Uptrendz: API-Centric Real-Time Recommendations in Multi-domain Settings

Emanuel Lacic[1], Tomislav Duricic[1,2], Leon Fadljevic[1], Dieter Theiler[1], and Dominik Kowald[1,2](✉)

[1] Know-Center GmbH, Graz, Austria
{elacic,tduricic,lfadljevic,dtheiler,dkowald}@know-center.at
[2] Graz University of Technology, Graz, Austria

Abstract. In this work, we tackle the problem of adapting a real-time recommender system to multiple application domains, and their underlying data models and customization requirements. To do that, we present Uptrendz, a multi-domain recommendation platform that can be customized to provide real-time recommendations in an API-centric way. We demonstrate (i) how to set up a real-time movie recommender using the popular MovieLens-100 k dataset, and (ii) how to simultaneously support multiple application domains based on the use-case of recommendations in entrepreneurial start-up founding. For that, we differentiate between domains on the item- and system-level. We believe that our demonstration shows a convenient way to adapt, deploy and evaluate a recommender system in an API-centric way. The source-code and documentation that demonstrates how to utilize the configured Uptrendz API is available on GitHub.

Keywords: Uptrendz · API-centric recommendations · Multi-domain recommendations · Real-time recommendations

1 Introduction

Utilizing recommender systems is nowadays recognized as a necessary feature to help users discover relevant content [14,15]. Most industry practitioners [3], when they build a recommender system, adapt existing algorithms to the underlying data and customization requirements of the respective application domain (e.g., movies, music, news, etc.). However, the focus of the research community has recently shifted towards building recommendation systems that simultaneously support multiple application domains [4,7,16] in an API-centric way.

In this work, we demonstrate Uptrendz[1], an API-centric recommendation platform, which can be configured to simultaneously provide real-time recommendations in an API-centric way to multiple domains. Uptrendz supports popular recommendation algorithms, e.g., Collaborative Filtering (CF), Content-based Filtering (CBF, or Most Popular (MP), that are applied across different

[1] https://uptrendz.ai/.

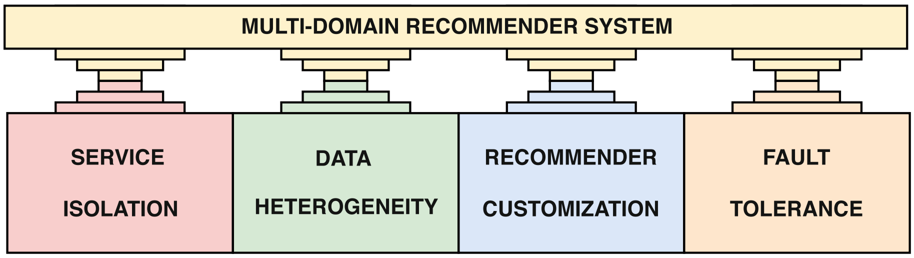


Fig. 1. Aspects that need to be addressed when building a recommender system for a multi-domain environment [10].

application domains. The focus of this demonstration is to show how domain-specific data-upload APIs can be created to support the customization of the respective recommendation algorithms. Using the MovieLens-100k dataset [6] and a real-world use-case of entrepreneurial start-up founding[2], we show how such an approach allows for a highly customized recommendation system that can be used in an API-centric way. The source-code and documentation for this demonstration is available via GitHub[3].

2 The Uptrendz Platform

The Uptrendz platform is built on top of the ScaR recommendation framework [11]. As shown in [10] and Fig. 1, the microservice-based system architecture addresses four distinctive requirements of a multi-domain recommender system, i.e., (i) service isolation, (ii) data heterogeneity, (iii) recommender customization, and (iv) fault tolerance. Uptrendz provides a layer on top of the framework to dynamically configure an application domain and to instantly provide an API to (i) upload item, user and interaction data, and (ii) request recommendations.

Domain-Specific Data Model. As discussed by [1], different domains may employ the same recommender algorithm but can differ with respect to what kind of data is utilized to build the model (e.g., interaction types, context, etc.). Given an API-centric approach, we show that in order to support the customization of recommender algorithms with domain-specific parameters, the underlying platform needs to unambiguously know which source of information should be used to calculate the recommendations. To do that, the Uptrendz platform first allows generating customized data upload APIs for multiple item and user entities (see Table 1). Second, with respect to interaction data, both user-item and user-user interactions can be configured. The interaction API is further customized in accordance to what kind of interactions the respective application domain actually supports, i.e., (i) registered users, anonymous sessions or

[2] https://cogsteps.com/.
[3] https://github.com/lacic/ECIR2023Demo.

