The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Making Personalized Recommendation through Conversation: Architecture Design and Recommendation Methods

Sunhwan Lee, Robert Moore, Guang-Jie Ren, Raphael Arar, Shun Jiang
IBM Almaden Research Center
650 Harry Rd
San Jose, California 95120

Abstract

Due to popularity in texting and messaging, a recent advancement of deep learning technologies, a conversation-based interaction becomes an emerging user interface. While todays conversation platforms offer basic conversation capabilities such as natural language understanding, entity extraction and simple dialogue management , there are still challenges in developing practical applications to support complex use cases using a dialogue system. In this paper, we highlight such challenges and share practical knowledge learned from our experiences on developing a leisure travel shopping application that combines a personalized recommendation system and a conversation system. Such efforts include a conversation design, extraction of user intents, communication of variables between a dialogue system and analytics engines, and dynamic user interface designs. In particular, we introduce our approach to overcome the unique challenges, *understanding user's intent*, when dialogue system met personalized recommendation system. Furthermore, we propose a *semantic mapping* as a novel method to utilize undefined user's preferences when producing recommended items.

Finally, examples of recommendations based on natural language conversations are provided in order to exhibit how components in the overall architecture are seamlessly orchestrated. In general, our framework provides guiding principles and best practices on the implementation of task-oriented dialogue system connected with other components in the overall architecture.

Introduction

In the past few years, we have seen a resurgence of interest in conversational user interfaces as the major computer and Internet companies have released virtual agents and platforms including Siri, Alexa, Cortana, Google Assistant, Watson and more. While chatbots have been around since the ELIZA system in the 1960s, the current platforms for building conversational interfaces, Api.ai, Wit.ai, Luis, IBM Watson Conversation, etc., combine powerful natural language classification techniques with authoring tools that are accessible to non-programmers. The result has been a proliferation of chatbots and virtual assistants on the web and on mobile devices. In addition, with advances in far-field microphone arrays, voice interfaces, which enable a system to hear the user even from across the room, have entered the living room.

On the other hand, recommender systems emerged as an independent and active research area in the mid-90's, when explicit user rating was the main resource to build the system (Hill et al. 1995), (Resnick et al. 1994). Depending on how users and items are modeled, there are two main approaches to building recommender systems: collaborative filtering (Koren, Bell, and Volinsky 2009) and content-based approach (Adomavicius and Tuzhilin 2005). Recommender systems can also be differentiated based on the type of source data on which the system is built - explicit or implicit user feedback. Because different approaches have their own limitations, a hybrid recommender system (Burke 2002), which not only uses observations from implicit feedback but also utilizes various contextual information, was also proposed.

Motivated by an emergence of this new user interface, we would like to share our experiences on building a personalized recommender system whose inputs are provided from the conversation interface. There are previous attempts to integrate a personalized recommendation in a dialogue system, but we find very few literatures describing the overall procedure and providing practical guidances to generate recommendations according to users' interaction with a dialogue system. One such implementation is introduced in (Atzori, Boratto, and Spano 2017) but the main focus is limited on the interaction between a recommender system and chatbots in the front end. Our paper aims to share the following aspects of building and designing the overall architecture for realizing a personalized recommender system using a dialogue system. First, we introduce the design of architecture as well as individual components and their work flows. Second, the method to extract user's intent and preferences from a dialogue system and how they are matched with the features of destinations for recommending destinations is explained followed by web-conversation user interface design.

Travel Advisor in Dialog System

Todays travel shopping sites and applications have almost universally adopted the design of search forms where users provide travel dates, locations and a limited set of search criteria and in turn get a generic list of flight, hotel, car rental and tour package options. The user engagement is limited

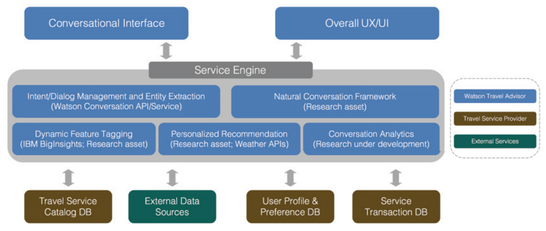

Conversational Interface
Overall UX/UI
Service Engine
Intent/Dialog Management and Entity Extraction (Watson Conversation API/Service)
Natural Conversation Framework (Research asset)
Dynamic Feature Tagging (IBM BigInsights; Research asset)
Personalized Recommendation (Research asset; Weather APIs)
Conversation Analytics (Research under development)
Watson Travel Advisor
Travel Service Provider
External Services
Travel Service Catalog DB
External Data Sources
User Profile & Preference DB
Service Transaction DB

