Approaching natural conversation chatbots by interactive dialogue modelling & Microsoft LUIS

Alexander Wachtel, Dominik Fuchß, and Sophie Schulz
Education Support Centre Deutschland GmbH
AI Research Lab
Karlsruhe, Germany
{wachtel, fuchss, schulz}@ESCde.net

Prof. Dr. Walter F. Tichy
Karlsruhe Institute of Technology
Chair for Programming Systems
Karlsruhe, Germany
walter.tichy@kit.edu

***Abstract*—In this paper, we preset guidelines on how to model a dialogue-based conversation with a chatbot build on Microsoft conversational services. Therefore, we created a free framework which manages dialogues and user sessions. For this purpose, we use a systematic classification of the input and intents. Especially, we used Microsoft Language Understanding Intelligent Service (LUIS) for intent extraction. LUIS provides a platform to extract the user need by calculating the intent probability. Unfortunately, LUIS does not provide dialogue management. For that reason, developers need to create own dialogue behavior in Visual Studio project. Therefore, we extracted and use it as a free guideline for the community and other platform developers.**

***Index Terms*—Dialogue Systems; Interactive Conversation; Dialogue Modeling; Dialogue Framework; Bot Framework.**

I. Introduction

Natural-language systems have been the subject of research for many years. It has great charm to be able to communicate with computers in the same language as with humans. Natural Language can become the new user interface (UI) for all applications. Crucial for the quality of such a system is to make its behavior human-like.

If you want to use an interface that is flexible enough to handle natural language interaction a chatbot is a good choice. Chatbots have the great advantage that questions to the user are part of the natural communication via dialogues. Therefore, lack of information from vague end users can be easily handled. A major challenge in this context is to build possible dialogue flows. These should be flexibly definable but at the same time also organized and therefore maintainable.

Therefore, we provide a free framework to handle dialogues [1]. Experts may be can handle dialogue modeling on cloud platform and do not need this framework. Novices, on the other hand, may not know where to start and use with framework to build first chatbots on Microsoft conversational services.

This paper is structured as following: Section II give an introduction to the Microsoft Cognitive Services, especially LUIS. Then Section III present our framework design for dialogue modeling on Microsoft LUIS projects. Followed by the Sections IV gives a short overview of the related work. Section V discusses the need for the system. Finally, Section VI presents a conclusion and future work.

II. Microsoft Cognitive Services

Language Understanding Intelligent Service is a natural language processing and interpretation framework developed by Microsoft [2]. It allows the developer to quickly enhance existing programs with the control via natural language. This is achieved by a domain specific web service which can be used by the program and enables the user to get interpretations of given natural language input. The results are intents rated by probabilities. In the following I would like to dive a little deeper into the following topics:

- model creation
- model training
- bot integration and actions
- Advantages and Disadvantages

A. Model creation

The first step when creating a LUIS application is the modeling of the domain. The user has to specify all necessary intents and entities. Intents in this context are considered a result which the service will later output as an interpretation of the natural language input. They often model all executable commands. Entities meanwhile are parameters for these intents e.g. when booking a flight, the command might take a parameter which indicates on which date the flight should be booked. Intents may have several entities. Entities may either by user defined or predefined entities. Currently the following pre-built entities are available:

- number
- ordinal
- temperature
- date time
- dimension
- money
- age
- geography
- encyclopedia

Additionally, it is possible to create hierarchical or combined entities which consist of child entities. Thus, a more refined modeling of certain entities is possible by either grouping or specializing certain custom entities.

B. Model training

After modeling the domain, the created model has to be trained. LUIS uses a machine learning approach in order to train the model from example inputs. These inputs have to be provided by the developer for each intent. In order to get a sufficiently trained model LUIS encourages the developer to enter several inputs/examples for all intents. The same is true for entities which are not predefined. Each utterance (as an input is called) has to be submitted to the system. After all utterances have been submitted the model may be queued for training. It is possible at any time to provide more utterances in order to improve the model or clarify of ambiguous inputs.

C. Patterns

LUIS provides a simple dialogue component which enables the developer to ask for missing entities. When creating the model an entity may be marked as necessary and annotated with a corresponding question. Given an intent is fairly high rated but missing the obligatory entity the system may then output the annotated question in order to get the missing information. LUIS then appends the next given input to the previous one and thus provides the missing information. This step may be repeated for several entities if necessary.

D. Bot Integration and Actions

Luis additionally allows binding actions to certain events. When triggered the action bound to the intent will execute. This way the developer is able to call basically every Web-Service (REST-API, JSON-Request, ...) available on the internet. For example, you may want to find all flights offered by Lufthansa for a certain day.

