Integrating and Accessing University Information APIs Using Natural Language Processing Tools

Shantanu Hadap / Shubha Shubha

Attending or working at a university requires students and faculty to manage a large amount of information to be successful. Both school administration and academic challenges have requirements to be met. One solution to this problem is to provide a virtual assistant which gives the required information to students and faculty. In this work, a virtual assistant that can completely understand conversational English and provide any needed information has been created. Talking to a virtual assistant to get the information is more convenient than the conventional way of doing it. Getting information from the virtual assistant does not require typing or browsing or any type of human interventions, which makes it more time-efficient and accessible. This solution has been tried in a private environment with a small scope at the University of St. Thomas and the results have been encouraging. Sixty percent (60%) of the users believe the assistant is very useful and the remainder finds it moderately useful.

Integración y acceso a API de información universitaria utilizando herramientas de procesamiento de lenguaje natural

Asistir o trabajar en una universidad requiere que los estudiantes y la facultad administren una gran cantidad de información para tener éxito. Tanto la administración escolar como los desafíos académicos tienen requisitos que cumplir. Una solución a este problema es proporcionar un asistente virtual para estudiantes y profesores que brinde la información requerida. En este trabajo se crea un asistente virtual que puede comprender completamente el inglés conversacional y proporcionar la información necesaria. Hablar con un asistente virtual para obtener la información es más conveniente que la forma convencional de hacerlo. Obtener información del asistente virtual no requiere tipear o navegar ni ningún tipo de intervención humana, lo que lo hace más eficiente y accesible. Esta solución se ha probado en un entorno privado con un pequeño alcance en la Universidad de St. Thomas y los resultados han sido alentadores. El 60 % de los usuarios cree que el asistente es muy útil y el resto lo encuentra moderadamente útil.

Integrating and Accessing University Information APIs using Natural Language Processing tools

Shantanu Hadap, Shubha Shubha
hada6576@stthomas.edu, shub6690@stthomas.edu
University of St.Thomas

Abstract

Attending or working at a university requires students and faculty to manage a large amount of information to be successful. Both school administration and academic challenges have requirements to be met. One solution to this problem is to provide a virtual assistant which gives the required information to students and faculty. In this work, a virtual assistant that can completely understand conversational English and provide any needed information has been created. In this work, a virtual assistant is created that can completely understand conversational English and provide any needed information. Talking to a virtual assistant to get the information is more convenient than the conventional way of doing it. Getting information from the virtual assistant does not require typing or browsing or any type of human interventions which makes it more time-efficient and accessible. The implication of this work is that students, faculty and staff will have access to a virtual assistant to help out and thus minimize the stress of a school year. This solution has been tried in a private environment with a small scope at the University of St. Thomas and the results have been encouraging. Sixty percent (60%) of the users believe the assistant is very useful and the remainder finds it moderately useful.

Introduction

Is it possible to have a virtual assistant to students and faculty that will completely understand conversational English and can provide the relevant answer or information? 'Canvas Panda' is one such assistant which is developed specifically for the University of St. Thomas. This app runs on smart assistants like Google Assistant or on web and mobile chat-bot applications like Facebook messenger and Skype. The user needs to specify "Talk to Canvas Panda" for the Google Assistant to take them to the app. This assistant app then gets the input questions from the users. These questions are mapped with the related intent defined in the Google Dialogflow. The intents can be with parameters or without parameters. The intents are mapped to the related methods in the cloud functions and these methods make the REST API calls to different university APIs like library, student/faculty directory, computer labs, etc. It can also access secured API services such as Canvas APIs [1] and student account APIs. The received information is cleaned and passed back to the assistant. The assistant provides the required information in English sentences as output.

Materials and methods

Day to day university queries need a time-efficient solution with minimum user interactions and easy accessibility. To develop this solution, an NLP tool called Google Dialogflow [2] is used. An agent/app with an invocation name 'Canvas Panda' has been created in Google Dialogflow actions [3]. This app maps user conversational queries to specific intent using training phrases. Common queries based on university environment have been listed and intents have been created for each of these queries. Inside intents, training phrases have been defined. The training phrases are the different possible ways in which a query can be asked by the user. Dialogflow ML can map the query to intent even if it is not structured properly or is grammatically incorrect. Dialogflow also keeps on training and improving this mapping. The training phrases can also contain parameters/slots. The intents are linked to the Google Cloud functions [4] which contain methods for each intent. The cloud function is written using Node.js [3] and has been deployed on the Google Cloud platform. The cloud function calls university specific open APIs such as library service API, directory service API, etc. Integrating the third-party API can be achieved by implementing authentication methods like OAuth. The result obtained by API is cleaned and sent back to Google Assistant which reads or displays the response to the end user.

Figure 1. Architecture design of Canvas Panda. The APIs integrated are shown in green and the possible integrations are shown in yellow.

Results

The preliminary version of this app has been tested in a private environment and a survey has been done with a small group of five people. The survey result shows that 60\% of the users believe the assistant is very useful/accessible and the remainder finds it moderately useful/accessible. All the users find the app to be time-efficient. Users were asked to vote for their preferred choice of device for this application. The response was distributed as follows: 29% voted for smart-speaker, 29% for mobile, 18% for smartwatch, 12% for website and 12% for web messenger.

Figure 2. Survey results, text output using Google Assistant on mobile phone.

Conclusion

Our main aim was to develop a time-efficient and easily accessible solution for day-to-day university queries of students and faculties. The project was initiated with a narrow scope and limited API integrations. The response of the people participating in the survey reflects that the draft version of the app build using Google Dialogflow satisfied our focused aspects of efficiency and accessibility.

The scope of this project could be further increased by including more actions/queries such as student account details, library systems, campus news and announcements, lab/study room bookings and availability, etc. Integration of different third-party university applications like Canvas, Outlook, etc. could be a future addition to this application.

References

[1] Instructure Inc. Canvas lms – rest api and extensions documentation.

[2] Google Inc. Build an agent from scratch using best practices.

[3] Google Inc. Build fulfillment with the Actions on Google Node.js client library.

[4] Google Inc. Google Cloud Functions documentation.

Acknowledgements

We would like to thank Michael Dorin and Abe Kazemzadeh for their guidance; and Brett Coup, Eric Tornoe and Benjamin Durrant for providing technical resources required for the project.