TEDU Assistant

Final Report

CMPE 492 Senior Project II

Advisor Yücel Çimtay

Jury Aslı Gençtav
Emin Kuğu

Members Deniz Akşimşek
Enes Akmil
Gizem Kurtcuoğlu
Mehmet Batuhan Şenol

Introduction

It can be time consuming or complicated to access specific information from university websites. Our project, TEDU Assistant, [1] is a voice assistant/chatbot which will provide students, academic and administrative staff with natural-language access to all kinds of information related to the university. This provides a plain and convenient interface which adapts to user needs.

Design goals

Extensibility: It should take minimal effort to add features to the system. Clear extension points should be defined that allows extensions to a module to be decoupled. In particular for this system, it should be possible to add more language support and more kinds of questions that can be answered.

Maintainability: The system should not be difficult to maintain. When defects are found, fixing those defects should be possible with localized changes. Fixing defects and adding features should not introduce spurious defects.

Usability: The system should be easy to use. Users should be able to know the capabilities of the system and how to use these capabilities to fulfill their needs without needing instruction. They should not feel confused or frustrated while using the system. They should be able to give feedback about the system.

Performance: Firstly, the system should respond to all user interaction on a timely basis. Secondly, the system must be able to function with a minimal computational expense in terms of memory use, CPU time and power consumption. To achieve this, we plan to make extensive use of caching and memoization.

Availability: Ideally, the system should be ready to respond to users at all times to build user confidence. To achieve this, we need to ensure high uptime.

Security & Privacy: The system must not provide information to users who are not authorized to see it. It should also make sure that users' questions are not intercepted by any third parties. Our system will take a conservative approach to security by only providing information that is readily publicly available on university websites.

Final Architecture and Design

The system was implemented as a web application using the Flask web framework.

The basic use case of asking a question is performed as follows:

Speech-to-text conversion is performed on the client side of the application using the Web Speech API. The question text is sent to the server's /ask resource.

The server comprises three modules: Web, Response generation and Web scraping. The web module initially handles the request, passing it on to the response generation module.

The response generation module processes the text in two ways. The first is intent classification. This is performed using a machine learning pipeline consisting of a TF-IDF vectorizer, used for converting the text to feature vectors, and a decision tree classifier which determines the intent of the message out of a set of predefined intents ("room number", "exam date", etc.). This system is trained using manually produced and classified data. Based on the intent, the correct Responder subclass is instantiated.

The second aspect of NLP in response generation is named entity recognition (NER). This is performed via the spaCy library and a pre-existing model (xx_ent_wiki_sm). This process allows the parameters of the action (name of instructor, course code etc.) to be extracted.

The created responder instance is invoked with the intent and entities. Most responders then invoke the web scraping module to retrieve data from the university website, which they then format into a response. Responses as implemented in our system comprise a plain-text part containing short answers or explanation along with an HTML part for detailed data such as tables. The response object is formatted as HTML by the web module and returned to the client as a response to the original /ask request, at which point the client can present it to the user.

The application has a limited amount of client-side components. The main program merely passes data from the speech recognizer to the server via HTTP request and presents the server's response. As the server returns data in a format that is ready to present, client-side processing is not needed. The largest component of the client (in terms of the amount of code) is the speech recognizer widget.

Tools and technologies used

NLTK: A Python library for natural language processing, used here to tokenize text and remove stopwords.

spaCy: Another NLP library. Used in this project for named-entity extraction, as it has better support for the Turkish language in this aspect compared to NLTK.

SciKit-Learn: A library with implementations of common machine learning techniques, as well as abstractions to compose them. Used here for intent classification.

BeautifulSoup: A library for web scraping that parses HTML and makes it possible to extract data from it via CSS selectors.

Web Speech API: A Web Standard for browsers to provide speech recognition and synthesis capabilities.