III. Framework Design

To integrate LUIS in our project, we designed a new structured dialogue framework based on the *Microsoft Bot Framework* and the dialogue component of it [3]. The most important aspects of this framework are the maintainability and extensibility. For this reason, we separated the LUIS classifiers to make them more exchangeable. Moreover, our framework provides structures to design an interactive dialogue between user and bot. This is enabled by several dialogue types and random responses, which can be set during bot development.

A. Requirements of a chatbot

a) Classification of user request: The interpretation of user input is a key skill for a properly performing chatbot. For classification, we use several LUIS instances to classify the chat bot input within several categories. The LUIS classifiers are trained before execution in three categories. During execution the three instances handling requests in parallel. The first category distinguishes the different topics of the bot. By this classification a context is build and the bot adapts to the user's context. In our realization we also decide between small talk and featured topics. Thus, our LUIS instance includes small talk (e.g. weather) as well as featured topics (e.g. event). A second LUIS instance classifies question types. According to this information the bot recognizes what information is searched. By using the results of classification, the appropriate course of events can be chosen within the topic dialogue. This enables concrete and correct answers to user requests. To react on special user actions a third LUIS classifier has to be trained. In our setting those actions include cancellations as well as confirmations and rejections. If desired, they can be expanded by other fixed commands (e.g. log on/ off) during development. In avoidance of false classification results we chose a threshold. If the threshold is not reached, the system must declare its incomprehension. Otherwise, the results are used to choose the correct topic and dialogue.

b) Topics and dialogues: Based on this multi-level classification, the chatbot builds dialogues and contexts. The combination of topic and question classification allows distinguishing different types of requests. This makes it easy to classify a sentence such as "When does the event take place?" In our setting, LUIS identifies the classification results: "Event" as topic and "When" as question type. Thus, it is clear that the user asked for the time of an event and the correct conversational flow can be chosen.

In our approach it makes sense to choose the topics broad. If a topic is divided fine-grained and mapped directly to intents in LUIS, the bot quickly runs the risk of ambiguous classification. Therefore, detailed classification of the actual request within the topic is done by a further step of the classification to distinguish within the topic. In addition to the classification process through various classifiers, we also designed the dialogues dynamically. Every dialogue consists of a course of steps, which represent tasks to be completed. Those steps are arranged dependent of user input and reactions. In contrast to the *Microsoft Dialog Framework*, our framework enables to choose processing steps at run time - not only to define them. To support the maintainability of the software we kept the modeling of individual steps, known from the Dialog Framework.

c) Small talk: In context of chatbots, small talk refers to requests from a user that have no relation to the featured functionality of the bot. These typically include questions about the weather, age or origin. In our framework we generally distinguish between two types of small talk. First, the *single step small talk dialogues*, that are requests which are answered by a response from the bot. After this answer, the dialogue is automatically closed. This avoids effort to define every single subject in this context. The answers should be mostly generic to respond to different formulations. In order not to appear too stiff, we recommend providing different answer possibilities. In our setting we selected randomly one of the predefined answers. On the other hand, the *multi step small talk dialogues* simulate a conversation with the user. This is useful to act more like a human. An example of this is the question "How are you?". A single step small talk would be misplaced and rude. On such questions it is often neater to ask a counter question: "I'm fine. And you?". After that the dialogue can be continued and redirected naturally: "Back to topic. Do you have any questions about the events?".

