Conception of a Conversational Interface to Provide a Guided Search of Study Related Data

https://doi.org/10.3991/ijet.v14i07.10137

Rene Berger, Markus Ebner, Martin Ebner(✉)
Graz University of Technology, Graz, Austria
martin.ebner@tugraz.at

Abstract—Since the beginning of software development, solution approaches and technologies have changed massively, including the requirements for a user interface. At the very beginning, it was the desktop application, with a classic Graphical User Interface (GUI), which fulfilled the needs of a user. Nowadays, many applications moved to web respectively mobile and the user behavior changed. A very modern concept to handle the communication between a computer and a user is a chatbot. The range of functions of a chatbot can be very simple up to complex artificial intelligence based solutions. This publication focuses on a chatbot solution for Graz University of Technology (TU Graz), which should support the student by finding study related information via a conversational interface.

Keywords—Chatbot; conversational interface; natural language understanding

1 Introduction

1.1 Applying the styles to an existing paper

This publication is about researching, designing, implementing and evaluating a chatbot for the TU Graz, which provides a search concept for students to simplify finding study related information. The bot should be a standalone client messenger and not integrated in one of the major messenger platforms like Facebook Messenger[1] or Slack. Although, it is a standalone messenger, it should be similar to existing ones, so that there is no confusion how to use it. The web client should also be responsive to provide a good mobile user experience. The design should adhere to the corporate identity of TU Graz. To be able to implement a standalone chatbot, a front-end and back-end solution has to be developed. Therefore, several frameworks were evaluated. To increase future maintainability, JavaScript was used on client and server side. An essential point of the bot is the communication with the TU Graz search proxy, which provides all study related data in the form of an Extensible Markup Language (XML) result. This has to be parsed and the desired data has to be extracted. It should also provide some kind of artificial intelligence to improve the user experience. To

[1] https://de-de.messenger.com/, accessed 19 April 2018

integrate a suitable artificial intelligence platform an evaluation was done. The most used and best known tools were analyzed and based on the given requirements the most appropriate one was chosen. The TU Graz Searchchatbot is personal, domain specific and follows the information based approach. It covers information about employees of the TU Graz, rooms, subjects, books and organizations. It also provides a site search of the website. For that reason a crawler was implemented, to extract the necessary data to answer the question of the user. After the implementation, a test period started, to evaluate the acceptance and satisfaction of the chatbot in comparison to the already existing search solutions.

The following three Research Questions were addressed in this study:

- How big is the general interest of a chatbot in the university area?
- Is a chatbot able to replace a conventional graphical user interface?
- Does the help in searching through a chatbot lead to more satisfactory search results than via a search form?

2 Introduction to Searchbots

2.1 Implementation of the prototype

In the last years, there is undoubtedly some kind of revolution in the software industry. In the past web and mobile applications changed the requirements of a software dramatically. The chatbot or a conversational interface is a further development to the conventional user interaction. The reason why chatbots became so popular is, that messenger applications are heavily used by people, especially in terms of mobile. Table 1 shows the usage of messenger applications in 2016, which indicates the importance of a conversational interface to expose services via a chatbot.

Table 1. Chat statistics in 2016 [1]

Network	Origin	Monthly active users
WhatsApp	US	1 billion
Facebook Messenger	US	900 millions
Viber	Israel	784 millions
Viber	China	762 millions
Line	Japan	560 millions
Instagram	US	500 millions
Kik	Canada	275 millions
Snapchat	US	220 millions (est.)
Hike	India	100 millions
Palringo	UK	40 millions

A chatbot has many advantages over a classic user interface. The user is able to directly communicate with the information system and ask for the desired information. It is no longer needed to go through multiple steps to find the information the user is

looking for. Communicating with a chatbot is also more natural then using a traditional GUI.

