A multiturn recommender system with explanations

Luan Soares de Souza
Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, Brazil
luanssouza@usp.br

André Levi Zanon
Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, Brazil
andrezanon@usp.br

Lucas Padilha Modesto de Araújo
Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, Brazil
padilha.lucas@usp.br

Marcelo Garcia Manzato
Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, Brazil
mmanzato@icmc.usp.br

Abstract

Recommendations engines use interactions between users and items to predict the preferences of the users and generate recommendations for them. However, because they rely on historical data, the user's interest at the moment may not be captured. In this context, Conversational Recommender Systems (CRSs) have been proposed in order to provide recommendations that provide suggestions based on the user's current interests by eliciting information from the user in turns in which the system can ask for the user to understand more about the current interest in the time or recommend. In that regard, we propose CRSs for cold-start users as a Knowledge Graph search. The system also employs a Natural Language Processing module to explain the recommendations, bringing transparency to the recommendation algorithm.

Keywords: Conversational Recommender Systems, Explainable Recommendation, Multi-turn Recommendation

1 Introduction

Recommender Systems (RSs) provide suggestions to users by finding relations between underlying similarities between interactions [1]. However, despite the success of methods such as factorization machines and deep neural networks to provide recommendations, these recommendation engines are 'static', meaning that they are trained offline on historical behavior [4]. As a result, they fall short regarding users' current needs since historical data may be noisy considering the real user interests in items [2].

To overcome such problems, Conversation Recommender Systems (CRSs) have been defined as eliciting user information dynamically and taking actions in real-time multi-turn interactions using natural language, to achieve recommendation related goals [2, 3, 9]. Therefore, CRSs are based on turns, in which the system chooses whether to ask the user for information about items of the user's preference or to recommend based on the system's current knowledge of the user's information.

Current approaches to CRSs are modeled based on the tasks of (1) deciding if the system should ask or recommend, (2) ranking properties to know user preferences, and (3) ranking items to recommend to users [2, 5–8, 11]. Nevertheless, there are still some open issues regarding side information used in order to ask questions to users and recommendations [2, 3]. Furthermore, most works rely on pre-trained data to generate recommendations, which may not be the case for cold start, in which no information is known about the users.

Therefore, we propose a multiturn recommender system based on Linked Open Data (LOD) to leverage contextual information from cold-start users such as the genre, actors, and directors of movies the user is interested in at the moment, to recommend items as Knowledge Graph (KG) interactive search. Considering these three tasks of conversational recommender systems discussed in this section, in task (1) we used a Multi-Armed Bandit (MAB) algorithm to adaptatively choose between asking about new information or recommending. In task (2) we used a scoring metric based on the KG that considers the relevance of the property in the total KG and based on the properties previously chosen, and, in task (3) we used the Personalized PageRank algorithm as a recommendation algorithm. When a recommendation is made, the user may need additional information to make decisions [12]. To provide supporting information about items in the decision making process, our proposal also offers explanations.

The manuscript is organized as follows: in section 2, we explain the recommendation engine and how each of the three tasks of CRSs was developed algorithmically and how the explanations are generated. Then, in section 3 the architecture of the deployed chatbot is described. Finally, section 4 addresses conclusions and future works.

In: XXII Workshop de Ferramentas e Aplicações (WFA 2023), Ribeirão Preto, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2023.

ISSN 2596-1683

2 Recommendation Engine

The proposed multiturn recommender system uses LOD to generate recommendations to cold-start users by building a profile of properties through turns. At every turns the system is faced with a decision: to ask about characteristics that the user is searching for in a movie or to recommend. Therefore, the system has three main tasks: (1) rank properties relevant to the user based on previous interactions to ask for user opinion; (2) provide recommendations based on the movie characteristics that the user liked; and (3) decide whether the system should recommend or ask.

Considering task (1), given a property (e.g. genre, actor, music composer, director), and a value (e.g. drama, Viola Davis, Bill Conti, Quentin Tarantino), of the Wikidata[1] KG, the system offers ranked properties and asks the users about their preferences.

