Conversational Recommender System Chatbot Based on Functional Requirement

David Theosaksomo, Dwi H. Widyantoro
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
davidtheosaksomo@gmail.com, dwi@stei.itb.ac.id

***Abstract*— To make a decision in choosing a product with insufficient information or knowledge, we can use a recommendation. Recommender system is a system that gives recommendation to user. In recommendation system that recommends product, user requirements can be given as technical requirement or functional requirement. User that not familiar with technical specification of a product will be easier to specify the needs via functional requirement. With the nature of interacting with user via conversation, chatbot can be used as a conversational recommender system. In this work we will be developing chatbot as a conversational recommender system based on user functional requirement. The developed system will be evaluated using usability testing method.**

Keywords—chatbot; recommender system; conversational; functional requirements;

I. INTRODUCTION

In our daily life, there is a chance we got into situation where we must make a decision, but we don't have enough information or knowledge. For example, when we are in a foreign city and want to eat at a restaurant, but we don't know about restaurants in that city. To solve the problem, we could ask a family or friend that live in the city for recommendation. To make a decision with insufficient information, we can use recommendation. Recommendation can speed up our process of decision making.

Recommender system is a system that gives recommendation to user. The system can recommend things in form of product, service, or contents that comply with user needs or interests. To be able to give recommendation, the system needs information about user interest or requirement. To obtain those information, different methods can be used. One of them is to establish a conversation with the user. Conversational recommender system is a recommender system that uses conversation as an interaction with user.

In a recommender system that recommends a product, user requirement can be defined as technical requirement or functional requirement. Technical requirement is user requirement based on the technical specification of a product. Functional requirement is user requirement based on the aim of use of the product or the product function for the user. With lot of varieties in products and the complexity of product technical specification e.g. in cars, smartphones, laptops, users may not aware or fully understand of a product technical specification. For user that is not familiar with a product technical specification, it is easier for them to specify their requirement in form of functional requirement.

With the nature of interacting with user via conversation, chatbot can be used as a conversational recommender system. Chatbots are increasing in popularity and we may find them or even using them in our daily life. Chatbot are used across industries and have a lot of varieties, from chatbot that helps to obtain simple information to intelligent chatbot that we use as personal assistant. The growth of artificial intelligence field such as deep learning enables smarter chatbots to be developed.

Baizal et al. developed a conversational recommender system based on user functional requirement that tries to imitate a salesperson when giving a recommendation to user [1]. However, the system is using forms and menus to interact with users, and not yet using a conversational interface. This paper proposes using a chatbot as an interface to build a conversational interaction for the recommender system. Chatbot built in this work offers something unique that differentiate it from other conversational recommender system, that is this chatbot collects and uses user functional requirement to give recommendation to user instead of technical requirements.

This paper is divided into five sections. This section, Introduction gives the introduction and general information about this paper. Section II gives information about works that is related to this paper. Section III gives analysis about the chatbot requirement and present the chatbot design. Section IV presents the implementation of the chatbot, and Section V present the evaluation process of the chatbot. The last section, Section VI presents the conclusion and future works.

II. RELATED WORKS

There are works done previously that is related to conversational recommender system. Bridge in 2002 used conversational approach on a recommender system with a mixed-initiative interaction [2]. This research developed Sermo, a conversational recommender system that uses dialogue grammar that defines the interaction with the user. Sermo demonstrated that using dialogue grammar can make a more "conversational" recommender system. Aktas et al. in 2004 built a Conversational Case-based Recommender System, ServoGRID, that combines Resource Description Framework (RDF) and Case-based Reasoning method to make a selection on earthquake simulation code [3]. Wärnestål et al. in 2007 presented two interview strategies called interview and delivery

978-1-7281-5247-9/19/$31.00 ©2019 IEEE

strategy [4]. The purpose of the interview strategy is to collect information about user preference for certain domain entity type (e.g. genres, actors for movie domain) or specific item (e.g. specific movies). The purpose of the delivery strategy is to present the result of the interview strategy i.e. giving recommendation to user. The interview and delivery strategies then presented as a model for conversational music recommender system called CORESONG.