Back then, the first developed chatbot was ELIZA [2] by Joseph Weizenbaum. It was possible to communicate with the bot in natural language and ELIZA was able to emulate several different dialogue partners. The most successful one was a psychotherapist, which used a thesaurus to make it possible to have an ongoing conversation with the user. ELIZA was programmed to recognize keywords and to apply appropriate transformation based on context. Each keyword has special transformation rules [3]. Nowadays, chatbots are much more complex and several types are existing, as figure 1 shows. A chatbot is a combination of three subtypes. They can be personal or impersonal, domain or non domain specific and have a task, information or conversation based goal.

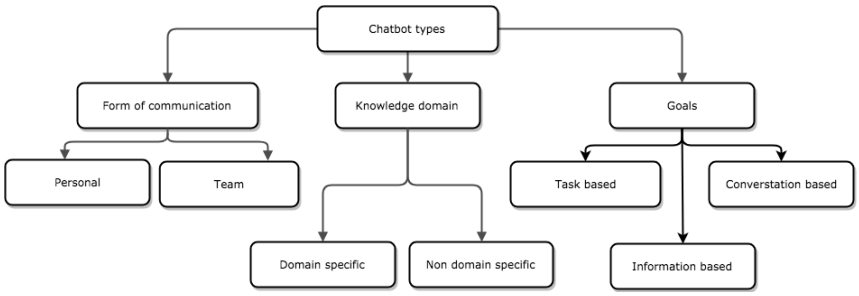

Fig. 1. Chatbot types

The difference between a personal and a team bot is the user basis. A personal bot satisfies the needs of a single user within a single context, while the team bot has to switch between multiple user inputs and is used in a shared channel. A typical example for a personal bot is a personal assistant. Team bots can be used for team organization in messenger applications. [4]

In terms of the knowledge domain categorization, there are domain and non domain specific bots. Domain specific ones are typically implemented for a single service or a specific product. It more or less represents a product or a brand. The team bot on the other hand exposes multiple services, it is a so called super bot. A very well known bot of this category is Amazon's Alexa [5]

As already mentioned there are three types of goals which can be followed. The task based bot is implemented to execute a certain task. The conversation flow is predefined and the main goal is to finish a job. However, the conversation based bot tries to communicate with the user as long as possible without executing a specific job. The main goal for this approach is an ongoing conversation with the user. Last but not least, the information based bot provides information to specific topics. The conversation should be short and purposeful. A typical example for this category is a Frequently Asked Questions (FAQ) bot. [5]

Those types can be applied for business to business as well as for business to consumer bots, although they have different objectives. A business bot is goal-driven, the conversation flow should be short and jobs should be executed very easily. The con-

sumer bot has a different approach how to communicate; it is more personal can be also off topic or just entertaining. Often the main goal of a consumer bot is to stay in touch with a brand.

Two major platforms have emerged to offer chatbots. The best platform for the consumer to business approach is the Facebook Messenger. This messenger provides an easy to use Application Programming Interface (API) for bot interaction. It is available for mobile and web and it is very easy to get in touch with potential customers. The most known business messenger is Slack It is widely used for business bot integration such as bots for time tracking or project management support.

As already mentioned, the work developed had the goal to implement a search chatbot for the TU Graz website. It should cover all the features of the already existing TU Graz search mobile application, but should expose the service via a conversational interface.

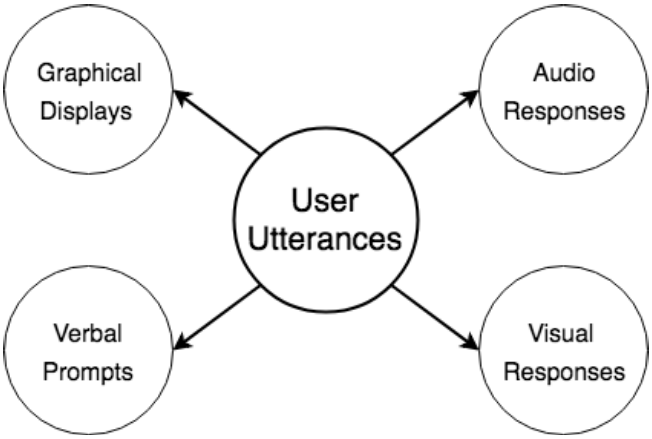

Fig. 2. Types of user utterances, based on [6]