Figure 1: Overall architecture of Travel Advisor

and travel companies know little beyond user search. To address the issues, we have developed a web-based conversation application that provides personalized travel recommendations. A dialogue system is a primary interface for user's interaction and recommendations of destinations are provided at the right time by understanding user's utterances and intents. However, users can also interact with the web interface to modify their preferences captured in the conversation or to interact with the results of recommendations. The expectation is to increase the level of user engagement by providing more flexible and personalized shopping experiences by enabling the interaction with a natural language. Furthermore, the conversation logs are stored in the back end for further analytics such as deriving a personal insights or quantitative measure of user's level of satisfaction through the conversation.

Architecture

In this section, we introduce the architecture and its components required to develop a personalized recommendation system in a dialogue system. Figure 1 shows a main platform and components in the middle box, and input data sources and the front end components, such as user interface design, in bottom and top box respectively. Since the processing of input data sources will be covered in later section, the details of components related with a dialogue system, service integration engine, and conversation-web user interface design are provided in this section.

Conversation services and templates

Conversation services is a main framework for developing chatbots by understanding the user's utterance. In order for the development of a recommendation system on top of a dialogue system, we learned three important aspects that must be considered thoroughly. *Context variables* must be set by the dialogue service to accommodate variations of entities. Once context variables are set up, then *entities* or key words can be detected from the conversation. For the development of destination recommendations, the preferences of users like nightlife, restaurants, beaches, are created and captured as entities in the dialogue system based on context variations. Table 1 shows some examples from entities used as features to describe the destinations, and some examples of context variables. In fact, coming up with the comprehensive set of entities and their variations requires a lot of domain knowledge and manual efforts. Even though we referred to many travel related popular websites for a list of entities, we would like to reserve further improvements on the quality of destination description as future works.

Recognizing not only a user's preferences using entities but also the *intent* (Allen and Perrault 1980) is essential to determine a right time to invoke the recommendation engine. While current major chatbots and virtual assistant platforms vary, most use the same general paradigm, what we might call the intent-entity-context-response (IECR) paradigm. In this paradigm, 1) the user's *intent*, or conversational action, is determined largely through natural language classification, 2) the *entities* in the user's utterances are extracted through keyword matching, 3) the *context* of the current and prior turns in the conversation are preserved in variables and all of these are used by a dialogue manager to determine 4) the system's response back to the user, in text or voice. While IECR platforms provide powerful natural language processing tools, they provide little or no guidance on how to create an interaction that is like a natural conversation (Moore et al. 2017). From a natural language perspective, English works the same way no matter the context in which it is used, whether a poem, document, song or conversation. However, there is an organization to natural conversation that is not provided for by natural language processing tools. To fill this gap, we draw on the field of Conversation Analysis, which provides a model of sequence organization (Schegloff 2007) in natural human conversation, as well as empirically derived patterns of conversational activities (Moore and Arora 2016).

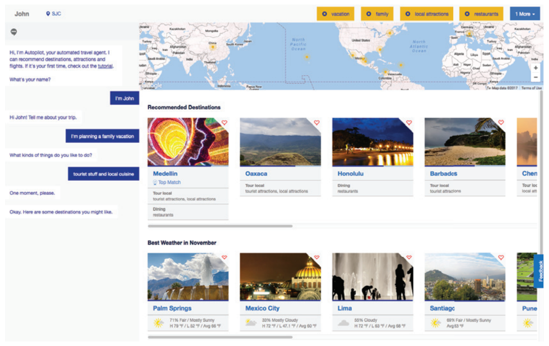
In our conversational system, we use natural language classification to determine the type of action the user is taking, for example, "I want to go someplace with historic buildings" is classified as a destination request, while "show me flights from Chicago to Paris" as a flight request. The intent or action type determines the form of the appropriate response: a set of destinations vs. a set of flights. We then use keyword matching to extract entities from these phrases such as historic buildings, Chicago, and Paris. When the dialogue manager collects all of the preferences required for a particular request type, for example, origin, destination, departure date and return date for a flight request, it signals to the back end to look up information or make recommendations based on the current set of preferences. The conversation service essentially translates what the user says into a query that is actionable by the back end.

