Suggest me a movie for tonight: Leveraging Knowledge Graphs for Conversational Recommendation

Rajdeep Sarkar, Koustava Goswami, Mihael Arcan and John P. McCrae
Insight SFI Research Centre for Data Analytics,
Data Science Institute, National University of Ireland Galway
`firstname.lastname@insight-centre.org`

Abstract

Conversational recommender systems focus on the task of suggesting products to users based on the conversation flow. Recently, the use of external knowledge in the form of knowledge graphs has shown to improve the performance in recommendation and dialogue systems. Information from knowledge graphs aids in enriching those systems by providing additional information such as closely related products and textual descriptions of the items. However, knowledge graphs are incomplete since they do not contain all factual information present on the web. Furthermore, when working on a specific domain, knowledge graphs in its entirety contribute towards extraneous information and noise. In this work, we study several subgraph construction methods and compare their performance across the recommendation task. We incorporate pre-trained embeddings from the subgraphs along with positional embeddings in our models. Extensive experiments show that our method has a relative improvement of at least 5.62% compared to the state-of-the-art on multiple metrics on the recommendation task.

1 Introduction

Conversational Recommender Systems (CRSs) are goal-oriented dialogue systems focusing on recommending products to users through multi-turn dialogues. In these dialogues, the user and the recommender system take turns to interact with each other. CRSs hold enormous potential in the e-commerce industry wherein users can be recommended products based directly on the understanding of their requirements. However, users are not always entirely aware of their preferences while purchasing products. A CRS enables the user to make an informed decision while purchasing a product by taking into consideration the information about different products and matching them to their needs. This is extremely useful in situations of high-involvement products. By understanding the context and learning the user's preference, CRS can suggest products to the users which in turn will lead to higher consumer satisfaction and low buyer's remorse.

The three central components of CRSs are the *dialogue manager*, the *user modelling system*, and the *recommendation engine* (Jannach et al., 2020). The *dialogue manager* focuses on managing the dialogue states along with user intents and decides on the next actions to take. The *user modelling system* models the user profile using the dialogue content. The *recommendation engine* generates an appropriate recommendation to users by considering the dialogue states and user profiles into account.

More recently, works on leveraging information from Knowledge Graphs (KGs) have been gaining ground. KGs contain factual information about real-world entities in a structured format. Such external knowledge helps in adapting to a specific domain as they contain rich information about products and their features. This knowledge helps the recommendation engine to utilise additional knowledge about products which in turn leads to better suggestions. Leveraging such graphs helps in making a better recommendation as additional information is made available to the system apart from the dialogue content and user behaviour. Therefore, we focus on improving recommendations provided to users by incorporating KGs in dialogue systems for movie recommendation. However, to leverage the benefits of

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: `http://creativecommons.org/licenses/by/4.0/`.

KGs, there is a need for constructing appropriate subgraphs for the targeted domain, thereby reducing the amount of redundant information made available to the system. Subgraphs that are rich in domain information containing a low amount of noise are desirable. We, therefore, study the benefits of subgraphs created using N-hop and PageRank (Page et al., 1999) approaches. Furthermore, we analyse which subgraphs are best suited for the task at hand.
In this work, we build upon the work of Knowledge-Based Recommendation Dialog (KBRD) (Chen et al., 2019). The authors extract movie entities from dialogues and utilise information from a KG to suggest movies to users. We incorporate pre-trained entity embeddings and make use of positional embeddings to improve the performance of the system. The main contributions of our work are as follows:

- We conduct extensive experiments on different subgraph extracted from DBpedia(Auer et al., 2007) to show that the entire information contained in this resource is not beneficial and that there is a need for optimal subgraph creation technique.

- We show that using pre-trained entity embeddings supplemented with positional embeddings allows the model to learn better entity representation for recommendation.