A conversational interface should be as natural as possible, so that the user does not have to adapt his/her behavior when using the TU Graz Searchchatbot. To guarantee that, the onboarding phase should be very clear to the user. A bot is able to contribute to a conversation with for different types of interactions as figure 2 shows. The TU Graz Searchchatbot follows the graphical display approach to interact with the user. [6]

For providing artificial intelligence, dialogflow is used. Dialogflow, former api.ai, was launched in 2010 and acquired by Google in 2016. It parses the query and returns a JavaScript Object Notation (JSON) object with the most suitable intent based on the information stored in the intent. In addition, several other artificial intelligence platforms were evaluated, especially wit.ai. In general, there are consumer and business platform tools. Dialogflow and wit.ai are belonging to consumer tools; an example for business tools is Watson. Dialogflow was chosen because of its rich feature set, it is tested over eight years now and it delivers good intent matching results. Furthermore,

the pricing model of Dialogflow fits for this project, because it offers free usage of text queries.

Dialogflow consists of five main parts namely agents, intents, entities, context and fulfillment. Agents are the natural language understanding (NLU) modules, which is the starting point of your application. To recognize what the user wants, the intent matching comes into play. To match an intent Dialogflow needs data to train a machine-learning model. The more data you provide the better is the intent matching, although Dialogflow not just understands the phrases you have entered it also matches phrases which means the same thing. To identify and extract information a user mentioned the entity matching is needed. Dialogflow provides build-in entities, such as date, time and geo-state. This is a good starting point but with high probability, you need to define your own entities when developing a chatbot application. Same as for the intent matching, also for the entity matching training data has to be added.

Context plays vital role in the success of a chatbot conversation. It helps the chatbot to talk more like human by answering within a context in a linear and non linear dialog. In general, as long as there is no fulfilment of the user's needs, the context stays the same.

2.2 Architecture

The TU Graz Searchchatbot application is a full stack standalone web solution. Therefore, it consists of four parts which are:

- Single Page Application Client
- Back-end/Middleware
- TU Graz search proxy
- Third party NLU platform for artificial intelligence support

To outline how the bot works, figure 3 illustrates the final architecture of the application. Dialogflow does not support a PHP SDK as several other platforms. Due to that fact, Node.js[2] was used for the back-end. Therefore, the same programming language could be used on front-end and back-end, which is also a benefit for maintainability.

The basic flow works as follows. The user starts the bot by visiting the website and receives a valid session token. After that, the user sends a message and the session token to the Node.js back-end. For evaluating the correct intent, the Node.js back-end passes the message to the Natural Language Understanding (NLU) platform, which is Dialog flow. It will respond with a JSON object to the Node.js back-end with all the necessary information to perform the search, which are the matched intent, the extracted entities, the context and the fulfillment state. For example if a user enters a phrase like "Do you have contact information about Martin Ebner?", the Dialog flow agent respectively the NLU agent will match the intent Contact Information, with the extracted entity @sys.name Martin Ebner. That information will be passed to the

[2] https://nodejs.org/en/, accessed 24 April 2018

Back-end/Middleware if the fulfilment of the dialog is achieved. Otherwise a linear or non-linear dialog will be performed.

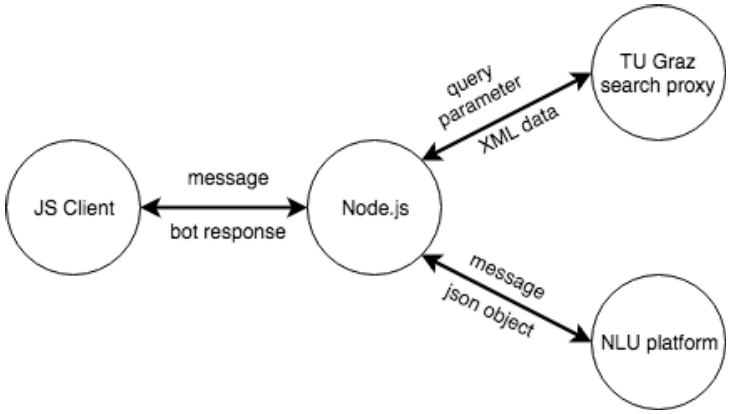

Fig. 3. Final architecture

The search itself is done by the TU Graz search proxy. It responds with a list of found items in a XML data format. After receiving the data, the Node.js back-end parses the items, extracts the desired information and returns it to the front-end. The client handles the response data and displays the message to the user. Figure 4 shows an example communication with the chatbot.