Baizal and Widyantoro in 2014 developed a framework to build a conversational recommender system based on functional requirement. This work integrates Navigation by Asking (NBA) strategy and Navigation by Proposing (NBP) to collect user requirement [1]. The knowledge representation used in this system is ontology. The framework built consist of ontology structure and computational model to generate interactions with the user.

In conversational recommender system, there is a problem that may arise. User may experience difficulties in specifying the requirement or giving incomplete queries. Baizal et al. in 2016 proposed a query refinement model to helps user specify their requirement as functional requirement [5]. The model aims to reduce the number of questions given during the interaction and creating a more efficient interaction. In the research a semantic relation on ontology was used to produce questions pointing to functional requirement. The model produced able to reduce the recommendation candidate after certain iteration of the interaction.

With the raising popularity of chatbot, other works developing chatbot as recommender system starting to emerge. Ramachandran et al. in 2015 built a conversational system for TV program discovery using recommendation [6]. The work combined advanced technology for NLU, state tracking, dialogue management, Knowledge Graph Inference, and personalized recommendation to build a novel recommender system. The TV programs are represented by a vector of features of the movie e.g. genre, actors. The recommendation candidates are scored by a K-Nearest Neighbor algorithm and adjusted by explicit user preference e.g. user disliked the movie's actor. Holotescu in 2016 built MOOCBuddy, a chatbot that gives recommendation of Massive Open Online Course (MOOCBuddy) e.g. Coursera, edX, FutureLearn [7]. MOOCBuddy is built for Facebook Messenger and gives recommendation based on social media profile and user interest. Colace et al. in 2017 built BotWheels for recommending tires [8]. BotWheels uses Petri Net for modeling and formalization. The performance result is good and have potential to be improved. Yuichiro et al. in 2018 built a Graphical User Interface (GUI) chatbot that gives recommendation based on user preference [9]. User preference is acquired by giving question to the user and receiving feedback from user. In this works a balancing is done to the interaction strategy to balance between asking and giving recommendation with parameter values, initial recommendation threshold (α), feedback weight (β), dan recommendation threshold decrement (γ). To define the values, a simulation model with performance evaluation is done to find values with the best performance. With a lot of conversational recommender system chatbot that emerges, there is an opportunity to build a chatbot as conversational recommender system based on functional requirement.

III. Chatbot Requirement and Design

In this part we analyze the capabilities needed on the chatbot to be a conversational recommender system. There are five capabilities that the chatbot need to fulfill:

1) Natural Language Understanding (NLU): Users are interacting with chatbot in a form of natural language. The chatbot we built must be able to understand message from user and extract the information needed. NLU capabilities for the chatbot can be achieved via intent classification and entity extraction. This is a common technique used to build NLU capabilities for a chatbot.

2) Keeping Conversation Context: When conversing with the chatbot, user expects to have a natural conversation. One of the characteristics of a conversation is conversation context. To be able to converse well, the chatbot must be able to keep context and responds according to context. The method chosen to implement this capability is the concept of input and output context. Input context is a context needed to activate a certain user intent. Output context is the context activated when certain user intent is received. With this method, the chatbot can converse according to context that is currently active.

3) Do Action: Chatbot needs to perform an action as a response to user input, such as writing message back. List of action the chatbot can perform is defined explicitly. Action to be performed by the chatbot is determined by the received intent. For every intent, actions that will be executed if that intent is received is defined. For one intent, the action defined can be more than one. The action executed can depends on whether an entity is present in the user message. For every action, a program function is defined. This function will be called if the action is executed. The function can contain any set of program action. In this function we define the action according what we want the chatbot to do.

4) Retrieving Product for Recommendation: The chatbot's knowledge of the product is given via ontology. The ontology used is ontology from the work by Baizal et al. [1]. To retrieve needed information from the ontology, there are different methods that can be used. One of the methods is parsing the ontology file and store the information in common data structure e.g. JSON or native data structure e.g. string. From then we can process the data. Other method is to run a SPARQL server with a dataset from the ontology that we are going to use. We can get the information we need by executing SPARQL query to the server.