Conversation-Web user interface

One of the unique aspect of our application is to build a dialogue system in conjunction with a web-based user interface. It provides more flexible and intuitive way of interactions such as a visualization of list of recommendations

Table 1: Examples of entities and context variables

Entity	Context variables
local attractions	local hangouts, local hot spot
nightlife	party city, night clubs, jazz clubs
landmarks manmade	towers, statues, monuments
landmarks natural	rainforests, fall leaves
art museums	exhibit, gallery, painting



Hi, I'm Autopilot, your automated travel agent. I can recommend destinations, attractions and flights. If it's your first time, check out the tutorial.

What's your name?

I'm John

Hi John! Tell me about your trip.

I'm planning a family vacation

What kinds of things do you like to do?

tourist stuff and local cuisine

One moment, please.

Okay. Here are some destinations you might like.

Figure 2: Conversation-Web user interface

on a map, showing detailed information about the destination when clicking the destinations, and modifying the list of preferences through a web interface. However, it requires more sophisticated orchestration in the back end because two user interfaces should function in a synchronized way.

The user interface shown in figure 2 has multiple components requiring such synchronizations. The top portion has sections for user's name, location, and the list of preferences. These components are updated as the conversation captures relevant information or users can manually update the fields. The right pane is reserved for showing the results of analytics, the outputs from a recommendation engine in this case. There are multiple sections to display recommendations based on different categories such as weather, user's preference. These sections are also adaptive to the conversation between a user and the dialogue system. For example, if a user mentions a particular month, then the recommendations based on the weather condition are updated with a user specified month. A map is displaying geographic locations of recommended destinations and users can interact with destination cards below the map using conventional ways of interacting with web user interface. The contents are refreshed when a dialogue system detects the intent to request new recommendations.

Service integration engine

In addition to physically hosting modules in the cloud environment, the role of service integration engine is to orchestrate the functions among conversation services, recommendation engine, and web-conversation user interface. Because our application is not a stand-alone dialogue system, the communication between components in the architecture is very important to implement desired features in the application. As a result, many factors like scalability, resiliency, and communication bandwidth must be considered thoroughly when designing and building the service integration engine.

Personalized Recommendation

Various types of personalized recommendation have been developed and proposed as new kinds of data sources emerges. We believe that a dialogue system is a new platform which generates another new type of data source, and thus practical experiences that we share in this paper will be helpful for further research and practical developments. Although other platforms like natural language search take user's intents and preferences in a natural language and return the recommended items, building a recommendation system with a dialogue system requires the support of continuous interactions between a dialogue system and users. The continuous interactions stem from the flexibility of a dialogue system and the conventional way of capturing a user's preference tends to create a discontinued results of recommendation. In most of natural language based recommender system, recommendations are returned with a single turn interaction between a user and the system. In a dialogue system, however, users can continuously indicate preferences in as many turns as they want. Also modification, addition, and removal of preferences are expected in a dialogue system and thus a recommender system needs to manage such cases in flexible manner as well.

In this section, we provide the precise steps to process unstructured data for constructing features of destinations, to understand the user's preferences, and finally to generate personalized content-based recommendation system.

Content-based recommendation and semantic mapping

Content-based recommendation leverages attributes of items to be recommended. We used the entities, some of which are listed in table 1, as attributes to embed text descriptions of destinations in our catalog. Entities are identified in raw text descriptions from Wikipedia and Wikitravel, and the number of occurrence of entities are used to embed each destination into the vector of n-dimension, where n is the number of entities. Wikipedia has a general information while the contents of Wikitravel are more geared toward travel-specific information. Both services crowdsource the contents, which can be programmatically accessed via MediaWiki action API with the name of destination as a parameter.