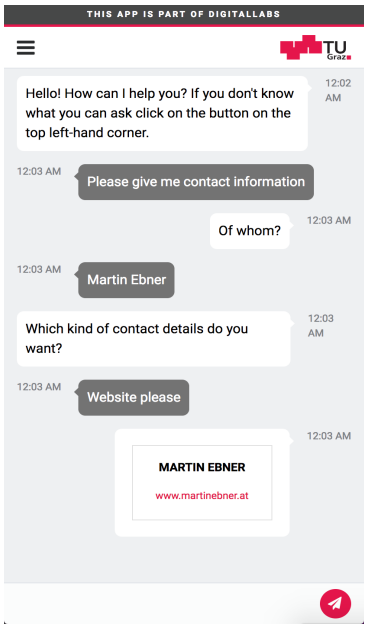

Fig. 4. TU Graz Searchchatbot

2.3 Evaluation of client

To ensure a pleasant user experience a Single Page Application (SPA) was developed. To accomplish that, JavaScript front-end frameworks were evaluated. Table 2 gives an overview of the analyzed candidates.

Table 2. Comparison of front-end frameworks

	Angular	React	Vue
Publisher	Google	Facebook	Vue Technology
Programming Language	Typescript	Javascript	Javascript
Componend based	Yes	Yes	Yes
State management	ngrx	Redux	Vuex
CLI	angular-cli	-	vue-cli
Integrated router	angular/router	Only external	Vue-router
CSS modules	Yes	Has to be configured manually	Yes
Separate HTML/JS	Yes	No	Yes
Official style guide	Yes	No	Yes

Angular, React and Vue are the most popular Javascript frameworks nowadays. Every framework has its benefits. In terms of rendering performance, React is the framework to choose. It is blazing fast and optimized, but the biggest disadvantage is, that many third party dependencies are needed to have a full framework tool set. Vue is a lightweight framework, which supports many features out of the box, like a router, state management and a Command Line Interface (CLI). It is a well designed framework for small to medium sized projects. In comparison to React, Vue is more a framework, however React has library characteristics. The third candidate in this comparison is Angular. It is the most stable and maintainable framework in relation to Vue and React. It has an official style guide, which explains exactly how a project should be structured and implemented. Due to that, the entry into an Angular project is straight forward. The framework is based on Typescript, hence a type safe implementation increases the code quality. Angular has also a clear separation of logic. This framework provides modules which are wrappers for components and services. Services are containing business logic, however components are responsible for the representational logic. React and Vue has also some kind of separation of logic, but Angular handles it in a more structured way than the other two. All three frameworks have a state management system. Initially the state management was developed by Dan Abramov in the form of Redux. Angular supports a fork of this implementation, namely ngrx. Vue provides also a state management system with Vuex, which is a fork of redux as well. Summarizing all aspects, Angular was chosen for the client implementation. [7], [8], [9], [10]

2.4 Evaluation of back-end

With an increase in importance of JavaScript, Node.js frameworks are very common nowadays. Before comparing frameworks let's take a quick look what Node.js is. Node.js is a server-side JavaScript platform, based on the Google Chrome V8 engine.