2 Related Works

Rich (1979) suggested the use of natural language to suggest books to users by asking them questions in natural language and learning their personalities and choices. Recently, there has been a shift from the traditional collaborative system techniques for the recommendation (Sarwar et al., 2001; Zhang et al., 2016) to CRSs (Christakopoulou et al., 2018). Recommendation systems have made a shift towards deep neural networks as shown in the work of Zhang et al. (2019). CRSs have also moved towards the use of neural networks (Li et al., 2018; Chen et al., 2019). Christakopoulou et al. (2018) use recurrent neural network-based models to recommend videos to users. Zhang et al. (2016) explore the use of knowledge bases in recommendation tasks. They leverage heterogeneous information from the knowledge base to improve the performance of the recommendation system. Sun et al. (2018) propose an embedding based approach to learn semantic representations of entities and paths in a KG to characterise user preferences towards items. Wang et al. (2019) improve the performance of the recommender system by learning the embeddings for entities in the KG using TransR (Lin et al., 2015) and refining and discriminating the node embeddings by using attention over the neighbouring nodes of a given node. Wang et al. (2018) and Li et al. (2020) focus on solving the task of goal-oriented conversation recommendation for cold-start users. Wang et al. (2018) simulate the propagation of user preferences over the set of knowledge entities by automatically and iteratively extending a user's potential interests along with links in a graph. While learning the entity representations, prior works have used pre-trained embeddings for a node and fine-tuned those embeddings using the node's neighbourhood. Instead of fine-tuning the pre-trained embeddings, we extract subgraphs well suited for the domain, and learn entity embeddings on them, thereby allowing the model to capture information apart from the node's neighbourhood, in turn improving the performance of the system.

CRSs have been used to solve the goal-oriented recommendation dialogue task for specific domains. Li et al. (2020) generate new venues for recommendation using graph convolution networks and encode the dialogue contents using hierarchical recurrent encoder-decoder (HRED) (Sordoni et al., 2015) and thereby recommend locations to users. Li et al. (2018) paved the way for the use of deep learning in CRSs. They released the ReDial dataset wherein users are recommended movies based on the conversation they have with the recommender. They use a hierarchical recurrent encoder-decoder (Sordoni et al., 2015) to develop the conversational recommender model trained on the ReDial dataset. Chen et al. (2019) extend the work of Li et al. (2018) by incorporating the use of KG to recommend movies to users. They also show that dialogue and recommendation in CRSs are complementary tasks and benefit one another. However, they do not take into account the sequential nature of items. Instead of creating a sessions graph to learn the user-representation, we infuse positional embeddings into the entity embeddings to preserve the temporal preference of the user over the movie entities.

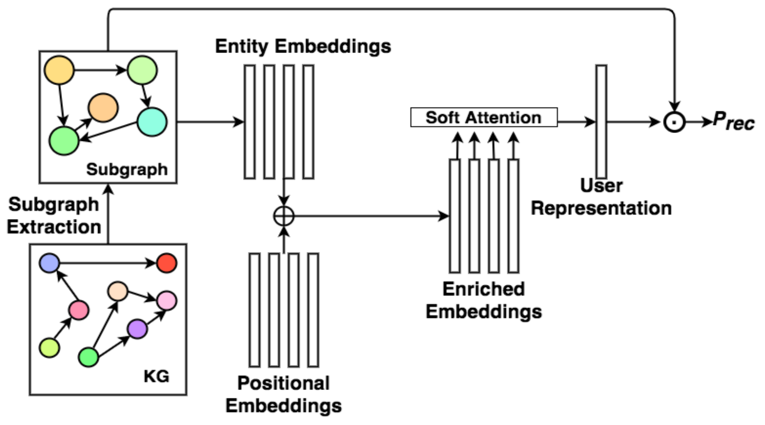


Figure 1: Overview of our model. A subgraph is constructed from the KG. The entity embeddings are learnt on the extracted subgraph. Entity embeddings belonging to a particular user is enriched with positional embeddings ($\bigoplus$ represents elements wise addition) and then passed through a soft-attention layer to represent the user. The score of each entity is calculated by taking a dot-product ($\odot$ represents the dot-product of two vectors) of its embedding with the user representation. Finally, the probability of each entity is calculated by passing the scores of all entities through a softmax layer.

3 Dataset

For this work, we use the ReDial dataset (Li et al., 2018), which is an annotated set of dialogues where a recommender suggests movies to a recommendee. The dataset consists of 10,006 dialogues with a total of 182,150 utterances. We split the dataset into 80% for training and 10% each for validation and testing.