One challenge of building a natural language based recommendation systems is to handle undefined preferences from users. Although context variables capture variations of attributes but it is impossible to extend the number of context variables to capture all possible variations. These challenges exist in all platforms whose main interactions happen in the format of natural languages, including a dialogue system. To overcome this challenge, we adapted a novel method of leveraging semantic distances between user's preferences and predefined entities. For example, if the word *family* is captured as a keyword in the conversation, then the semantic distance between *family* and existing entities are computed using WordNet corpus in NLTK python module (Bird, Klein, and Loper 2009). Then a vector, whose elements are a semantic distance between *family* and entities, represents a user's preference. Examples of semantically similar and dissimilar entities given undefined preferences are shown in table 2. Content-based recommender system uses a cosine similarity between two vectors, one from user's preferences and another from attributes of destinations, to recommend destinations in response to user's taste.

Table 2: Examples of semantically similar and dissimilar entities with undefined preferences

Preference	Similar entities	Dissimilar entities
family	sports, beaches	wineries, spas
bachelor party	bars, events	relaxing, aquarium
anniversary	romantic, nightlife	adventurous

Examples

Two examples of recommended destinations given a conversation flow are shown in Example 1 and Example 2.

Example 1

SYS: Hi, I'm your Travel Advisor. What's your name?
USR: I'm Amy
SYS: Hi Amy! Tell me about your trip.
USR: I'm planning a **family vacation** in the **USA**
SYS: What kinds of things do you like to do?
USR: can you give an example?
SYS: Tourist attractions, museums, nightlife?
USR: oh, **tourist stuff** and **local cuisine**
SYS: One moment, please.
SYS: Okay. Here are some destinations you might like.

Recommendation: Honolulu, Orlando, Boston, Charleston, Sacramento, Santa Barbara, Fort Lauderdale, Columbus, Omaha, Richmond

Example 2

SYS: Hi, I'm your Travel Advisor. What's your name?
USR: I'd like destinations in **December**
SYS: What kinds of things do you like to do?
USR: **hiking** and **relaxing**
SYS: One moment, please.
SYS: Okay. Here are some destinations you might like.
USR: **great weather** too
SYS: One moment, please.
SYS: Okay. Here are some destinations you might like.

Recommendation: Colorado Springs, Denver, Tucson, Chattanooga, Santiago, Tegucigalpa, San Francisco, Knoxville, Santa Barbara, Branson

In the conversation, inputs to the recommendation system are underlined and highlighted in bold fonts. Two examples attempt to demonstrate different types of entities related with things to do, location, time, and weather of preferred travel, and how they are reflected in the results of recommended destinations. More rigorous evaluation of the performance of proposed recommendation method will be done by deploying the recommendation in the production and monitoring the performance. But we tried to demonstrate working examples which were generated by the integration of a dialogue system and the recommendation engines.

Future Works

We presented the overall architecture for developing the recommendation system interacting with a dialogue system, and introduced the role of individual components in the architecture for sharing practical experiences and unique challenges. There are still several enhancements as future works.

As a new type of data sources, conversation logs can be interesting data points to the recommendation system. When web interface became popular, data from user's interactions, such as click and scroll, contributed to improve the performance of recommendations. We foresee similar trends as more real-world applications are adapting a dialogue system as their platform. It will certainly help understanding the users from multiple angles. Further analytics on the correlation between conversation logs and user's action to the recommended items through click or like can be used as a learning process to create attributes and their associated weights to items. The flexibility of a dialogue system can enrich the set of attributes and alleviate a lot of manual efforts to define features or attributes representing items.

References

Adomavicius, G., and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.* 17(6):734–749.

Allen, J. F., and Perrault, C. 1980. Analyzing intention in utterances. *Artificial Intelligence* 15(3):143 – 178.

Atzori, M.; Boratto, L.; and Spano, L. D. 2017. Towards chatbots as recommendation interfaces.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Hill, W.; Stead, L.; Rosenstein, M.; and Furnas, G. 1995. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 194–201.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Moore, Robert J., R. H., and Arora, A. 2016. The machinery of natural conversation and the design of conversational machines: Applying models from conversation analysis to conversational ux design. In *American Sociological Association annual meeting*.

Moore, R. J.; Arar, R.; Ren, G.-J.; and Szymanski, M. H. 2017. Conversational UX design. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 492–497.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, 175–186.

Schegloff, E. 2007. *Sequence Organization in Interaction: Volume 1: A Primer in Conversation Analysis*. Primer in conversation analysis. Cambridge University Press.