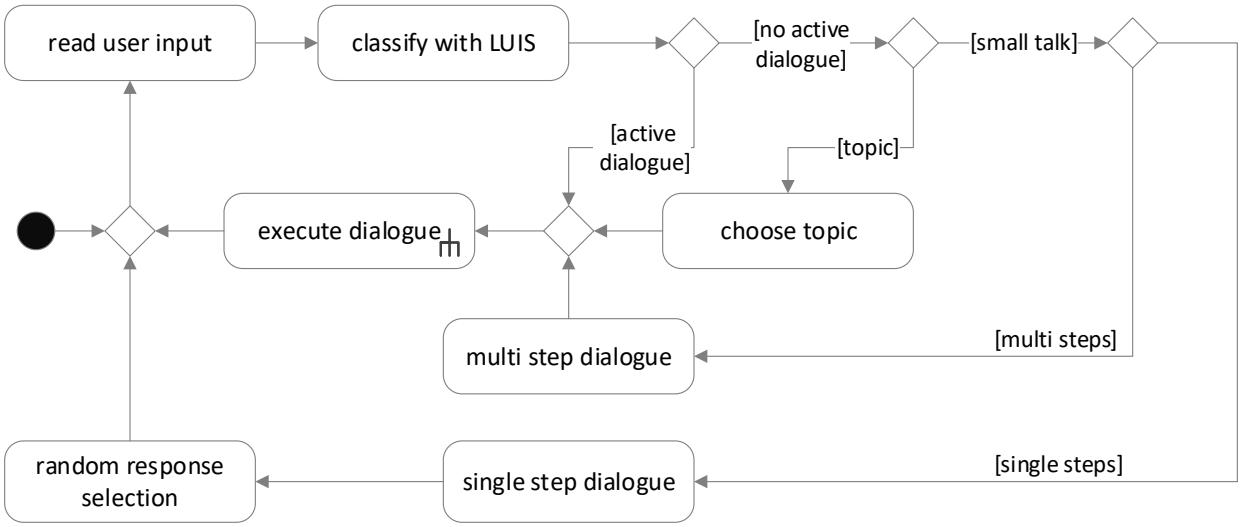

Fig. 1. Dialogue management of the chatbot

B. Dialogue management

Dialogue management is an important task when building a chatbot. Fig. 1 shows our realization of that. The activity diagram always starts when the user input is read in. This must first be interpreted by the chatbot. At first, the three classifications described in subsection III-A classify the user input. Then, the bot checks its state for an active dialogue, which waits for a reaction from the user. If an active dialogue is found, it is executed. Elsewhere, the bot starts a new dialogue for the classified topic or small talk. For this purpose, we first classify the input with LUIS using our three classifiers. Afterwards, it must be decided whether the control flow must be passed to an old, still active dialogue or to a new dialogue. An "active dialogue" is a dialogue that waits for a required user input. When a new input has been read in, the old dialogue is continued and the further steps are performed. If there is no active dialogue, a new dialogue must be started. There are two different types of dialogues. On the one hand the small talk dialogues and on the other hand the topic dialogues. Topic dialogues are the feature dialogues of the bot. They are mapped to intents and represent functionality of the bot.

Small talk dialogues are divided into one step dialogues and multi-step dialogues. One step small talk dialogue offers the possibility to choose one randomly from a set of predefined template answers. Afterwards, no further step is awaited. Therefore, after the random answer is submitted to the user, the interaction is back at the beginning and the user can enter new input. Multi step small talk dialogues are executed as in the case of an active dialogue or a topic dialogue.

The execution of a multi-step or topic dialogue is shown in Fig. 2. A dialogue consists of a set of steps. These can be partially processed and partially still to be processed. Each dialogue contains an initial set of steps to be completed. Steps can be added while executing the dialogue. As you can see in Fig. 2 the current step is executed first. After that the dialogue steps will be updated. If a user input is required for the next step or if there are no more steps to be processed, the execution is terminated. Otherwise, the next step is executed. Thus, dialogues can be built dynamically in our framework. As a result, this kind of dialogue management enables the bot to interact with the user in a more natural way than with fixed flows.

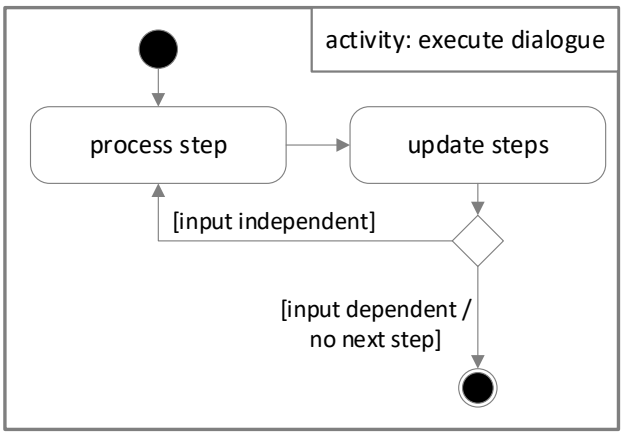

Fig. 2. The concept of dialogue execution

IV. Related Work

Similar to LUIS, Amazon Voice Service (AVS) [4] uses skills that describe the intentions. AVS works according to the following slot-filling approach, in which for a successful execution all parameters must be present and with a missing one parameter this is queried by the system.

The implementation of Google is called DialogFlow [5] and provides (i) an interface ready to speech understanding the end-user inputs in written language, (ii) a modeling of a dialogue and (iii) integration of the agent into common communication channels allowed. The following are the individual solution steps in the Detail described.

While with LUIS and Alexa the point (i) and point (iii) also possible (ii) the modeling of a dialogue within the Dialogue Flows is a difference. In LUIS modeling is done by one Developer in the bot framework, however, needs a developer of the dialogue less programming knowledge with Dialogflow. This is done by setting the answers, which, like the utterances - also called training phrases in Dialogflow - are entered as complete sentences become. When classifying an intention to end user input these answers are random, yet unchanged.

