs.

The features that were used are listed below:
1. City
2. Locality (Sector)
3. Area (Size)
4. Price
5. Property type
6. Purpose (buy or rent)
7. Bedrooms
8. Bathrooms
9. Kitchen
10. Power Room

### 4.4 Inputs for Similarity classification

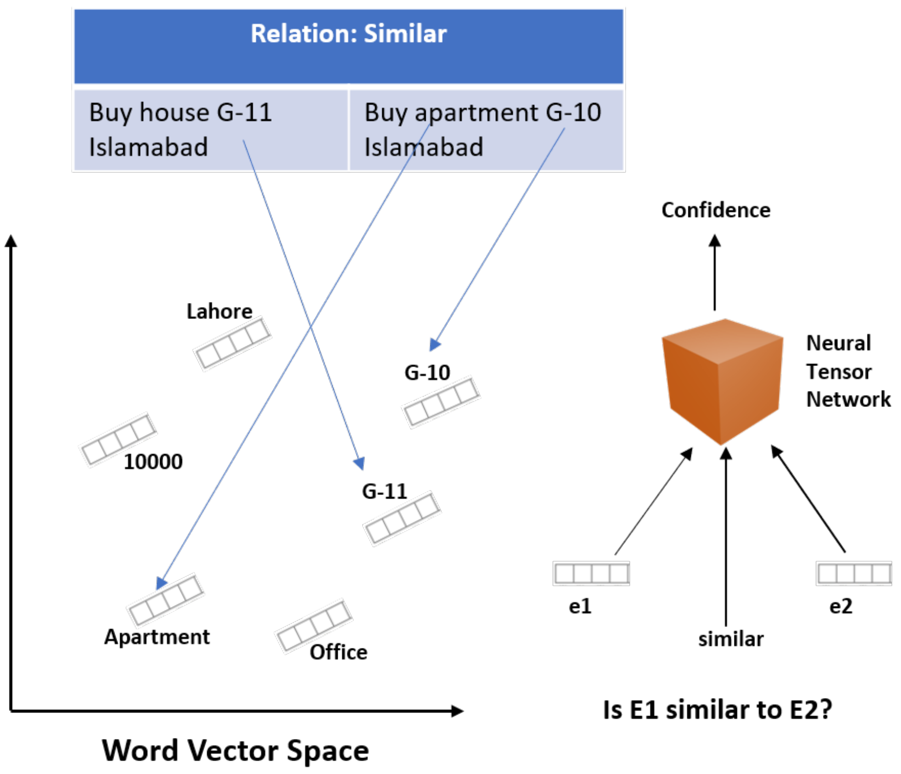

Fig. 3: Word2Vec mapping giving an overview of how entities in database are represented in the word vector space and how neural networks are used to find the confidence score.

The inputs required for testing include property entities, word embeddings, development set and the test data itself. While testing, the threshold is calculated at run-time for each sample in the test file. This threshold is later used to check if the predicted value is either above or below the threshold so that the label of 0 or 1 can be assigned accordingly. The entities that have been labeled as 1 or are above the threshold are then collected and are arranged in descending order of similarity scores. The best match properties, according to preferences and user history are then displayed to the user.

A separate set of words is generated from the entities in the whole data set. Words can be described as the unique "words" that make up an entity along with a special unique ID assigned to it for referencing it later on. Additionally, we also generated a tree structure that replaces each word in the entities data set with its relevant ID from the words set. The algorithm makes words from the entities and then maps them to a word-vector space as shown in Figure 3. This is done with the *word2vec* neural network [17]. It simply groups the vectors of similar words in a vector space. This means that it detects the similarities mathematically. *word2vec* creates vectors that are distributed numerical representations of the word features.

## 5 Results and Discussion

Testing was done on the data with original properties only, excluding all synthetic data. We labeled the test data manually with labels 1 for positive samples and -1 for negative. We used accuracy to measure the working of our model on unseen test data; accuracy was defined as the number of correct labels predicted divided by total test samples. Initially, we tested the model by giving the properties in the following format:

```
Property1 similar Property1
```

meaning the model had to determine if the two entities on either side of the similar relation are the same, then the accuracy should be 100 percent. For 20,000 similarity relations in the test file, in the above format, the expected results were obtained (100 percent). Hence this ensured that the model was at least recognizing two same properties.

