


Please cite the Published Version

Kabir, RH, Pervaiz, B, Khan, TM, Ul-Hasan, A, Nawaz, Raheel  and Shafait, F (2019) DeepRank: Adapting Neural Tensor Networks for Ranking the Recommendations. In: 3rd Mediterranean Conference on Pattern Recognition and Artificial Intelligence, MedPRAI 2019, 22 December 2019 - 23 December 2019, Istanbul, Turkey.

DOI: https://doi.org/10.1007/978-3-030-37548-5_13

Publisher: Springer

Downloaded from: <https://e-space.mmu.ac.uk/625001/>

Usage rights:  In Copyright

Additional Information: "The final authenticated version is available online at https://doi.org/10.1007/978-3-030-37548-5_13."

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from <https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines>)

DeepRank: Adapting Neural Tensor Networks for Ranking the Recommendations

Raaiha Humayun Kabir¹, Bisma Pervaiz¹, Tayyeba Muhammad Khan¹,
Adnan Ul-Hasan², Raheel Nawaz³, 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. Ranking results are vital in order to display the most relevant search results to the user. Similarly, ranking properties according to the user’s context is important for enhancing user experience. In this paper we propose an expressive neural tensor network to find similarity between two property entities to rank the properties when searched for. Previous similarity techniques do not take into account the numerous complex features used to define a property. We showed that performance can be enhanced if the property entities are represented as an average of their constituting features before finding the similarity between them. After training the model using the context and the user’s preferences, we tested it to find out the similarity between the properties suggested to the user and the one’s in the user’s history, and then ranked the results from most to least similar. Our model hence takes into account each feature dynamically and predicts 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 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 user are hence not very accurate.

A lot of work has been done on how to find good recommendations for a user who is querying the system. In this paper our goal is to improve the recommendations generated by taking into the account a users context. The user’s context

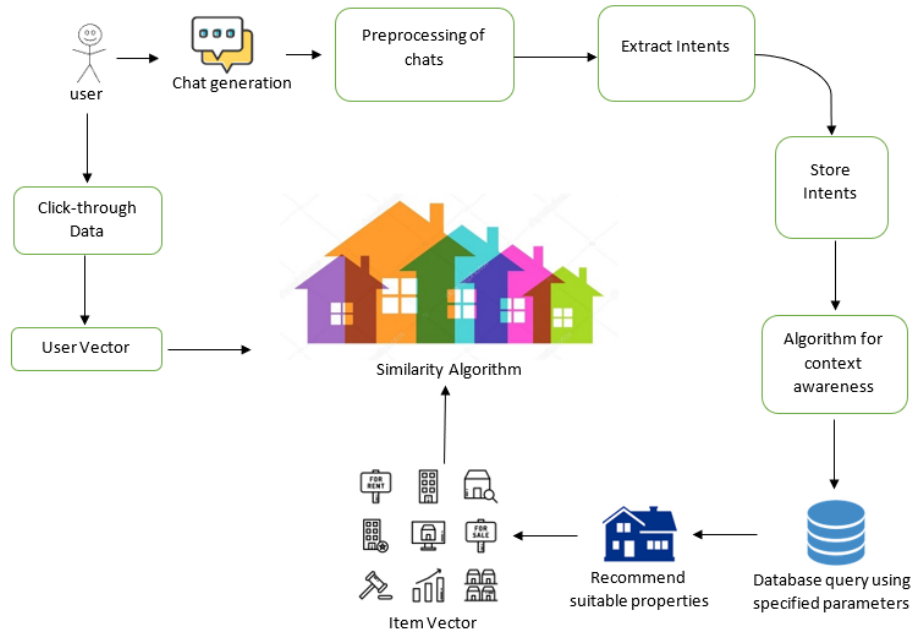


Fig. 1. Visual representation of the system used to store context and preferences, recommend properties and display ranked recommendations.

is defined by their browsing data on a particular website and the chat history retrieved from the Chat bot 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 a very complex as a single property can have multiple features that describe it fully. Moreover, there are many personal variables which 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 time 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 pre-processed 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 [18] 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 which feature of a property is the most important for the user in a particular time period. Hence, it takes into account both the user’s manually entered preferences and the inferred preferences.