For the task of leveraging a KG, we use DBpedia as the KG for this task. It is an open KG that stores structured content from Wikimedia projects. As DBpedia is served as open linked data, as suggested by the KBRD task, entities (movies, actors, directors, etc.) present in the conversations are linked to DBpedia entities using the method put forward by Daiber et al. (2013), where the authors perform entity linking using the generative probabilistic model. Entities are furthermore split into two categories: movie entities and mentioned entities. Movie entities are those entities, which are annotated movies present in the ReDial dataset. Mentioned entities are the entities that are linked to DBpedia as suggested in Chen et al. (2019). Entities related to actors, producers, and directors are examples of mentioned entities. The dataset consists of 6,924 movies and 4,765 mentioned entities. Among the movie entities, 824 movies from the dataset could not be linked to DBpedia. Similarly, 121 mentioned entities could not be linked to DBpedia. We introduce the *unlinked* entities as isolated nodes in the graph. Since the total number of nodes and edges present in the DBpedia KG has a magnitude of 10^6 and 10^7 respectively, we need to extract a relevant subgraph for this task. We build the initial set of subgraphs by considering N-hop neighbours of the movie nodes present in our dataset. When constructing a graph using the N-hop technique, the subgraph grows exponentially which induces noise in the subgraph. To counteract this issue, we use the PageRank algorithm (Page et al., 1999) to extract subgraphs for the task.

4 Methodology

To address the recommendation task, we begin with constructing a subgraph from the DBpedia KG. We then learn entity embeddings on the subgraphs thus constructed using the method described by Balažević et al. (2019). The learned embeddings are used for user representation. Finally, we compute the similarity of the user with the movie entities. Overview of our model is shown in Figure 1.

4.1 Subgraph creation

To construct the subgraphs for analysing the information contributed by different kinds of subgraphs, we use the N-hop neighbours and the PageRank algorithm. In all of the cases, the initial set of seed nodes are the set of all the movie entities present in the ReDial dataset. During this process, we do preserve

Table 1: Properties of subgraphs constructed for the movie domain from DBpedia. As the number of hops increases, the number of nodes and edges increase while the density decreases. Subgraphs constructed using PageRank show a slight difference in their properties as result of different personalisation values.

Subgraph	**#Nodes**	**#Relations**	**#Edges**	**Density**
2-hops	64,368	214	153,852	7.42e-05
3-hops	101,158	410	265,910	5.19e-05
5-hops	217,334	497	642,633	2.72e-05
PageRank	80,181	399	187,491	5.83e-05
PPR($\alpha_{per} = 0.9$)	68,004	321	162,322	7.02e-05
PPR($\alpha_{per} = 0.7$)	69,012	333	165,124	6.93e-05

Table 2: Distribution (in %) of the distance of target nodes from the source nodes in different subgraphs. All the subgraphs show similar distributions with probability of a target node being unreachable close to 0.18.

Subgraph	0	1	2	3	4	5	≥ 6	Unreachable
2-hops	9.82	4.64	21.11	14.68	29.94	1.15	0.47	18.16
3-hops	9.77	4.62	21.04	15.27	29.45	1.39	0.43	18.01
5-hops	9.70	4.58	20.92	15.52	29.38	1.67	0.28	17.92
PageRank	9.83	4.66	21.18	14.35	28.44	2.30	0.77	18.44
PPR($\alpha_{per} = 0.9$)	9.82	4.64	21.12	14.71	29.63	1.47	0.37	18.20
PPR($\alpha_{per} = 0.7$)	9.82	4.64	21.12	14.71	29.86	1.23	0.39	18.19

the relational information in the subgraphs. When constructing the subgraph using the N-hop method, we include all the edges and nodes reachable from the seed set by paths of length N. We consider cases when N=2, 3 and 5 starting from the initial set of seed nodes.