As the interaction starts, the user informs his/her age and selects a property/value tuple of interest. Based on these pieces of information, our algorithm reduces the original graph into a subgraph with the movies that contain the initial tuple chosen and are adequate to the user age, in order to reduce the search space. Then, the system ranks the property/value tuples of movies that had the previously chosen tuple considering: the entropy of the properties; the relevance of values in the current subgraph; and the relevance of values in the full KG, as in Equation 1.

$$score(p, value) = \alpha * zscore(entropy(subgraph, p)) + \beta * zscore(pagerank(subgraph, value)) + \gamma * zscore(pagerank(fullgraph, value)) \quad (1)$$

Each of the three terms in Equation 1 represents the relevance of the property in the current subgraph, the local relevance of the value considering the current subgraph, and the global relevance considering the entire graph, respectively. Therefore, the first term is the entropy of the property (actor, director, genre, etc) p in the current graph ($subgraph$). The second term is responsible to compute the local relevance of the value ($value$) based on the Personalized PageRank of the current graph ($subgraph$) with 80% of the weight to movies and values that the user liked and 20% to the rest of the nodes. The final term is the global relevance based on the Non-Personalized PageRank of the full Wikidata graph. All the terms are normalized with a z-score and weighted in order to transpose them into the same interval from 0 to 1. The weights set were 0.33 for each term. To choose a movie to recommend (task (2)) the same Personalized PageRank is used with the same 80% of the weight to movies and values that the user chose interacting in task (1) and 20% to the rest of the nodes.

If the user chooses a property/value tuple, then the current graph is reduced to a subgraph with the movies connected to the chosen tuple and all the other property/value tuples related to those movies. The user can also not like any of the provided tuples, and as a result, the current graph is reduced by removing all the property/value tuples shown to the user. Finally, in task (3), the system chooses between asking for more preferences from the user and making a recommendation based on the interactions, with a Multi-Armed Bandit (MAB) algorithm.

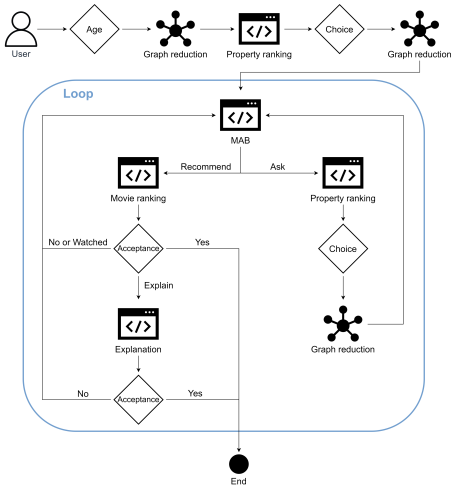


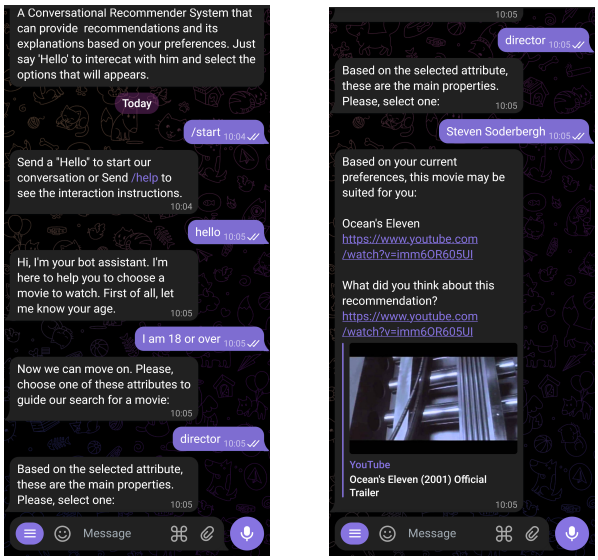
Figure 1. Aplication conversational flow.