2 Related Work

There are several recommendation systems that 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.

Argal et al [4] presented the history of how chatbots evolved with time. First of all, AIML language used topics and categories to answer queries. Automatic answer generating technique became popular where only the most significant words were searched. Later, chatbots started using the information in the questions asked by the user to continue the conversation. However, the chatbots then could only converse about the domain specific information on which they were trained. Lee et al [10, 1, 7] provided us with deep insights about how a user’s context can be defined and what are the problems associated with the systems that are context-aware. In Shawar et al [15] the chatbot described is designed to generate recommendation for hotels. They were able to answer queries of 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 [5] used a Hybrid Content Based Fuzzy Conformal Recommender System for finding items online without the hassle to go through piles of data available online. The technique used by Qadir et al [13] 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. 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 [19] talks about collaborative deep ranking but has used the Bayesian framework model to do so. Liu et al [12] 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 [9] talks 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. Singhal et al [17] talks 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 [8] 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 [18]. 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. Prior to 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 are 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 can also enhance the deep learning modern recommendation systems [17] to elevate user experience.

3 Neural Tensor Networks for Deep Ranking

This section introduces how we adapted the NTN to work on complex real estate data. The NTN performs the evaluation on the property entities and finds out the similarity score to rank the properties based on the user’s context.

3.1 Problem with 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 when used, 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 as it is only able to calculate the distance between two vectors. 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

We use neural tensor networks (NTN) for ranking different properties on the basis of the property’s features and the user’s context. The Neural tensor network 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 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 which can be displayed. To use Neural tensor networks, we need to define a relationship among 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 below 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.

As described in [18], 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:

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

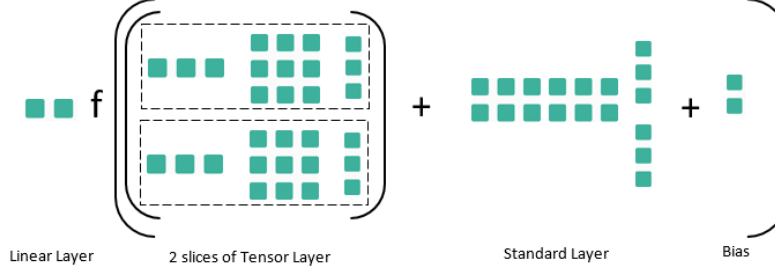


Fig. 2. Representation of a Neural Tensor Layer, for visualizing equation (1), used to compute similarity.

This equation from [18] 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.

$$e_1^T W_S^{[1:k]} e_2 \quad (2)$$

Equation (2) presents the bilinear tensor product. Each slice of 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, prevents the problem of activation's blowing up. The gradient of \tanh is stronger than sigmoid which means that the derivatives are steeper.

3.3 Minimizing the cost

The cost function used to minimize the cost is 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 Andrew N.g et al [18] and is also described with detail. The NTN parameters used (as by Andrew N.g et al [18]) are

$$\Omega = u, W, V, b, E \quad (3)$$

$$J(\Omega) = \sum_{i=1}^N \sum_{c=1}^C \max(0, 1 - g(T^{(i)}) + g(T_c^{(i)})) + \lambda \| \Omega \|_2^2 \quad (4)$$

4 Experimentation

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

4.1 Data set

The real estate data consists of original properties in Pakistan which have been obtained from the real estate website in Pakistan; Aarz.pk. Our data consists of properties that are available on the website, browsing history of 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. 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 along side 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 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 divide into three parts; the training data set, development data set (validation data set) and testing data set. The training data set 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 counter example. 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 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 came to the conclusion that the features of a plot are 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 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 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