On the other hand, the construction of subgraphs using PageRank is a two-step process. An initial subgraph is constructed by considering 3-hop neighbours of the seed set. PageRank scores of the subgraph nodes are computed and the top-*k* nodes with the highest PageRank scores are considered for the second step. To construct the final subgraph, we consider the top-*k* nodes with respect to the PageRank values along with the movie nodes. We take the set of nodes thus constructed and their 1-hop neighbours to present the final subgraph. Since the random walk in PageRank can start from any random node, we use the Personalised PageRank (PPR) (Bahmani et al., 2010) to make the subgraphs more movie-centric. We distribute high personalisation scores to the movie entities so that the random walk starts with high probability from the movie nodes. In our experiments, we distribute α_{per} between the set of movie nodes and $1 - \alpha_{per}$ among the rest of the nodes in the subgraph extracted after the first step. To construct such movie-centric graphs, we choose $\alpha_{per} = 0.7, 0.9$. Table 1 shows the graph properties of different subgraphs constructed. The 5-hop subgraph contains the highest number of nodes and edges while having the lowest density. We learn entity embeddings for these subgraphs constructed for use in the model.

4.2 Entity Embeddings

To incorporate information from the relational graph into our model, we need to represent the entities as a d-dimensional vector. More specifically, each entity is represented as a $\mathbf{R^d}$ vector. In this paper, the embedding of an entity $\mathbf{i}$ is represented by $\mathbf{e_i} \in \mathbf{R^d}$. Balažević et al. (2019) learn entity embeddings by solving link-prediction tasks in relational graphs. Since those embeddings are well suited for the task of link-prediction, they contain relevant information for predicting new relations in an incomplete graph. Connectionist models such as demonstrated by Schlichtkrull et al. (2018), leverage the local neighbourhood of nodes to learn nodal embeddings. Although relational graph neural networks help in learning better entity representation for relational graphs, KGs do not contain all factual relations. As shown in Table 2, for all of the subgraphs, at least 18% of the target nodes are unreachable from the previously mentioned entities. Therefore, we choose the above-mentioned method for learning entity embeddings.

4.2.1 TuckER Embeddings

Graph Neural Networks (Wu et al., 2020) allow the information to flow through a graph using connectionist models. However, such models do not allow information flow between nodes that are not connected through paths of any length. The nature of the learned embeddings is such that it helps in link prediction between two entities. Since, in our case, many target entities are not connected to source nodes, we use embeddings learned through Tucker decomposition (Tucker, 1966) for better entity prediction.

Tucker decomposition decomposes a tensor into a smaller core tensor and several smaller matrices. Balažević et al. (2019) show that Tucker decomposition can be used to learn entity embeddings from relational graphs. They also showed that embeddings learned using this technique outperforms techniques such as TransE (Bordes et al., 2013) or DistMult (Yang et al., 2015) on link prediction task over multiple datasets. Tucker decomposition for a tensor $\boldsymbol{X} \in \mathbf{R}^{I \times J \times K}$, gives tensor $\boldsymbol{Z} \in \mathbf{R}^{P \times Q \times R}$ and matrices $\boldsymbol{A} \in \mathbf{R}^{I \times P}$,$\boldsymbol{B} \in \mathbf{R}^{J \times Q}$, and $\boldsymbol{C} \in \mathbf{R}^{K \times R}$ as outputs.

$$\boldsymbol{X} \approx \boldsymbol{Z} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C} \tag{1}$$

$$\phi(e_s, r, e_o) = \boldsymbol{Z} \times_1 \boldsymbol{e_s} \times_2 \boldsymbol{w_r} \times_3 \boldsymbol{e_o} \tag{2}$$

For link prediction tasks, $\mathbf{X}$ is the adjacency tensor, consisting of the adjacency matrices for each relation in the KG, with `1` at index (s, r, o) if a relationship exists between e_s and e_o on relation r, else `0`. We set $A = C$, to obtain the same entity embeddings in both the source and object positions. The core tensor $\mathbf{Z}$ is the set of parameters that is linearly proportional to d_e and d_r, the entity embedding size and the relation embedding size respectively and $\mathbf{Z} \in \mathbf{R}^{d_e \times d_r \times d_e}$ space. Our method learns the entity embedding matrix $\mathbf{A} \in \mathbf{R}^{n_e \times d_e}$, relation embedding matrix $\mathbf{B} \in \mathbf{R}^{n_r \times d_r}$ space as shown in Equation 2, where $\mathbf{e}_s$, $\mathbf{e}_o \in \mathbf{R}^{n_e \times d_e}$ are the corresponding rows of A and $\mathbf{w}_r$ is the corresponding row of B. The model is trained to minimise the Bernoulli negative log-likelihood loss function which maximises the estimated probability of a relation $\phi(e_s, r, e_o)$ over known relations.