Because MAB algorithms can learn in an online setting, if the user is constantly choosing a property, MAB approaches can use this implicit signal as positive feedback to continue to ask for more information, since the system is constantly displaying relevant property/value tuples. In contrast, if the users does not like any of the properties displayed, the system uses the user's negative feedback. It starts recommending movies that have the information provided in past turns. When the system recommends a movie that the user has already watched and liked, then the system considers it as positive feedback to the system to recommend another movie, since a relevant recommendation, despite repeated, was provided. Otherwise, if the user did not like the recommendation, then the system uses this as a negative reward to continue asking for property/value tuples in order to understand more about the user and build the profile.

Questions about the preferred properties are made to the users until the system has no other properties to ask, about and only movies to recommend or the user can also ask for a recommendation. The recommendation process finishes when the user accepts the suggestion or when there are no more movies with the properties informed by the user. Figure 1 displays the conversational loop in which diamonds represent user input, the graph icon a reduction on the graph, and the algorithm icon represents the Explanation algorithm, the MAB algorithm and the PageRank algorithm that will rank movies and properties in order to recommend an item or aks the user about a tuple.

[1] https://www.wikidata.org/

In the beginning, the user informs the age and only the movies rated above the informed age are maintained, then the system ranks the properties of the graph and displays to the user, that chooses the favorite, reducing the graph again. The interaction loop starts with the MAB algorithm choosing between displaying properties or recommending. If it displays properties the graph is reduced considering the user choice, by maintaining only the items with the properties previously chosen. If it shows a recommendation, two outcomes can occur, the user can accept the recommendation, ending the conversation with the system, or it can inform that the movie was already watched and then the loop continues with the MAB choosing between recommending or asking a question to the user. An example of the recommendation process is shown in Figure 2.



(a) Interaction 1. **(b)** Interaction 2.

Figure 2. System interaction.

When a recommendation is made, the system offers an option to the user for an explanation to the reason why the movie suggested is appropriate. These explanations are generated has two main steps: (1) preprocessing of movies' reviews; and (2) real-time explanation generation.

Our explanations are aspect-based extractive summaries that use positive sentences from reviews to describe a recommendation. To enables, that, we have a preprocessing phase (1) to prepare the reviews to real-time explanations. Our preprocessing steps, applies a Part-Of-Speech Tagging to identify possible aspects. After that, the sentiments and embeddings of each aspected are extracted with pretrained models. Only positive aspects candidates are considered in to filter the sentences that can compose the summary. After this process, each movie has its own aspects and sentences that we use to generate the explanation.

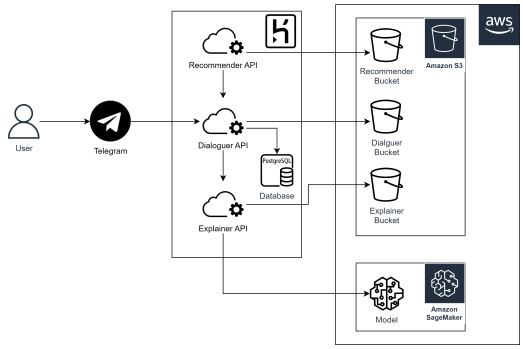


Figure 3. Aplication architecture.

In the real-time explanation generation step (2), the embeddings of the properties chosen during the interaction are extracted and then used as a centroid to generate our summary. Therefore, the sentences with the embeddings most similar to the embedding of the properties appear in the explanation. After reading the explanation, the user can accept or reject the recommendation if he/she wants to continue with its interaction.

3 Architecture

Our architecture is composed of three main modules: 1) Dialoguer[2]; 2) Recommender[3]; and 3) Explainer[4]. The Dialoguer is in charge of getting messages from the users and deciding what to do. For instance, if a user sends a message to start a conversation, the Dialoguer will call the Recommender to start a multi-turn recommendation process. The Recommender is responsible for generating recommendations based on the user's responses. Besides, the Recommender reduces the graph and decides when to make a recommendation or to ask for more information. The MAB algorithm chosen to decide when the system should ask about properties or recommend was the Thompson Sampling algorithm [10]. The Explainer, if the user requires it, will provide a post-hoc explanation for the recommendation using the properties that the user prefers as a reference. An architecture overview of our application is presented in the Figure 3.