It is a big advantage for JavaScript developers to implement a full stack solution, without switching the programming language. They are serious alternatives to Symfony or Laravel, which are based on PHP. There is one Node.js framework, which stands out in terms of acceptance by the community. It is called Express.js[3] and it was published in 2010. For this publication also another Node.js framework was evaluated, namely Hapi.js. Hapi.js was built by Walmart to alleviate issues occurred while using Express. [11]

Hapi.js, which stands for HTTP API, provides a lot of features out of the box like authentication, caching, validation and more. It is also stress tested under a realistic production atmosphere, and it exists a test coverage of hundred percent. Hapi.js is in comparison to Express more configuration centric and the learning curve is steep. Express has a lightweight minimalistic approach, based on the core Node.js http module and connect components which are called middleware. The philosophy of Express is configuration over convention. Due to the huge community support, there are many additional features available. [12], [13]

Summarizing it can be said that Hapi.js is the better framework for enterprise application. For the TU Graz Searchbot, where the back-end acts more or less as a middleware the minimalistic approach of Express is the better option.

2.5 Feedback

The TU Graz Searchchatbot was available from 01.09.2017 to 28.02.2018 on DigitalLabs. DigitalLabs is a platform of the TU Graz for evaluating applications and tools. The chatbot was deployed on a separate route and was ready to use after a user hits the Uniform Resource Locator (URL). No user related data were stored.

After the go-live of TU Graz Searchchatbot, the evaluation phase started. A feedback form was integrated to be able to make a statement about the bot. Only student users participated. Among others, following questions were asked:

- How satisfied where you with the Searchchatbot?
- Which search concept would you generally prefer in the future?
- Do you think that the application / the Searchchatbot persist in the long term?

How satisfied where you with the Searchchatbot? The result to this question indicates a positive signal for the TU Graz Searchbot. More than a half of the participants are in some way satisfied. Considering that a conversational interface is a new approach to communicate with the user, it is a promising result. Table 3 shows the result in detail.

Which search concept would you generally prefer in the future?: As table 4 shows, that almost half of the users are interested in the bot concept. As already mentioned, people are not used to chatbots and therefore it may need further testing phases to optimize the user experience to convince other users.

[3] http://expressjs.com/de/, accessed 23 May 2018

Do you think that the application / the Searchchatbot persist in the long term?: As table 5 shows, 58.33% of the users appreciates the bot concept and are of the opinion that the TU Graz Searchchatbot should be an additional solution to the current search solution.

Table 3. Result of "How satisfied where you with the Searchchatbot?"

Answer	Percentage
Very satisfied	0
Satisfied	25
Rather dissatisfied	33.33
Dissatisfied	41.67

Table 4. Result of "Which search concept would you generally prefer in the future?"

Answer	Percentage
Chatbot	8.33
Searchfield	50
Chatbot and Searchfield	33.33
Nothing	8.33

Table 5. Do you think that the application / the Searchchatbot persist in the long term?

Answer	Percentage
Chatbot	8.33
Searchfield	50
Chatbot and Searchfield	33.33
Nothing	8.33

3 Discussion

The feedback of the Searchchatbot indicates positive signals. Due to the fact that the Searchchatbot is in the prototype phase, there are many things to improve. Basically we analyzed which improvements can be done by the implementation of the Searchchatbot, and which improvements are related to the API of the search proxy.

In terms of natural language processing there will be some improvements as well towards dialogflow. At the moment dialogflow is based on a decision tree. Machine learning is supported in terms of given examples, but if a user asks a question which the chatbot cannot handle, there will be no intent recognition improvement if the user asks exactly the same question again. The simple reason for that is, that there is no artificial intelligence providing that kind of learnings. Due to that, there have to be more feedback iteration phases to analyze the given user input. After every iteration, patterns for intent recognition can be adapted and more training data could be added, to improve the usefulness of the bot. Already the first iteration will offer a big improvement as a lot of new questions could be added to the trainings dataset. Another

evaluation of a separate classifier like Watson Classifier[4] could be done, to recheck if there would be an improvement in terms of intent matching and entity extracting.

The biggest advantages in terms of user experience would be additional information about study related data. The TU Graz search proxy should continue to be used, but there have to be other possibilities to retrieve data. Providing an API which is dedicated to that purpose would be required.

4 Conclusion

The aim was to build a chatbot, which supports the student by finding study related information. A standalone solution has to be developed, which was done with Angular on the front-end and Express on the back-end side.