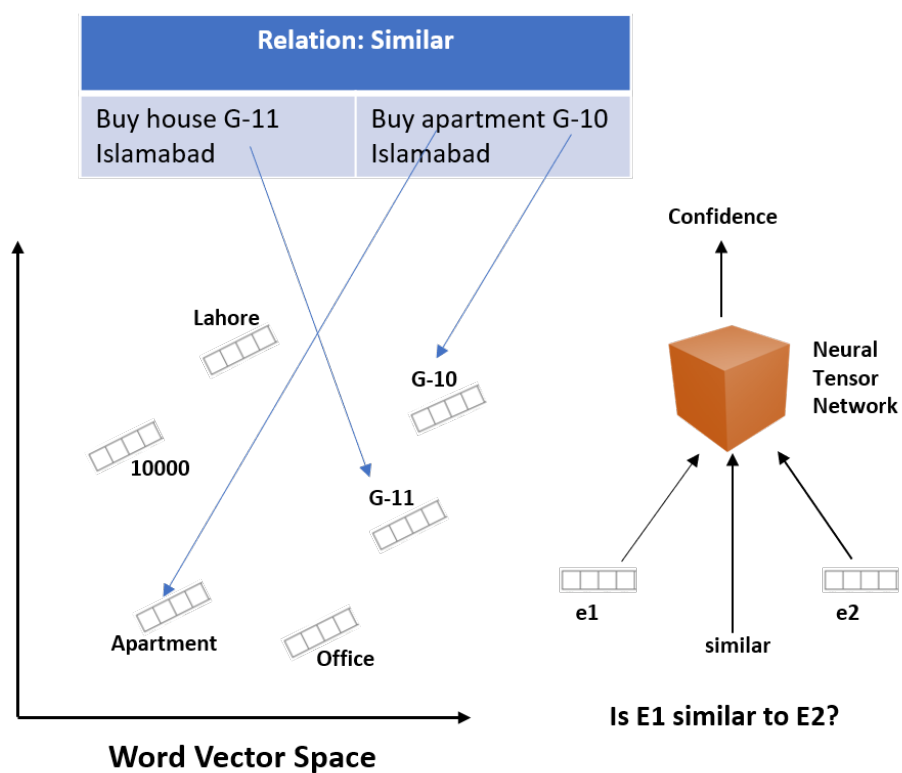


Fig. 3. Word2Vec mapping giving an overview of how entities in database are represented in the word vector space and how neural networks are used to find the confidence score.

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.

4.4 Inputs for Similarity classification

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 labelled as 1 or are above threshold are then collected, and are arranged in a descending order of similarity scores. The best match properties, according to preferences and user history are then displayed to the user.

Separate set of words are 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 which replaces each word in the entities data 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. It simply groups the vectors of similar words together 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 labelled 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 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 exactly 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 exactly same properties.

In the next phase, we had to test how our model performed when the two properties are not exactly 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 neighbouring 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 on 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 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 an 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

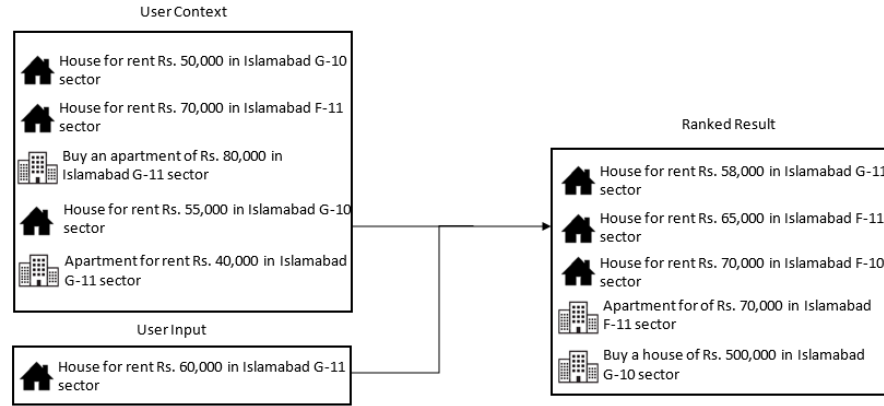


Fig. 4. Example of a positive use case. The model successfully ranks the recommendations based on the user's context.