4.2.2 Positional Embeddings

During the dialogue flow, entities are mentioned sequentially. By virtue of this, entities mentioned later have a larger effect on future recommendations. Wu et al. (2019) constructed a session graph to represent the sequential nature of items belonging to a user. By leveraging the properties of such graphs, they learned latent item embeddings using graph neural networks. To avoid the construction of session graph and increasing the complexity of the model, we use positional embeddings as described in Devlin et al. (2019), which allow us to infuse sequence information into the entity embeddings. The pattern of embeddings induced by Equation 3 and 4 encodes positional information of entities in the model.

$$\mathbf{POS_{(pos,2i+1)}} = \cos(pos/10000^{2i/d_{model}}) \tag{3}$$

$$\mathbf{POS_{(pos,2i)}} = \sin(pos/10000^{2i/d_{model}}) \tag{4}$$

The subscript **pos** in Equation 3 and Equation 4 refers to the index position of the entity in the dialogue, while d_{model} refers to the dimensionality of the entity embeddings. The subscript **i** refers to the index in the vector representing the positional embedding. The final entity representation of the entity $\mathbf{e_i}$ seen at position $\mathbf{pos}$, given the scaling factor β, is given by:

$$\mathbf{e_{i,pos}} = \mathbf{e_i} + \mathbf{POS_{pos}}/\beta \tag{5}$$

4.3 User representation

A user is represented by a set of entities already encountered in the dialogue. If a user has interacted with $\mathbf{n}$ entities, the user representation is represented by the matrix $\mathbf{U} \in \mathbf{R^{nxd}}$. We then map the matrix $\mathbf{U}$ to a vector in $\mathbf{R^d}$ space to find the similarity of the user with different entities. We borrow a part of the soft-attention used by Wu et al. (2019) to construct a representation of the user in the $\mathbf{R^d}$ space. We can represent the user by choosing the embedding of the last-seen item. We use soft-attention to better represent the user.

$$\alpha_i = \mathbf{q}^\top \sigma(\mathbf{W_1 U_n} + \mathbf{W_2 U_i} + \mathbf{c}) \tag{6}$$

$$\mathbf{u} = \sum_{\mathbf{i=1}}^{\mathbf{n}} \alpha_\mathbf{i} \mathbf{U_i} \tag{7}$$

Here, $\mathbf{U_n}$, $\mathbf{U_i}$ are the representation of the user when $\mathbf{n}$, $\mathbf{i}$ entities have been encountered respectively. $\mathbf{q} \in \mathbf{R^d}$ and $\mathbf{W1}, \mathbf{W2} \in \mathbf{R^{dxd}}$ are trainable parameters.

4.4 Similarity score of use with products

The final similarity scores of users with the movie entities is calculated by computing the dot product of the user representation with the movie entity representations passed through a softmax layer.

$$\mathbf{P_{rec}} = softmax(\mathbf{u} \bigodot \mathbf{E^T}) \tag{8}$$

We optimise our model by using the cross-entropy loss on the set of recommended movies.

5 Experimental settings

This section presents the parameters to train our system and the evaluation metrics used to evaluate our approach. Furthermore, we describe previous approaches, against which we compare our proposed approach[1].

5.1 Model settings

For learning the entity embeddings using Tucker decomposition, we set the learning rate = 0.0001, batch size = 1,536 and epochs = 512 for all of the subgraphs. We set the dropout values of all the layers to be 0.1 for every layer. For the recommendation engine, we set the batch size to be 32, and the learning rate to be 0.001 to avoid overfitting of the model. Similar to Chen et al. (2019), we use the Adam optimiser (Kingma and Ba, 2014) with β_1 = 0.9, β_2 = 0.999 and ϵ = 10^{-8} to optimise our model. We set the value of the scaling factor β to 1,000. We train the entity recommendation model for 1,000 epochs with a validation patience of 5.