The core of our recommendation process is the Recommender, which is responsible for recommending and deciding when to recommend or to ask for more preferences. This module was developed with Python and hosted on the Heroku Cloud Platform[5]. The Properties used in the recommendation process were extracted from WikiData and the states of the KG is persisted in a bucket in the Amazon S3[6] after each interaction.

[2]https://github.com/luanssouza/crs-api-dialoguer
[3]https://github.com/luanssouza/crs-api-recommender
[4]https://github.com/luanssouza/crs-api-explainer
[5]https://www.heroku.com/
[6]https://aws.amazon.com/s3/

The Dialoguer module is hosted on Heroku Cloud Platform and is in charge of receiving a message from the users, processing it, making requests to other services, and then answering requests from users, besides controlling the interaction turn of the chats. Some interactions are persisted in a bucket and others in a PostgresSQL[7] database.

The Explainer receives properties of the user profile and generates summaries based on them to explain a recommendation. It is hosted on Heroku Cloud Platform with a bucket in the Amazon S3 with filtered sentences. To get the properties embeddings, the explainer uses a pretrained model hosted in the Amazaon SageMaker[8].

Dialogs with a users are based on the interactions presented in Figure 1. The first kind of interaction is the greeting, when the user sends a message to start a conversation. Then, the Dialoguer asks the user about his age range. As the user answers, the Dialoguer starts the interaction with the Recommender, informing the age range of the user and receiving a list of properties. These properties are presented to the user, who is asked to select the one they prefer. Based on the user response, the Dialoguer informs the preferred property of the user to the Recommender, which shrinks the graphs of properties and ranks those that remain. These ranked properties are presented to the users by the Dialoguer and they are requested to select the preferred one. In this step, which repeats until the Recommender decides to make a recommendation or the user requests for one, the user can browse for properties to find one that interests them. When a recommendation is made, the Dialoguer searches for a trailer of the movie in the YouTube API[9] and presents it to the user. In this stage, the user can request for an explanation provided by the Explainer or inform them if they will take the suggestion or not. If the user doesn't accept the recommendation, more properties are presented to the users and all the following steps repeat until the user accepts a suggestion.

4 Conclusion

In this paper we described an explainable movie CRS for cold-start users using KG and NLP to provide explanations to users. Our source code is available under the MIT license for academic and social use.

The CRS initially filters movies that are rated below the user's age, then it initially asks the user for the initial property that the user is interested in to start the conversational loop. In this step, a MAB algorithm chooses whether the system should ask for more properties or recommend a movie. If the system asks for a property then the graph is limited only to the movies that contain the current and previous properties that the user has chosen in addition to the other properties that these movies have. If the system recommends a movie that the user already watched then the system begins the flow by choosing to ask or recommend again. Finally, if the user accepts the recommendation, then the loop is finished. Every time that a recommended movie is shown, the user can also choose to see an explanation that provides additional information about the recommendation.

As future work, we intend on testing different recommendation and reinforcement learning algorithms that can be trained with offline data obtained in order to compare with the cold-start approaches.

[7]https://www.postgresql.org/

[8]https://aws.amazon.com/sagemaker/

[9]https://developers.google.com/youtube/v3

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors would like to thank the financial support from FAPESP, process number 2022/07016-9.

References

[1] Charu C Aggarwal et al. 2016. *Recommender systems*. Vol. 1. Springer.

[2] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and challenges in conversational recommender systems: A survey. *AI Open* 2 (2021), 100–126.

[3] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.

[4] Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2020. Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2425–2428.

[5] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 304–312.

[6] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2073–2083.

[7] Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. 2020. Latent linear critiquing for conversational recommender systems. In *Proceedings of The Web Conference 2020*. 2535–2541.

[8] Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. 2018. Preference elicitation as an optimization problem. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 172–180.

[9] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval*. 235–244.

[10] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3-4 (1933), 285–294.

[11] Ivan Vendrov, Tyler Lu, Qingqing Huang, and Craig Boutilier. 2020. Gradient-based optimization for bayesian preference elicitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 10292–10301.

[12] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.