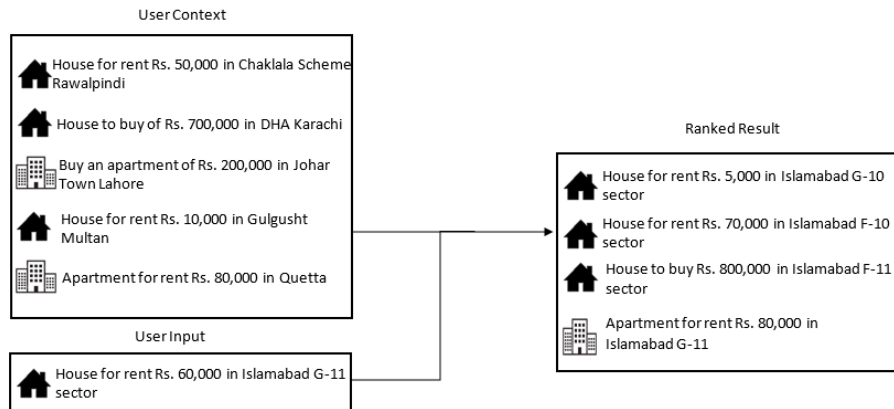


Fig. 5. Example of a negative use case where the model was unable to rank the recommendations. User's context in this example is not clear; it contains properties where none of the features are similar.

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. Hence, was not able to rank the recommendations. The problem was that the user's context was so haphazard that a pattern could not be recognized and the model could not determine what the user was actually 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 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 are 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. 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 which can translate into their context. When the user enters his/her preferences, they are used to fetch the properties from the database and then 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 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 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 [16, 1]. For example, if a user has visited 3 other real estate websites and has viewed some properties, a more com-

prehensive 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 from. This means that less accurate or no priorities can be inferred resulting in the algorithm recommending unranked properties only based on the features of the property which the user themselves 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, 6, 14, 11] 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, 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 an other 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 collection 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 definitely be interested in bigger homes that fits 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.

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. Argal, A., Gupta, S., Modi, A., Pandey, P., Shim, S., Choo, C.: Intelligent travel chatbot for predictive recommendation in echo platform. In: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). pp. 176–183 (2018). <https://doi.org/10.1109/CCWC.2018.8301732>

5. 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>
6. 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)
7. Dey, K., A.: Understanding and using context. *Personal and Ubiquitous Computing* **5**(1), 4–7 (2001)
8. 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)
9. 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)
10. 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)
11. 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)
12. 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)
13. 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)
14. Sedhain, S., Sanner, S., Brazian, D., Xie, L., Christensen, J.: Social collaborative filtering for cold-start recommendations. In: *RecSys*. pp. 345–348. ACM (2014)
15. Shawar, Abu, B., Atwell, E.: Chatbots: Are they really useful? *LDV Forum* **22**(1), 29–49 (2007)
16. Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback. In: *SIGIR*. pp. 43–50. ACM (2005)
17. Singhal, A., Sinha, P., Pant, R.: Use of deep learning in modern recommendation system: A summary of recent works. *CoRR* **abs/1712.07525** (2017)
18. 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)
19. Ying, H., Chen, L., Xiong, Y., Wu, J.: Collaborative deep ranking: A hybrid pairwise recommendation algorithm with implicit feedback. In: *PAKDD* (2). *Lecture Notes in Computer Science*, vol. 9652, pp. 555–567. Springer (2016)