In the next phase, we had to test how our model performed when the two properties are not the same. We manually determined if two properties are similar or not by assessing whether the features of the two properties are similar to each other. For example, if there is a house in the sector G-10, Islamabad with a price of Rs. 50,0000 with a size of 20 Marlas, it would be similar to a house in the neighboring sectors of G-10 (like G-11 or F-10) with a price range close to Rs. 50,0000 (within the range Rs. 40-60,0000) and a size of around 20 Marlas. Following this method, we built the whole test data to determine whether our model is predicting correct similarity relation on the original data set.

On the 20,000 lines of test data, the accuracy was 86.6 percent. Our confidence in this accuracy is high since our test data included very different samples of similarity.

Figure 4 shows a particular use case where the algorithm performs well and displays the ranked results. Suppose a user converses with the chatbot to find a suitable home for themselves. We can see from the diagram that the user's context contains information about certain properties that they visited for the past few days before conversing with the chatbot. The user then asks the bot for a recommendation close to a house for rent of around Rs. 60,000 in the sector G-11, Islamabad. The output that the user receives from the chatbot has a certain pattern to it based on the user's context. Our model inferred from the user's context that they are more interested in renting a house, within a range of Rs. 50,000 to 70,000 in Islamabad, but the sector can vary to some extent and is not entirely fixed. Hence the recommendations are generated from the database

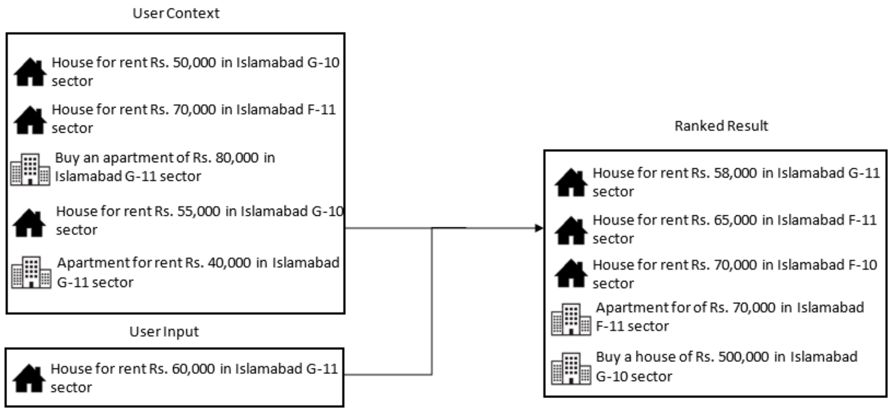
User Context
House for rent Rs. 50,000 in Islamabad G-10 sector
House for rent Rs. 70,000 in Islamabad F-11 sector
Buy an apartment of Rs. 80,000 in Islamabad G-11 sector
House for rent Rs. 55,000 in Islamabad G-10 sector
Apartment for rent Rs. 40,000 in Islamabad G-11 sector
User Input
House for rent Rs. 60,000 in Islamabad G-11 sector
Ranked Result
House for rent Rs. 58,000 in Islamabad G-11 sector
House for rent Rs. 65,000 in Islamabad F-11 sector
House for rent Rs. 70,000 in Islamabad F-10 sector
Apartment for of Rs. 70,000 in Islamabad F-11 sector
Buy a house of Rs. 500,000 in Islamabad G-10 sector

Fig. 4: Example of a positive use case. The model successfully ranks the recommendations based on the user's context.

according to the user's query and are then ranked based on the priorities inferred from the user's context. So it can be observed that the top three properties displayed are the ones with the option of renting a house specifically, within an acceptable price range in Islamabad. Whereas the last two recommendations are not fulfilling the user's request based on their context (i.e. are either not houses or are not available for rent) and have been placed lower in the list.

Figure 5 is an example of a negative use case where our model is unable to rank the recommendations on the user's context. We can observe from the context that the user has visited many different properties online before conversing with the chatbot. The context contains properties from different cities of Pakistan, with different types of properties (house or apartments), with prices that are varying a lot. The user then requests a house for rent of Rs. 60,000 in the sector G-11, Islamabad. The chatbot does retrieve some properties of Islamabad, but the results displayed have not been successfully ranked. The results displayed are although according to the query of the user, for example, there are properties in Islamabad within an appropriate price range and an acceptable location; however, the model was unable to infer any priorities from the user's context. One plausible reason for such failure is the random search pattern of the user such that our model could not determine what the user was looking for.