The first method does not require us to run an additional server, but it need more effort to implement. In the other method we need to add additional server, but we can get information more easily via SPARQL query. For this chatbot we will use the second method.

5) Giving response to user: To generate a sentence as a response to the user, there are different methods that can be used. The simple method is to use template sentences. To generate the response sentence, one sentence is randomly taken from a list of template sentence. To make the template sentence more dynamic, the template can contain variables that can be replaced with runtime value.

Other methods include using statistical approach. This method involves the use of machine learning e.g. retrieval and generative methods. In retrieval methods, a sentence is chosen from a set of sentences by using statistical techniques, e.g. the vector similarity between the user message and the response sentence [10]. In generative methods, the bot is generating sentence by choosing word per word until it forms a complete sentence [11]. These methods have advantages, such as in retrieval, the most relevant response to the user message will be chosen, and in generative method the chatbot can generate infinite number of sentences. The disadvantages of these method are it is more complex and harder to implement. For this chatbot, the method used is by using template sentences, because it is sufficient for the function that we aim for.

The chatbot feature is implemented by defining configuration data for the chatbot. The configuration data needed for the chatbot are list of intent, action, context, entity, and template sentences. In the case of this chatbot, the feature implemented are:

1. Receiving user functional requirement.
2. Receiving product filter, which consist of brand and price filter.
3. Removing last added requirement.
4. Resetting all requirements and filter.
5. Gives product recommendation based on requirements and filters.
6. Receive feedback for the recommended product.

To implement the feature, we define 16 intents, 23 actions, 7 contexts, 3 entities, and 25 template sentences. The example of defined configured data can be seen on Table I-V.

IV. CHATBOT IMPLEMENTATION

The system made into components based on required chatbot capabilities. The system is built into components so that every component can built independently i.e. we can build a component with techniques or method of our choice without affecting the other components. The architecture of the chatbot is given on Figure 1. Each of the components of the chatbot will explained below.

A. NLU Component

The capabilities implemented in the NLU component are intent classification and entity extraction. Configuration data needed by the NLU component are:

1. List of intent
2. List of entity
3. List of possible values for the entities and the synonyms

We also need to define training data for this component. The training data consists of text representing message from user and its label. The label consists of the message intent and entity contained if exist.

With input of user message in form of a string, the output of this component is intent for the user message (intent with highest confidence level), entities extracted, and the rank of confidence for other intents.

TABLE I. LIST OF INTENT

No	Intent name
1	ask-recommendation
2	add-requirement
3	add-requirement-yes
4	add-requirement-no
5	cancel-last-requirement
6	give-feedback-positive
7	feedback-positive-yes
8	feedback-positive-no
9	give-feedback-negative
10	other-recommendation
..	…

TABLE II. LIST OF ACTION

No	Action name
1	start-recommendation
2	ask-requirement
3	ask-repeat-requirement
4	receive-requirement
5	check-constraint-fitness
6	delete-last-requirement
7	reset-requirement
8	receive-feedback-positive
9	receive-feedback-negative
10	retrieve-product
..	..

TABLE III. LIST OF CONTEXT

No	Context name
1	on-recommendation
2	recently-adding-requirement
3	add-requirement-followup
4	recently-got-recommendation
5	got-recommendation
6	add-requirement-followup
7	feedback-positive-followup

TABLE IV. LIST OF ENTITY

No	Entity name
1	functional-requirement
2	brand
3	price

TABLE V. LIST OF TEMPLATE SENTENCES

No	Template name
1	start-recommendation
2	ask-requirement
3	ask-repeat-requirement
4	receive-requirement
6	delete-last-requirement
7	reset-requirement
8	receive-feedback-positive
9	receive-feedback-negative
11	give-recommendation
..	..