Table 1. Supported attributes to configure the data upload API for items and users.

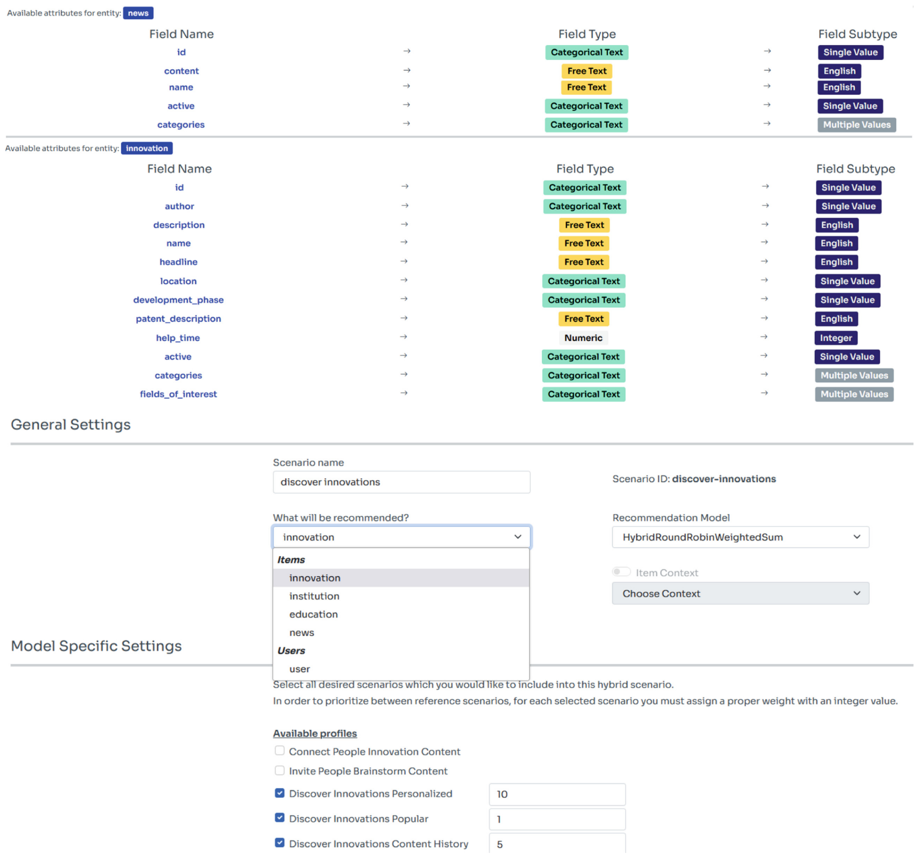
Type	Sub-Type	Description
Categorical Text	Single Value	String value, which usually represents a category. Used for **post-filtering** recommendation results.
	Multiple Values	List of string values, which usually represent an array of categories. Used for **post-filtering** recommendation results.
Free Text	English	**English text**, which is processed and utilized for **content-based** recommendations.
	German	**German text**, which is processed and utilized for **content-based** recommendations.
Numeric	Integer	Used for **post-filtering** recommendations (e.g., user age).
	Real	Used for **post-filtering** recommendations (e.g., price).
Date	–	Date information for the respective entity (e.g., creation date)

both, (ii) interaction timestamp tracking, and (iii) type of interaction (explicit or implicit).

Recommender Customization. The Uptrendz platform fosters the notion of defining personalization scenarios (i.e., use-cases) when creating recommendation APIs. The available selection of real-time recommendation models [11] for a given scenario depends on (i) what should be recommended (e.g., item or user entities), (ii) for whom the recommendations are targeted (e.g., registered or anonymous users) and, (iii) what kind of context is given [2] (e.g., item ID to recommend relevant content for). As we adopt a non-restricted configuration with respect to the number of freely defined user interaction types, algorithms that use this kind of data (e.g., Collaborative Filtering) can be customized to utilize any subset of the list of available interactions as well as to define how much weight a particular interaction type should have. With respect to post-filtering recommendation results, each model can use categorical (e.g., tags [12] or other semantic representations [8]) or numerical data attributes to ensure that the resulting recommendations either contain or exclude a particular value (see Table 1 for complete list of attributes).