Finally, given the right context, the model achieved what it was intended to. It mostly predicts the similarity between properties correctly. The real-world testing data in our system is the users past visited properties on the website and, the properties recommended to them by the chatbot. The chatbot checks

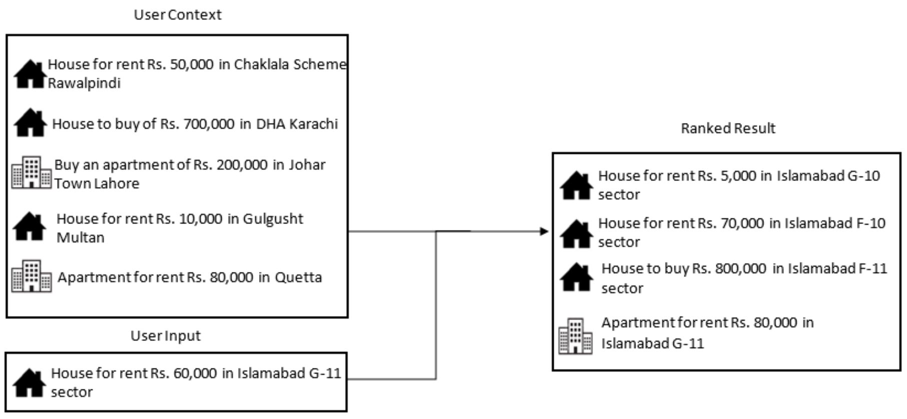
User Context
House for rent Rs. 50,000 in Chaklala Scheme Rawalpindi
House to buy of Rs. 700,000 in DHA Karachi
Buy an apartment of Rs. 200,000 in Johar Town Lahore
House for rent Rs. 10,000 in Gulgusht Multan
Apartment for rent Rs. 80,000 in Quetta
User Input
House for rent Rs. 60,000 in Islamabad G-11 sector
Ranked Result
House for rent Rs. 5,000 in Islamabad G-10 sector
House for rent Rs. 70,000 in Islamabad F-10 sector
House to buy Rs. 800,000 in Islamabad F-11 sector
Apartment for rent Rs. 80,000 in Islamabad G-11

Fig. 5: Example of a negative use case where the model was unable to rank the recommendations. User's context in this example is not clear; it contains properties where none of the features are similar.

the similarity of each property in the recommended list with each property in the users context. It then uses the predicted output (raw score) to sort them in descending order. From that list eventually, the ranked list of recommendations is displayed to the user.

## 6 Conclusion and Outlook

The main problem that we addressed was to generate recommendations that are personalized for each user based on their previous activity on the system. The use of Neural Tensor Networks to identify the similarity between two properties has proved to be very useful and efficient. Now, when the user is browsing on the website, their activity can be recorded and maintained that can be translated into their context. A user's preferences are used to fetch the properties from the database and then the similarity between the users context and properties fetched is tested. Thus, the problem of exclusion of relevant context has been addressed since now a users context is also used to rank and recommend properties.

In the future, this work can be extended by getting complete data of the features that were missing; training the model on those added features and then testing it again. Our work was limited to the city of Islamabad in Pakistan, and only for houses; this can be extended for all cities of the country. If the data for many diverse features can be obtained, the model can be trained to cater to many cities of Pakistan and can also be applied generally for any place in the world with even more accuracy.

Another suggestion for the extension that would make the results more accurate and stronger is to incorporate the user's browsing history related to similar websites (more real estate websites instead of a single site as in this paper) to gain more insight about the interests of the user [15, 1]. For example, if a user has visited 3 other real estate websites and has viewed some properties, a more comprehensive context can be built and the user's current priorities can be inferred with more accuracy due to more history.