For this chatbot, the NLU component built with Rasa NLU. Rasa NLU provides pipeline for intent classification and entity extraction. The pipeline used for this chatbot is tensorflow_embedding, which consist of tokenizer_whitespace component to tokenize words, ner_crf component for extracting entities, ner_synonims to process entity synonyms, intent_featurizer_count_vectors for extracting word vector, and intent_classifier_tensorflow_embedding for intent classification. With Rasa NLU, we perform a training on the training data to produce a model. The model will be used for intent classification and entity extraction.

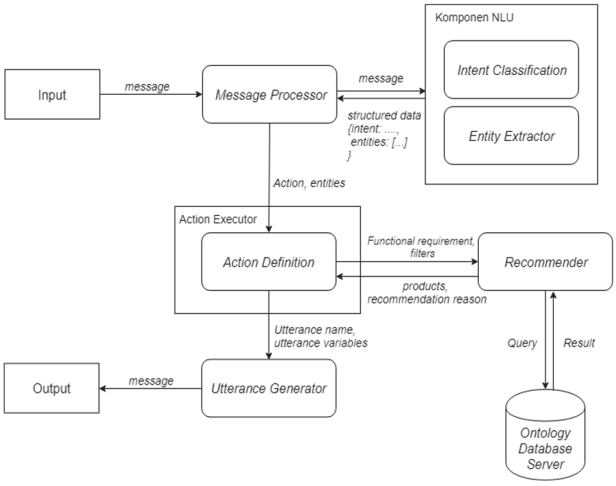

Fig. 1. Chatbot component architecture

B. Message Processor

Message processor is the first component that receives user message. This component will forward the message to NLU component and obtains intent and entity information. Then it will decide action that must be performed and calls action executor component to execute the action.

Data configuration needed for this component are list of intent, list of action, and list of contexts. For each intent we define the name of intent, input context, output context, action, and entity action for the intent. As mentioned in Section III, input context is the context needed to receive the intent. Output context is the context activated if the intent is received. Action is the action executed if the intent received, but if the message is containing entity, then entity action will be executed instead.

Message processor keeps track of list of contexts that is active. In this component also defined the threshold for intent confidence and entity confidence. This threshold is used to filter out intents and entities with confidence below the threshold. This component then will determine the matched intent, intent with highest confidence. All input context for that intent also needs to be active currently. If there are no matched intent, then the chatbot will use default intent, the fallback intent.

After the matched intent is determined, contexts that is listed on the output context of that intent will be activated. Every context that is previously activated that is not labelled permanent will be deactivated. If the message contains no entity, then the action listed on the intent action list will be executed. Otherwise the action listed on the intent entity action list will be executed. The action is executed by calling the action executor component. If the intent has no action defined, then the chatbot will do nothing. If the intent has multiple action defined, then every action will be executed in order.

C. Action Executor

Action executor component is responsible for executing action. In this component we define a program function for every action that is defined in the configuration data. The function will be called if the action is executed. The body of the function can contain algorithm, steps, or calls function from another component. For actions that generate response to the user, the action will call function from the utterance generator component. To the action related to the recommendation, the action will call function from recommender component.

D. Recommender

In this component we define functions that is related to product recommendation e.g. functions to get list of products, get a product from the list of products for recommendation, and get the reason of a product recommendation.

Function to get the list of products takes list of user functional requirement and filters. This function retrieves and stores list of products that satisfy the user requirement and filter. The list of products is taken from the ontology. To retrieve the list of products that satisfy user functional requirement and filters, this function executes a query to the SPARQL server that has the ontology as dataset.

The function to get a product for recommendation retrieves one product from the list of products that is previously retrieved. If the chatbot needs to give another recommendation, the function will retrieve the next product. The function to give recommendation reason will generate a reason consisting of the product technical specification that fulfil user functional requirement. This information is obtained by executing a query to the SPARQL server.