3 Multi-domain Support

In order to provide a multi-domain recommender platform, we support the notions of a system-level and item-level domain in accordance with [5]. For the former, items and users belong to distinct systems (e.g., Netflix and Amazon).



Available attributes for entity: news

Field Name		Field Type		Field Subtype
id	→	Categorical Text	→	Single Value
content	→	Free Text	→	English
name	→	Free Text	→	English
active	→	Categorical Text	→	Single Value
categories	→	Categorical Text	→	Multiple Values

Available attributes for entity: innovation

Field Name		Field Type		Field Subtype
id	→	Categorical Text	→	Single Value
author	→	Categorical Text	→	Single Value
description	→	Free Text	→	English
name	→	Free Text	→	English
headline	→	Free Text	→	English
location	→	Categorical Text	→	Single Value
development_phase	→	Categorical Text	→	Single Value
patent_description	→	Free Text	→	English
help_time	→	Numeric	→	Integer
active	→	Categorical Text	→	Single Value
categories	→	Categorical Text	→	Multiple Values
fields_of_interest	→	Categorical Text	→	Multiple Values

General Settings

Scenario name: discover innovations

Scenario ID: **discover-innovations**

What will be recommended?: innovation

Items
- innovation
- institution
- education
- news

Users
- user

Recommendation Model: HybridRoundRobinWeightedSum

Item Context

Choose Context

Model Specific Settings

Select all desired scenarios which you would like to include into this hybrid scenario.
In order to prioritize between reference scenarios, for each selected scenario you must assign a proper weight with an integer value.

Available profiles

☐ Connect People Innovation Content	
☐ Invite People Brainstorm Content	
☑ Discover Innovations Personalized	10
☑ Discover Innovations Popular	1
☑ Discover Innovations Content History	5

Fig. 2. Example of supporting multiple domains on the item-level (up) and configuring a hybrid recommendation algorithm (below) with previously created APIs.

For the latter, individual domains have different types of items and users which may share some common attributes (e.g., movies and books).

Demo Walkthrough: System-level Domain. When a domain is created on a system level, the underlying data is physically stored in a different location than the data of other domains. Hence, domains do not share any data between themselves and the underlying services are isolated so that the performance of one domain does not impact the performance of another domain (e.g., during request load peaks). We demonstrate how to create a movie recommender on a system level. To utilize the MovieLens-100k dataset [6], we first need to configure the respective data services to upload (i) movie, (ii) user, and (iii) interaction data. Each entity needs to be separately created in the Uptrendz platform in order to generate an API that can be used to upload the MovieLens-specific data attributes. This allows creating recommendation scenarios for (i) similar

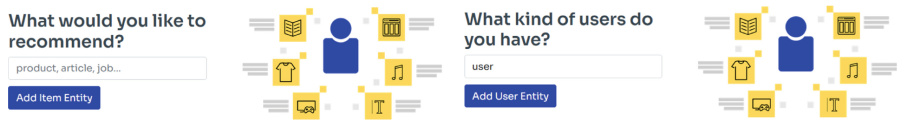

What would you like to recommend?
product, article, job...
Add Item Entity
What kind of users do you have?
user
Add User Entity

Fig. 3. Uptrendz requires the specification of (i) the item types that should be recommended (e.g., products or users, depending on the domain - left figure), and (ii) the user types for which recommendations should be generated (e.g., registered users or session users - right figure).

movies (CBF), (ii) popular horror movies (MP with post-filtering), (iii) movies based on ratings (CF), (iv) their weighted hybrid combination (e.g., for cold-start settings [13], and (v) a user recommender for a given movie.

Demo Walkthrough: Item-level Domain. To showcase how to configure Uptrendz to support multiple-domains on an item-level, we present the use-case of entrepreneurial start-up founding. Here, we recommend experts that can provide feedback to an innovation idea, support co-founder matching, help incubators, innovation hubs and accelerators to discover innovations but also provide relevant educational materials until the innovation idea matures enough to form a start-up. In this case, each recommendable entity has a separate data model and can be viewed as part of a standalone application domain. Figure 2 depicts how adding multiple item entities in the data catalog allows customizing data attributes for the respective domain. While configuring a recommendation algorithm, the respective item-level domain can be selected to be recommended. Here, via the example of a hybrid algorithm, only pre-configured algorithms can be utilized that belong to the same domain (i.e., innovation recommendations).