5.2 Evaluation Metrics

As our system focuses on improving the performance of the recommendation engine, we select the information retrieval metrics Mean Reciprocal Rank (MRR) and Recall@k as our metrics. MRR is calculated as the mean of the reciprocal of the rank of the target entities as predicted by the model. Recall@k measures if the target movie is present in the top-k recommended products.

[1]The codes and the models are available at https://github.com/rajbsk/KG-conv-rec

Table 3: Performance of models on recall@1, recall@10, recall@50 and MRR. The confidence intervals were calculated by conducting Student's t-test on 24 runs of each of the models ($p \ll 0.001$).

Model	**R@1**	**R@10**	**R@50**	**MRR**
Li et al. (2018)	2.23±0.24	13.14±0.63	29.22±0.77	0.060±0.0018
Chen et al. (2019)	3.00±0.20	16.30±0.30	33.80±0.70	0.066±0.0019
2-hops	3.26±0.13	17.15±0.52	34.35±0.58	0.078±0.0014
3-hops	**3.39**±0.15	18.00±0.40	35.33±0.35	0.082±0.0018
5-hops	3.36±0.14	**18.05**±0.45	**35.70**±0.37	**0.082**±0.0016
PageRank	3.28±0.13	17.53±0.46	35.20±0.39	0.080±0.0019
PPR($\alpha_{per} = 0.9$)	3.29±0.16	17.35±0.33	35.20±0.30	0.079±0.0011
PPR($\alpha_{per} = 0.7$)	3.31±0.17	17.21±0.43	34.28±0.48	0.079±0.0017

5.3 Comparison to previous approaches

We consider KBRD (Chen et al., 2019) as the baseline for our work. The KBRD system extracts different movie entities and mentioned entities from the dialogues and links the entities to DBpedia using the entity linking technique described by Daiber et al. (2013). The system then extracts a subgraph by considering 2-hop neighbours of movie entities. The entity embeddings are learned by using Relational Graph Convolutional Network (Schlichtkrull et al., 2018). The entity embeddings belonging to a particular user are then passed to a self-attention layer to learn the user-representation. The dot-product of the user representation is computed with the entity embeddings and the scores are passed through a softmax layer to get the recommendation probabilities.

We also compare the performance of our model with ReDial (Li et al., 2018). ReDial is a CRS with a dialogue generation component built using HRED (Sordoni et al., 2015). The recommendation system makes use of an autoencoder model and sentiment analysis model to suggest movies to users.

6 Results and Discussions

First, we perform a quantitative evaluation of the models. We then analyse the effect of different subgraphs on the performance. Thereafter, we proceed towards studying the performance while recommending disconnected entities and the effect of session sequence length. Finally, we analyse two examples, where the model results are not consistent with the gold-standard, to get insights for future work.

6.1 Quantitative Evaluation

In this work, we put forward the hypothesis that the use of pre-trained embeddings of entities supplemented with positional information improves the performance of the recommendation engine. The results present in Table 3 show the performance of different models. To compute the performance of each model, we did 24 runs of our models to obtain the distribution of the performance scores on a different metric. We compute the confidence interval for the performance of each model on a different metric by using the Student's-t distribution (Student, 1908). Two sample t-test showed that our results are statistically significant results over the previous models ($p \ll 0.001$). We achieve a performance of 3.36%, 18.05%, 35.70%, and 0.082 on the recall@1, recall@10, recall@50, and MRR metrics respectively with 5-hops subgraph. We improve upon the previous baseline by (Chen et al., 2019) by 12.00%, 10.73%, 5.32%, and 24.24% on the four metrics respectively. These results show the efficacy of our choice of using subgraphs and pre-trained embeddings.