E. Utterance Generator

The chatbot generates sentence as a response for the user by using template sentences. Configuration data defined in this component is list of template sentences. A template sentence

consists of the name of the template and the list of sentences that will be picked randomly as chatbot response. A sentence on the template can contain a variable that will be dynamically replaced by runtime values such as a name of entity.

When generating a response, utterance generator receives a name of template and variables needed by template as the parameter. The component then will retrieve the template and choose one sentence randomly. The variables inside the sentence, if any, will be replaced by the received values. The sentence then will be sent to the user as a response.

V. CHATBOT EVALUATION

The purpose of the evaluation is to evaluate whether the chatbot is able to give similar experience to the user compared to experience when taking a recommendation in real life e.g. from friends or salesperson. The test also evaluates user experience in adding a functional requirement and performing tasks on the chatbot. The test is performed using usability testing method. Usability goals aimed in the test is easy to use and easy to remember. The usability goals are chosen to align with the purpose of the evaluation, that is to evaluate the ease of performing tasks on the chatbot. With usability testing as the method, the test involved participants. Each participant did the test procedures, which are:

1) Filling the pre-test questionnaire: In this phase the participants asked to answer questions related to participant profile and behavior in using related technologies e.g. smartphone, messenger apps.

2) Exploring the chatbot: In this phase the participants are given brief explanation about the chatbot and then given the chance to use and explore the chatbot.

3) Completing the tasks on the chatbot: In this phase the participants are asked to do tasks on the chatbot. Participants are given 12 tasks. The task given are related to the chatbot feature, that is adding functional requirement, adding filters, erasing requirements, and giving feedback to the product recommended.

4) Filling the post-test questionnaire: The post-test questionnaire is the main source of participant feedback related to the chatbot evaluation. Participant are asked about their experience in using the chatbot and performing task relating to the usability goals (easy to use and easy to remember). The participant also asked how much the user feels that the experience on the chatbot is similar to the experience of the recommendation process on daily life.

The number of participants participating in this test is 10 people. There are 6 group task that is evaluated by the user, multiplied by the number of participants, so there are total of 60 feedback received. The evaluated aspect and the answer choice of the post-test questionnaire is given on Table VI and VII. The result of the post-test questionnaire is given on Figure 2-4.

Overall the feedback received from the user is relatively good on the evaluated aspect, which are user perspective of similarity to daily life recommendation process e.g. with salesperson or friends, the ease of adding functional requirements, and the ease of performing certain task on the chatbot.

TABLE VI. POSTTEST QUESTIONNARE RESULT FOR USABILITY GOALS ASPECT

Aspect	Answer Choice
Easeness in performing tasks	Able to perform the task with ease
	Able to perform the task
	Initially confused but finally able to perform the task
	Unable to perform the task
Easeness in remembering how to perform tasks	Able to remember with ease
	Able to remember but needs help
	Unable to remember

TABLE VII. POSTTEST QUESTIONNARE ANSWER CHOICE FOR THE CHATBOT RECOMMENDATION USER EXPERIENCE

Aspect	Answer Choice
User perspective of similarity to daily life recommendation process (1-5 scale)	5 (Very similar)
	4
	3
	2
	1 (Not similar at all)