Many dialogue systems have already been developed. Commercially successful systems, such as Apple's Siri, actually based on active ontology [6], and Google's Voice Search [7] cover many domains. Reference resolution makes the systems act natural. In 2015, [8] presented a survey on chatbot design techniques and compared different assistants such as Siri, Google Chrome and Cortana. The Mercury system [9] designed by the MIT research group is a telephone hotline for automated booking of airline tickets. Mercury guides the user through a mixed initiative dialogue towards the selection of a suitable flight based on date, time and airline. Furthermore, Wachtel [10] provides a dialogue solution on how algorithms can be recognized and learned from human descriptions.

V. Discussion

One of the most difficult tasks in building an intelligent performing chatbot is modeling of dialogues. LUIS classifies user input for provided intents, but has no further functionality for dialogue management after classification. In addition, users often behave differently. Therefore, it must be as easy as possible to model different dialogue flows. We approached this challenge by building a framework that forces dynamically built dialogues based on the static dialogue capability of Microsoft's Bot Framework. The Bot Framework offers the support of different messaging endpoints. Thus, it is possible to reach as many users as possible with one bot.

However, the framework we created does not only offer functionality for dialogue modeling. Moreover, it handles the administration of dialogue sessions. We make it possible to store data and context in attributes in the way programmers are used to. The framework is also designed for the use of different languages. It is also compatible to the dialogues from the Bot Framework. The most important task besides the dialogue modeling itself, is the simplification of building new bots. Therefore, many template methods are built into the framework. Thus, a developer has to write a few classes to get first results.

Following, we present our experience and a subjective opinion about the advantages and disadvantages of the presented framework. The most positive aspect of LUIS in our opinion is the ease to create models. It is pretty straight forward since you just model the commands you would like to be able to execute. The training is simple as well as you just provide some utterances on how this command may be triggered and LUIS does all the magic. Additionally, the clean separation of interpretation and execution is a clean solution to integrate natural language control into a given program. One of the first advantages of our framework is its maintainability. These are achieved by keeping the structure simple and easy to understand. The framework is easy to use due to templates. A chatbot can be created without to write much code. Moreover, the classifiers are interchangeable. As a consequence, the framework is not tied to LUIS alone. The dynamic structure of the dialogues enables a smooth and varying dialogue flow. Due to the coupling to the Bot Framework, all of its functionality can be used. This enables to integrate our framework into a lot of bots and to run it on different platforms. The loose coupling between the classifiers and the dialogue logic enables the exchange of technologies and a separate evolution of the different modules of the chatbot.

While the automated model training is easy to use and may be quite strong in a lot of cases it is hardly configurable and does not allow any exception handling by the developer resulting in a disadvantage. Secondly, the pure online solution may sound appealing but for several use cases may be far from perfect. The obvious need of an internet connection may already pose a problem in some cases, but even if provided the response times may suffer severely (this should be tested in future to provide quantitative measurements). Another outstanding problem is the manual modeling and coding of dialogue flows. In addition, the Bot Framework was chosen as the platform.

VI. Conclusion and future work

This paper proposes a free available framework for dialogue modeling on Microsoft LUIS projects. Additionally, the paper proposes some guidelines that can be adapted to any other cloud-based conversational platform. Furthermore, we will provide more pre-built dialogue models that can be used for free. This framework should empower the community to build more chatbots.

References

[1] Education Support Centre Deutschland GmbH, "Chatbot Framework," 2019. [Online]. Available: https://github.com/ESCdeGmbH

[2] Microsoft Corporation, "Cognitive Services - Language Understanding (LUIS)," 2019. [Online]. Available: https://eu.luis.ai

[3] Microsoft Corporation, "Microsoft Bot Framework," 2019. [Online]. Available: https://dev.botframework.com

[4] Amazon.com Inc., "Amazon alexa voice service (avs)," 2019. [Online]. Available: https://developer.amazon.com/en-US/alexa

[5] Google Limited Liability Company, "Dialogflow," 2019. [Online]. Available: https://dialogflow.com

[6] D. Guzzoni, "Active: A unified platform for building intelligent web interaction assistants," in *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on*. IEEE, 2006, pp. 417–420.

[7] J. R. Bellegarda, "Spoken language understanding for natural interaction: The siri experience," in *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 2014, pp. 3–14.

[8] S. A. Abdul-Kader and D. J. Woods, "Survey on chatbot design techniques in speech conversation systems," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 7, 2015.

[9] S. Seneff, "Response planning and generation in the mercury flight reservation system," *Computer Speech & Language*, vol. 16, no. 3-4, pp. 283–312, 2002.

[10] A. Wachtel, F. Eurich, and W. F. Tichy, "Programming Spreadsheets in Natural Language: Design of a Natural Language User Interface," *The Eleventh International Conference on Advances in Computer-Human Interaction*, March 2018.