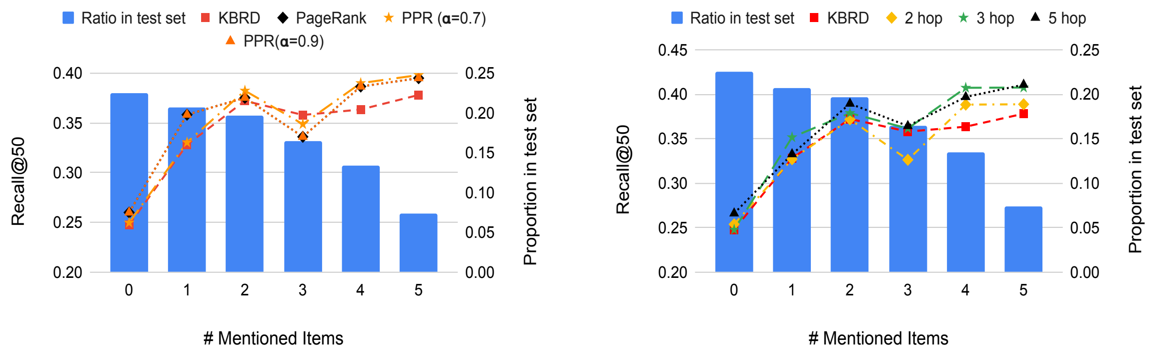
6.2 Performance Analysis

This section presents the performance of our model in different settings of the problem. It also presents an error analysis to understand the performance of the model.

Table 4: Evaluation on disconnected entities. We report the performance values on different metrics for the instances when the target entity is disconnected from the already seen set of entities in the subgraphs.

Subgraph	Recall@1	Recall@10	Recall@50	MRR
Chen et al. (2019)	1.41	12.01	34.28	0.051
2-hops	**2.23**	**12.84**	**36.75**	**0.060**

Figure 2: Performance of different models on Recall@50 with the number of items. As the number of items increase, our models have better performance compared to the baseline. Also, as the number of nodes and edges increase in N-hop, the performance improves. PageRank subgraph models perform similar when the number of items increase.



6.2.1 Effect of different subgraphs

Graphs with a higher number of nodes and edges contain more information. However, in the recommendation scenario, it is desirable to have subgraphs from KGs containing the maximum amount of information with a minimal amount of noise. Table 3 shows that for models with subgraphs constructed using the N-hop method, as the value of N increases, the performance of the model improves. However, the model performance when N=5 is not significantly better than the case when N=3. This points us towards the direction of analysing the performance of the models when using subgraphs constructed using PageRank. The results show that even though subgraphs constructed using PageRank or personalised PageRank contain a lower number of nodes and edges (Table 1), they perform at par with N-hop models. Since PageRank identifies important nodes in a graph, PageRank graphs are subgraphs of 5-hops graphs containing a lower number of nodes and edges as well as a lesser amount of noise. Table 3 further shows that giving more importance to movie nodes while constructing PageRank subgraphs gives better performance. When giving more importance to movie nodes, the PageRank algorithm constructs a subgraph focusing on the movie nodes. This method can be used to adapt easily to different domains. When pre-training entity embeddings using the method described by Balažević et al. (2019), the runtime is directly proportional to the number of entities and relations. With smaller and richer subgraph, the pre-training is faster while giving us comparable performance.

6.2.2 Recommending Disconnected Entities

Disconnected target entities are those entities that are not connected through a path of any length in the KG to entities that have already interacted with the user. We compare 2-hop model against KBRD. The two models make use of the same DBpedia subgraph and thereby have the same set of disconnected entities. We do not compare KBRD against our best performing model since the subgraph would change for 5-hop model. Similarly, we do not compare against ReDial as it uses auxiliary information such as sentiment information which neither KBRD nor our model uses. While ReDial uses auxiliary information, neither KBRD nor our model leverages such information apart from the information through KGs, our models can be adapted into the domain more quickly as compared to ReDial. Since the ReDial model

Table 5: Examples where our model produces inconsistent results. We take 3 cases from the test set to study the errors. The recommended movie in Dialogue 1 occurs in the fourth position and not in the first. For Dialogue 2 and 3, our model gives out the same set of recommendation for the first four movies even though the context is different.

Dialogue 1	Dialogue 2	Dialogue 3
Rec: Hello there! User: Hi. Rec: Tell me what kind of movies would you like to watch? User: I like all types of movies. Especially comedy and family Rec: I can recommend you Get Out	User: I would like to watch any movie. User: Tell me any movie. User: Like Avengers: Infinity War. Rec: Have you seen Scary Movie?	Rec: Tell me what would you like to watch? Rec: I just watched Avengers: Infinity War Rec: Have you seen it? Rec: or do you like Scary movies? Rec: have you ever seen Click? User: I like a lot of different movies. Thinking about some comedies something like Billy Madison Rec: have you ever seen You Don't Mess with the Zohan?

