Healthy Food Recommendation and Explanation Generation using a Semantically-Enabled Framework*

Sola S. Shirai[1][0000−0001−6913−3598], Oshani Seneviratne[1][0000−0001−8518−917X], Ching-Hua Chen[2][0000−0002−1020−0861], Daniel M. Gruen[1][0000−0003−0155−3777], and Deborah L. McGuinness[1][0000−0001−7037−4567]

[1] Rensselear Polytechnic Institute, Troy NY 12180, USA
{shiras2, senevo, gruend2}@rpi.edu, dlm@cs.rpi.edu
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
chinghua@us.ibm.com

Abstract. Recommender systems are an important service utilized in a wide variety of applications, but they rarely explain the processes and data used to generate the final recommendation. Furthermore, no convenient software resource exists to help developers create recommender systems that use a variety of strategies, such as knowledge-driven recommendation processes. We present a Python framework to support the development of recommender systems with an emphasis on using data sources containing rich semantics and providing explanations for each step involved in producing the recommendation. We illustrate this through an example food recommendation system developed using the framework.
Submission Type: Poster
Resource Link: https://tetherless-world.github.io/FREx

1 Introduction

Recommender systems, which aim to predict preferences and suggest suitable items to users, have become a standard tool in many application domains. Various recommendation techniques have been developed, and in recent years there has been an increasing amount of attention given to factors like the explainability of recommender systems [6]. However, knowledge-driven or rule-driven processes involved with recommendations have typically not received much attention.

While there are many software tools available to implement algorithms for computing various recommendation scores [5], such tools are not typically intended to encompass the entire recommendation process due to the many domain- and application-specific aspects that may need to be considered in a recommendation system. Naturally, there is no one-size-fits-all solution that can be automatically applied to every possible application and domain. Therefore, we believe

that the community can benefit from a tool that can help provide the groundwork for the typical processes involved in recommender systems and integrate explanations into the system from the ground up.

In this paper, we present preliminary work for FREx (a **F**ramework for **R**ecommendations with **Ex**planations), which provides a framework for developing knowledge-driven recommender systems that integrate explainability into the end-to-end process of the recommender system. Our framework is intended to best serve applications that utilize a source of well-structured semantics, such as an ontology, to provide domain models of the various items, users, and other entities involved in the application. Additionally, we apply our framework to implement an application for healthy food recommendation, to demonstrate its utility and to potentially serve as pedagogical material.

2 FREx Framework

FREx is a Python framework aimed at enabling effective prototyping of knowledge-driven recommendation systems with a focus on integrating explanations. FREx provides extensible classes that can be developed and integrated into a cohesive system for recommending items and providing explanations. At the core of FREx is a pipeline architecture for processing a series of steps to provide recommendations. This general architecture is shown in Figure 1.

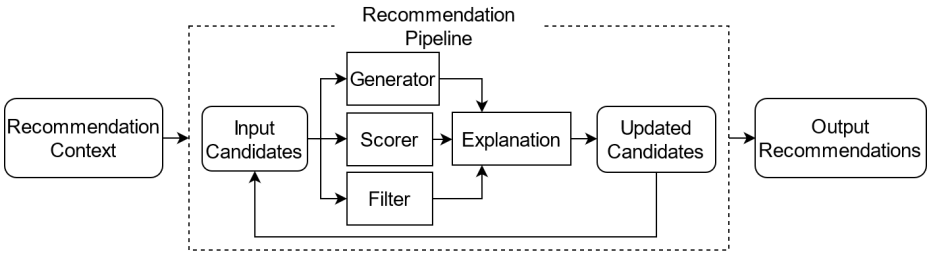

Fig. 1. An overview of the overall pipeline used to process candidates and produce recommendations from the system with FREx.

FREx's pipeline for recommendations is constructed of `PipelineStage`s. In a recommender system, three key stages that are typically carried out are to (1) generate, (2) score, and (3) filter candidate items to recommend. In FREx, the skeleton of each of these three stages is implemented – as the `CandidateGenerator`, `CandidateScorer`, and `CandidateFilterer` classes – and handles the creation or modification candidates as well as attaching explanations associated with the stage. Developers can extend these base classes to provide domain-specific functionality and supply customized explanations.

As candidates pass through the pipeline, they retain information about the context in which they were produced, the recommendation scores applied by scor-

ing stages, and the explanations applied by all stages. Precisely what constitutes "context" (for which various definitions exist [1,2]) will be entirely dependent on the application that is being implemented. Scores that are applied may be hand-crafted heuristics related to some knowledge-driven process, or they may be the result of other recommendation algorithms. Explanations are supplied by each pipeline stage, so the final candidates that are output by the pipeline will have a record of all explanations applied by the entire pipeline.

In order to make meaningful use of the data in a domain specific, knowledge-driven recommendation system, it is important to model the semantics of objects in the domain. FREx has been developed with the intention of enabling the use of Resource Description Framework (RDF) data with rich semantics to model domain objects and their properties. To facilitate this, Python dataclasses for the application are implemented that reflect the data models of the underlying ontology, supporting more object-oriented program design.