To support natural language understanding, several platforms was compared and evaluated. Finally, dialogflow was chosen, because of the good results of its intent recognition.

Chatbots will become more popular in the future, therefore the Searchchatbot is an interesting first step to provide such an application for TU Graz. NLU tools are becoming smarter blazing fast, so there will be improvements expected very soon. This means that also the user experience of the Searchchatbot will increase in terms of intent recognition.

This prototype showed a possibility of a searching solution via a chatbot. The feedback of this implementation indicates positive signals to continue with this concept. Summarizing it can be said that, there is an acceptance and an interest of it but since this is only a prototype there is room for improvement. The bot has also an experimental feature on board, namely voice recognition, which should be activated in the next implementation iteration.

The current search implementation with a search form can not be replaced with a chatbot at the moment, therefore more data API's have to be provided to increase the user experience and the meaningfulness of the bot. For search results about personal data, the bot provides good results. People were satisfied with the guided search and got their desired data with less effort than with a conventional search behavior. In other search areas the satisfaction of the search result varies. To improve that, more test phases must be carried out and based on that adoptions must be made.

5 References

[1] Munford, M. (2016) How chat apps are transforming the global conversation. BBC.
[2] Weizenbaum, J. (1978) Die Macht der Computer und die Ohnmacht der Vernunft; Suhrkamp Verlag.
[3] Bob Heller, Mike Procter, D.M.L.J.B.C. (2005) Freudbot: An Investigation of Chatbot Technology in Distance Education. EdMedia: World of Conference of Educational Media and Technology. Association for the Advancement of Computing in Education (AACE).

[4] https://www.ibm.com/cloud/watson-natural-language-classifier,accessed 3.08.2018

[4] Shevat, A. (2017) Designing Bots. Creating Conversational Experiences, first edition ed.; O'Reilly Media.
[5] Nimavat, K.; Champaneria, P.T. (2017) Chatbots: An overview Types, Architecture, Tools and Future Possibilities. IJSRD - International Journal for Scientific Research and Development.
[6] Crangle, C.E. (1997) Conversational interfaces to robots. Robotica. https://doi.org/10.1017/S0263574797000143
[7] Kunz, G. (2016) Mastering Angular 2 Components, first edition ed.; Packt Publishing Ltd.
[8] Banks, A.; Porcello, E. (2016) Learning React. Functional Web Development with React and Redux, first edition ed.; O'Reilly Media.
[9] Filipova, O. (2016) Learning Vue.js 2; Packt Publishing Ltd.,
[10] Eschweiler, S. Learn Redux - Introduction to State Management with React2017. Accessed 19.4.2018
[11] Hezbullah Shah, T.R.S. (2017) Node.js Challenges in Implementation. Global Journal of Computer Science and Technology: E Network, Web and Security 2017
[12] Mardan, A. (2014) Pro Express.js; Apress. https://doi.org/10.1007/978-1-4842-0037-7
[13] Brett, J. (2016) Getting Started with hapi.js; Packt Publishing Ltd. Livery Place.

6 Authors

Rene Berger is currently working as a senior developer at Parkside. He deals with software architecture, web development and machine learning. His focus is on developing web platforms, which are supported by artificial intelligence.

Markus Ebner is currently working as a Junior Researcher in the Department Educational Technology at Graz University of Technology. He deals with e-learning, mobile learning, technology enhanced learning and Open Educational Resources. His focus is on Learning Analytics at K-12 level. In addition, several publications in the area of Learning Analytics were published and workshops on the topic were held.

Martin Ebner is with the Department Educational Technology at Graz University of Technology, Graz, Austria. (E-mail: martin.ebner@tugraz.at). As head of the Department, he is responsible for all university wide e-learning activities. He holds an Assoc. Prof. on media informatics and works at the Institute of Interactive Systems and Data Science as senior researcher. For publications as well as further research activities, please visit: http://martinebner.at. Email id: martin.ebner@tugraz.at

Article submitted 2019-01-09. Resubmitted 2019-03-10. Final acceptance 2019-03-10. Final version published as submitted by the authors.