Target Movie	1st	2nd	3rd	4th
Get Out	Black Panther	Jumanji	It	Get Out
Scary Movie	Avengers: Infinity War	Black Panther	Thor: Ragnarok	Wonder Woman
You don't mess with the Zohan	Avengers: Infinity War	Black Panther	Thor: Ragnarok	Wonder Woman

does not make use of KGs, there is no scenario of disconnected entities.

Table 4 shows the comparison of our model against KBRD. Our model performs better than KBRD in all four metrics. The results show the correctness of our assumption that using pre-trained embeddings are more helpful in improving the performance of recommendation models using KGs.

6.2.3 Effect of Session Sequence Length

We consider the session sequence length as the number of entities encountered by the user. We analyse the performance of our model on different subgraphs against the baseline. We compare the performance of different models on Recall@50 and is shown in Figure 2. The figures show that as the number of mentioned items increases, the performance of the model improves. This is attributed to the fact that when the number of mentioned items increases, the model has more information about the user and thereby represents the user better. Figure 2 also shows that our model on different subgraphs outperforms the baseline. In addition to that, embeddings learned using graphs with a higher number of entities and relationships demonstrate better performance. This can be attributed to the fact that a higher number of entities and relations help in learning more information and thereby better entity representations.

6.2.4 Error Analysis

Table 5 displays three dialogues where our model does not perform correctly. In the case of Dialogue 1, the correct target movie is *Get Out*. However, our model assigns it the fourth highest probability. This can be explained by the fact that when no initial entities are mentioned in the dialogue, the model is not able to infer the context of the conversation. As of result of this, the model gives the same output irrespective of the context.

For Dialogues 2 and 3, the model gives the same result for both of the dialogues even although the context and the mentioned entities are different. Both dialogues have the movie in common and the results are biased towards that movie. The model does not discriminate the movies present in the user utterance or the recommendation utterance.

7 Conclusion

In this work, we propose a new model to improve the performance of the recommendation engine in CRSs. We introduce different subgraphs for the task to show the need for the construction of better subgraphs from KGs to propagate information into recommendation models. We show that our model outperforms the current-state-of-the-art on multiple metrics by considering the use of pre-trained KG embeddings and positional embeddings. Through multiple experiments, we demonstrate that our model performs better in recommending disconnected entities in KG. Also, as the number of entities mentioned in the text increases, the performance of our model improves.

In our future work, we plan towards incorporating the context of the dialogue text into the model. We will be working towards understanding the differences between the movies mentioned by the user and the recommender, resulting in a better understanding of the context which in turn can be used to produce better recommendations. Exploration of the performance of different KGs for this task is left for future work.

8 Acknowledgements

This work is supported in part by Government of Ireland Postgraduate Fellowship, Irish Research Council under project ID GOIPG/2019/3480. The work is co-supported by Science Foundation Ireland under grant number SFI/12/RC/2289 2 (Insight), co-funded by the European Regional Development Fund and by the Irish Research Council, under grant number IRCLA/2017/129. We would like to thank Mr Tarek Zaarour, Dr Theodorus Fransen and Ms Omnia Zayed for their valuable comments and suggestions towards improving our paper. We would also like to thank the anonymous reviewers for their insights on this work.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast incremental and personalized PageRank. *Proceedings of the VLDB Endowment*, 4(3):173–184.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Empirical Methods in Natural Language Processing*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1803–1813.

Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. 2018. Q&R: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 139–148.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2020. A Survey on Conversational Recommender Systems. *arXiv preprint arXiv:2004.00646*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems*, pages 9725–9735.

Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2020. Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-Start Users. *arXiv preprint arXiv:2005.12979*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Elaine Rich. 1979. User modeling via stereotypes. *Cognitive science*, 3(4):329–354.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562.

Student. 1908. The probable error of a mean. *Biometrika*, pages 1–25.

Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 297–305.

Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958.

Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38.