3 FoodRec System

Using FREx, we have developed an exemplar application, FoodRec, for recommending healthy foods to users. FoodRec incorporates knowledge into the recommendation process by (1) considering the classifications of ingredients and (2) utilizing guidelines to determine what is considered a "healthy" recipe. In the recommender, we leverage semantic information that is available through FoodOn's class hierarchy to infer additional knowledge about preferences and restrictions. For example, if a user is prohibited from eating pork, we can identify subclasses of pork in FoodOn to infer further restricted ingredients, like "bacon" or "ham." Similarly, we can infer more general preferences based on specific ingredients that are included in a user's favorite recipes (e.g., a user likes recipes that use bacon, so we can infer a preference for pork in general).

The FoodRec use case utilizes the FoodKG [3] as a data source for recipes, ingredients, and linked data about ingredient classification and nutrition. We use a prototype of several healthy eating guidelines for diabetic patients to serve as the guidelines to apply in the recommendation process. For this application, we implemented a FREx pipeline to recommend recipes similar to the user's favorite recipes (based on embeddings produced by RECIPTOR [4]) as well as knowledge-driven scoring and filtering. Filtering was conducted to remove recipes using prohibited ingredients and their subclasses (e.g., allergic restrictions by the user). Scoring stages were formed based on the healthy eating guidelines, which consisted of the conditions in which to apply the guideline (e.g., based on the user's age, sex, and BMI) and the directives to apply (e.g., eat fewer than 2,300 mg of sodium in a day). Through the FREx pipeline we developed, the final recommendations for recipes that were produced could include explanations about factors such as which guidelines the given recommendation adheres to.

4 Discussion

In its current state, FREx is intended to be highly flexible in terms of the kinds of data used as input and the methods used to produce recommendations. Our example use case, FoodRec, demonstrates how RDF data can be queried and converted into objects, which in turn can be passed through a pipeline to perform knowledge-driven recommendations. FREx can be similarly utilized in new domains by constructing the appropriate domain object models and querying services. New pipelines to perform different recommendation strategies can also easily be developed and incorporated.

A key limitation of our current iteration of FREx is the need for developers to define the domain object models for their application, as well as the need to manually craft explanation text. If the recommendation application is developed using an ontology, domain objects may be easily modeled, but it may be difficult to determine what pieces of information from the ontology are relevant to the application. Additionally, while our framework focuses on explanations about the recommendation stages, such explanations are not produced automatically. While it is possible to extend the Explanations class to apply some strategies like templating systems (e.g., "This item received a score of {*item_score*} because its {*property_name*} was below the threshold of {*threshold_value*}"), such templating systems would once again be fairly specific to the application or domain and not be widely applicable to certain scoring or filtering techniques.

5 Conclusion and Future Work

FREx provides a high level structure aimed at aiding developers to prototype recommendation systems that integrate explanations throughout the recommendation process. A flexible pipeline system is introduced to handle the common components of generating, filtering, and scoring items to recommend. We demonstrate the use of FREx for a food recommendation use case in the FoodRec application.

Moving forward, we aim to integrate FREx with other tools to better leverage semantics provided by ontologies. Our goal here is to incorporate tools that can automatically generate Python classes from the domain ontology to more easily support FREx's object oriented workflow. We additionally hope to expand our work to provide a greater level of automation in providing explanations. This will help to further reduce the burden on developers to manually supply FREx with explanation text, making prototyping of such applications even easier. Lastly, we plan to perform an evaluation of using FREx for developing recommender systems. This evaluation would mostly focus on the framework's usability, especially in terms of more efficiently creating knowledge-driven recommendation pipelines.

6 Acknowledgements

This work is partially supported by IBM Research AI through the AI Horizons Network.

References

1. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-Aware Recommender Systems. AI Magazine **32**(3), 67–80 (2011). https://doi.org/10.1609/aimag.v32i3.2364, https://ojs.aaai.org/index.php/aimagazine/article/view/2364, number: 3
2. Dourish, P.: What we talk about when we talk about context. Personal and Ubiquitous Computing **8**(1), 19–30 (Feb 2004). https://doi.org/10.1007/s00779-003-0253-8, https://doi.org/10.1007/s00779-003-0253-8
3. Haussmann, S., Seneviratne, O., Chen, Y., Ne'eman, Y., Codella, J., Chen, C.H., Mcguinness, D., Zaki, M.: FoodKG: A Semantics-Driven Knowledge Graph for Food Recommendation, pp. 146–162. Springer (10 2019). https://doi.org/10.1007/978-3-030-30796-7_10
4. Li, D., Zaki, M.J.: Reciptor: An effective pretrained model for recipe representation learning. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 1719–1727. KDD '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3394486.3403223, https://doi.org/10.1145/3394486.3403223
5. Lu, J., Wu, D., Mao, M., Wang, W., Zhang, G.: Recommender system application developments: A survey. Decision Support Systems **74**, 12–32 (2015). https://doi.org/https://doi.org/10.1016/j.dss.2015.03.008
6. Zhang, Y., Chen, X.: Explainable recommendation: A survey and new perspectives. Found. Trends Inf. Retr. **14**, 1–101 (2020)