Finally, in Fig. 3, we show how Uptrendz allows the specification of (i) different item types that can be recommended, and (ii) different user types for which recommendations should be generated. Our demo application includes different specification examples.

4 Conclusion

In this paper, we present Uptrendz, an API-centric recommendation platform that can be customized to provide real-time recommendations for multiple domains. To do that, we support the notions of a system-level and item-level domain. We demonstrate Uptrendz using the popular MovieLens-100k dataset and the use-case of entrepreneurial start-up founding.

In future work, we plan to support even more use cases from other domains, e.g., music recommendations [9]. Here, we also want to integrate fairness-aware recommendation algorithms for mitigating e.g., popularity bias effects.

Acknowledgements. This research was funded by CogSteps and the "DDAI" COMET Module within the COMET - Competence Centers for Excellent Technologies

Programme, funded by the Austrian Federal Ministry for Transport, Innovation and Technology (bmvit), the Austrian Federal Ministry for Digital and Economic Affairs (bmdw), the Austrian Research Promotion Agency (FFG), the province of Styria (SFG) and partners from industry and academia.

References

1. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 335–336. RecSys '08, ACM (2008). https://doi.org/10.1145/1454008.1454068, http://doi.acm.org/10.1145/1454008.1454068
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_7
3. Amatriain, X., Basilico, J.: Past, present, and future of recommender systems: An industry perspective. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 211–214 (2016)
4. Bonab, H., Aliannejadi, M., Vardasbi, A., Kanoulas, E., Allan, J.: Cross-market product recommendation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 110–119 (2021)
5. Cantador, I., Fernández-Tobías, I., Berkovsky, S., Cremonesi, P.: Cross-domain recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 919–959. Springer, Boston, MA (2015). https://doi.org/10.1007/978-1-4899-7637-6_27
6. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. Acm Trans. Interact. Intell. Syst. (tiis) **5**(4), 1–19 (2015)
7. Im, I., Hars, A.: Does a one-size recommendation system fit all? the effectiveness of collaborative filtering based recommendation systems across different domains and search modes. ACM Trans. Inform. Syst. (TOIS) **26**(1), 4-es (2007)
8. Kowald, D., Dennerlein, S.M., Theiler, D., Walk, S., Trattner, C.: The social semantic server a framework to provide services on social semantic network data. In: Proceedings of I-SEMANTICS 2013), pp. 50–54 (2013)
9. Kowald, D., Muellner, P., Zangerle, E., Bauer, C., Schedl, M., Lex, E.: Support the underground: characteristics of beyond-mainstream music listeners. EPJ Data Sci. **10**(1), 1–26 (2021). https://doi.org/10.1140/epjds/s13688-021-00268-9
10. Lacic, E., Kowald, D., Lex, E.: Tailoring recommendations for a multi-domain environment. In: Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning (RecSysKTL'2017) co-located with the 11th ACM Conference on Recommender Systems (RecSys'2017) (2017)
11. Lacic, E., Kowald, D., Parra, D., Kahr, M., Trattner, C.: Towards a scalable social recommender engine for online marketplaces: The case of apache solr. In: Workshop Proceedings of WWW'2014, pp. 817–822 (2014)
12. Lacic, E., Kowald, D., Seitlinger, P., Trattner, C., Parra, D.: Recommending items in social tagging systems using tag and time information. In: Proceedings of the 1st International Workshop on Social Personalisation (SP'2014) co-located with Hypertext'2014 (2014)

13. Lacic, E., Kowald, D., Traub, M., Luzhnica, G., Simon, J.P., Lex, E.: Tackling cold-start users in recommender systems with indoor positioning systems. In: Poster Proceedings of the 9th {ACM} Conference on Recommender Systems. Association of Computing Machinery (2015)
14. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI'06 Extended Abstracts on Human Factors in Computing Systems, pp. 1097–1101. ACM (2006)
15. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**(3), 56–58 (1997)
16. Roitero, K., Carterette, B., Mehrotra, R., Lalmas, M.: Leveraging behavioral heterogeneity across markets for cross-market training of recommender systems. In: Companion Proceedings of the Web Conference 2020, pp. 694–702 (2020)