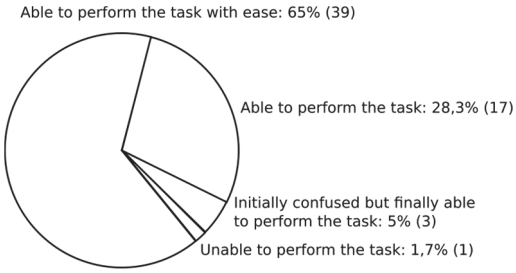

Fig. 2. Result for easeness in performing tasks aspect

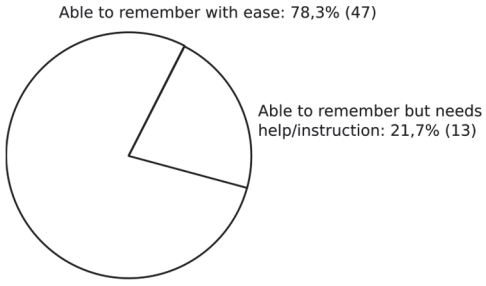

Fig. 3. Result for easeness in remembering how to perform tasks aspect

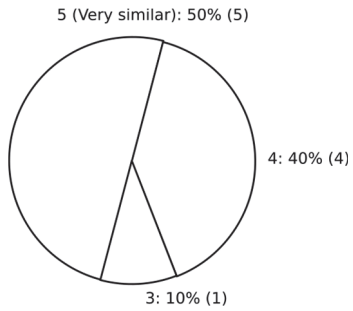

Fig. 4. Result for user's perspective of similarity to daily life recommendation process aspect

However, based on observation, sometimes the chatbot still fail or incorrectly recognize the user intents. This problem can be addressed by adding more training data or improving the NLU pipeline.

VI. Conclusion and Future Works

In this paper we built a conversational recommender system chatbot that collects and gives recommendation based on user's functional requirement. The chatbot that we built consist of 5 component, which are component to get information from user message (NLU component), component to process user message and keeping contexts (message processor), executing action (action executor), retrieving information related to product recommendation (recommender component), and generating sentence as chatbot response (utterance generator). The feature of the chatbot is implemented by defining the configuration data, which are list of intent, action, context, entity, and template sentences.

The usability testing performed on the chatbot receives a relatively good result from the participants. The aspect evaluated is the similarity of the user experience with the experience when getting a recommendation in daily life from other people, the ease of adding functional requirements, and the ease of performing certain task on the chatbot.

For the upcoming works, there are some areas for improvement that can be made. The NLU capabilities can be improved, i.e. improving the intent classification and entity extraction accuracy. To do this, we can try to use different techniques or tools to implement NLU capabilities, add more training data, or perform further model optimization. We can improve the chatbot capabilities in generating response to the user, by using different techniques or method. The chatbot interaction can be improved by using user experience techniques e.g. user-centered design to design the interaction.

References

[1] D. H. Widyantoro and Z. K. A. Baizal, "A framework of conversational recommender system based on user functional requirements," 2014 2nd International Conference on Information and Communication Technology (ICoICT), Bandung, 2014, pp. 160-165.

[2] Bridge, D.G. (2002). Towards Conversational Recommender Systems: A Dialogue Grammar Approach. ECCBR Workshops.

[3] M. S. Aktas, M. Pierce, G. C. Fox and D. Leake, "A Web based conversational case-based recommender system for ontology aided metadata discovery," Fifth IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA, 2004, pp. 69-75.

[4] Wärnestål, P., Degerstedt, L., & Jönsson, A. (2007, May). Interview and delivery: Dialogue strategies for conversational recommender systems. In Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007) (pp. 199-205).

[5] Z. K. A. Baizal, D. H. Widyantoro and N. U. Maulidevi, "Query refinement in recommender system based on product functional requirements," 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Malang, 2016, pp. 309-314.

[6] Ramachandran, D., Fanty, M., Provine, R., Yeh, P., Jarrold, W., Ratnaparkhi, A., & Douglas, B. (2015, September). A TV program discovery dialog system using recommendations. In Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (pp. 435-437).

[7] Holotescu, C. (2016). MOOCBuddy: a Chatbot for personalized learning with MOOCs. RoCHI.

[8] Colace, F & De Santo, Massimo & Pascale, Francesco & Lemma, Saverio & Lombardi, Marco. (2017). BotWheels: a Petri Net based Chatbot for Recommending Tires. 350-358.

[9] Ikemoto, Yuichiro & Asawavetvutt, Varit & Kuwabara, Kazuhiro & Huang, Hung-Hsuan. (2018). Tuning a conversation strategy for interactive recommendations in a chatbot setting. Journal of Information and Telecommunication. 1-16.

[10] Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. arXiv preprint arXiv:1711.01731.

[11] Li, Jiwei. "Teaching Machine To Converse" , Department of Computer Science and the Committee on Graduate Studies of Stanford University, 2017.