Moreover, the algorithm does not perform well when a user visits the website for the very first time and converses with the bot, as there is no previous history regarding the user. Hence, there is no (or very little) context, which the algorithm needs to extract the priorities of the user. This means that less accurate or no priorities can be inferred resulting in the algorithm recommending unranked properties based only on the features of the property that the user entered at the beginning of the conversation with the bot. One solution to this problem is to use the existing techniques of collaborative filtering [2, 5, 13, 10] to find out similarities between different users of the website. To implement collaborative filtering for cold start cases, some personal data like each user's likes, dislikes, the location where they currently reside in, and other such useful information has to be collected from each user when they visit the website. Then if a new user with no history wants ranked recommendations, their priorities can be inferred by finding out the similarity of this user with other users of the website. If the new user has some commonalities with another user, his priorities can be inferred from the other user because they had the same taste or requirements, hence their context would be relatively similar. The recommended properties can then be ranked based on these inferred priorities for the new user.

Another use of the collected personal information is that the user profiles can be more comprehensive and have details related to their hobbies, interests, and background so that a general idea of the user's context can also be incorporated while generating ranked recommendations for the user. For example, people with big families will be interested in bigger homes that fit the needs of their families or people with children might prefer houses that have parks or recreational areas near them.

## Acknowledgment

This research was partly supported by HEC Grant TDF-029 and National Center for Artificial Intelligence (NCAI).

## References

1. Abowd, D., G., Dey, K., A., Brown, J., P., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: HUC. Lecture Notes in Computer Science, vol. 1707, pp. 304–307. Springer (1999)

2. Ahn, Jun, H.: A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. Inf. Sci. **178**(1), 37–51 (2008)
3. Anwar, F., Iltaf, N., Afzal, H., Nawaz, R.: HRS-CE: A hybrid framework to integrate content embeddings in recommender systems for cold start items. J. Comput. Science **29**, 9–18 (2018)
4. Ayyaz, S., Qamar, U., Nawaz, R.: Hcf-crs: A hybrid content based fuzzy conformal recommender system for providing recommendations with confidence. PLoS ONE **13** (09 2018). https://doi.org/10.1371/journal.pone.0204849
5. Bobadilla, J., Ortega, F., Hernando, A., Bernal, J.: A collaborative filtering approach to mitigate the new user cold start problem. Knowl.-Based Syst. **26**, 225–238 (2012)
6. Dey, K., A.: Understanding and using context. Personal and Ubiquitous Computing **5**(1), 4–7 (2001)
7. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new POI recommendation. In: IJCAI. pp. 2069–2075. AAAI Press (2015)
8. Guan, Z., Bu, J., Mei, Q., Chen, C., Wang, C.: Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In: SIGIR. pp. 540–547. ACM (2009)
9. Lee, S., Park, S., Lee, S.: A study on issues in context-aware systems based on a survey and service scenarios. In: SNPD. pp. 8–13. IEEE Computer Society (2009)
10. Lian, D., Ge, Y., Zhang, F., Yuan, Jing, N., Xie, X., Zhou, T., Rui, Y.: Content-aware collaborative filtering for location recommendation based on human mobility data. In: ICDM. pp. 261–270. IEEE Computer Society (2015)
11. Liu, Y., Yang, J.: Improving ranking-based recommendation by social information and negative similarity. In: ITQM. Procedia Computer Science, vol. 55, pp. 732–740. Elsevier (2015)
12. Qadir, H., Khalid, O., Khan, M.U.S., Khan, Rehman, u.A., Nawaz, R.: An optimal ride sharing recommendation framework for carpooling services. IEEE Access **6**, 62296–62313 (2018)
13. Sedhain, S., Sanner, S., Braziunas, D., Xie, L., Christensen, J.: Social collaborative filtering for cold-start recommendations. In: RecSys. pp. 345–348. ACM (2014)
14. Shawar, Abu, B., Atwell, E.: Chatbots: Are they really useful? LDV Forum **22**(1), 29–49 (2007)
15. Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback. In: SIGIR. pp. 43–50. ACM (2005)
16. Singhal, A., Sinha, P., Pant, R.: Use of deep learning in modern recommendation system: A summary of recent works. CoRR **abs/1712.07525** (2017)
17. Socher, R., Chen, D., Manning, D., C., Ng, Yan-Tak, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 926–934 (2013)
18. Ying, H., Chen, L., Xiong, Y., Wu, J.: Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback. In: PAKDD (2). Lecture Notes in Computer Science, vol. 9652, pp. 555–567. Springer (2016)