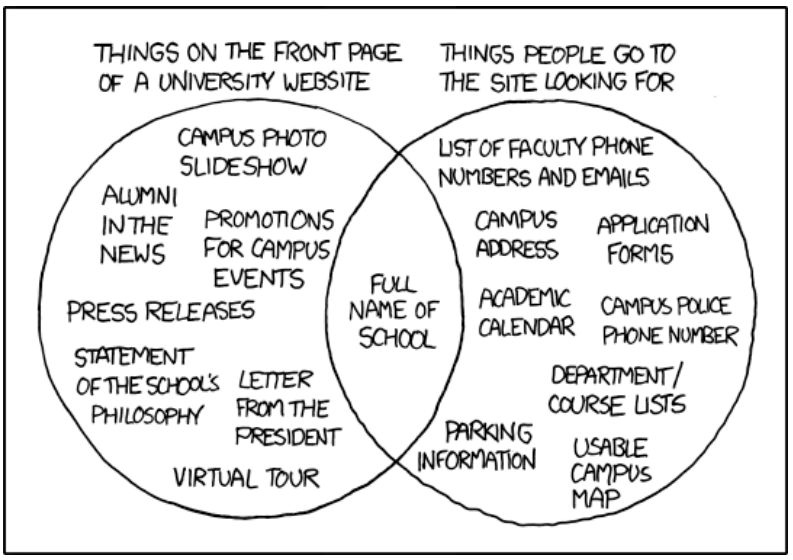
Flask: A web server framework, known for its simplicity. Used here instead of more "sophisticated" frameworks as the system has only a few HTTP actions.

Replit: An online, collaborative development environment. Used in some stages of the project when the team was unable to meet in person.

Contemporary issues and impact of engineering solutions

Conway's Law, formulated by Melvin Conway, is the idea that when an organization designs a system, the structure of the system will directly imitate the communication system of the organization.

Given universities' reputation for having complex bureaucratic structures, this does not bode well for the design of technical systems in universities. Indeed, university websites are an example of where Conway's law is in full force, with users unable to navigate the structure of the website/organization to access the information they need.



R. Munroe, "University Website," xkcd, https://xkcd.com/773/.

Natural-language systems, such as chatbots, have gained traction as a possible solution to this issue. In a chat interface, rather than the user navigating through the system, the system is responsible for interpreting the user's requests and presenting the information right away, walking the user through any intermediate steps that might be required.

Social issues

The TEDU Assistant system runs up against many of the current barriers of chat interfaces. Firstly, speech recognition technology performs poorly with multilingual input. Like many universities in Turkey and around the world, TEDU is multilingual through and through, with courses being conducted in many languages as well as staff and students from many parts of the world. Given the limitations of available speech recognition technology, we have made textual input available to the system.

One consequence of Conway's Law that is often neglected is that the inadequate information architecture it often produces leads to inaccessible systems. The WebAIM Million [2] project found that the mean website has around 50 accessibility failures. University websites are often inaccessible to visually impaired people, as well as people with cognitive disabilities due to their unnecessary complexity. TEDU Assistant provides a means of access by supporting multiple input methods and simplifying information access. In addition, the website was constructed with respect to the W3C's Web Content Accessibility Guidelines [3].

Economic issues

The TEDU Assistant system has the potential to reduce inefficiencies in university operations.

Test results

Results of running the planned tests [4] are as follows.

TA-TEST-001 Voice Recognition Accuracy

Partially passed.

The system can recognize voices with accuracy on the level of state-of-the-art recognition systems, but suffers with mixed-language speech.

TA-TEST-002 Question Parsing and Response Generation

Passed.

The response generation subsystem can successfully identify the best responder for a question by classifying questions by intent. It can also extract the parameters to pass to that responder using named entity recognition.

TA-TEST-003 Web Scraping Functionality

Passed.

The web scraping subsystem can successfully fetch the content of any publicly available web page.

TA-TEST-004 Internship responder

Passed.

When the system detects that a question is related to internships, it presents the list of internship coordinators, lets the user choose their department and internship, and provides contact information for the coordinator.

TA-TEST-005 Staff info responder

Passed.

The system can present an overview of information on a specific member of academic staff (e.g. Dr. Yücel Çimtay) as well as extracting specific information such as office room number.

TA-TEST-005 Final exams responder

Partially passed.

Because the final exam dates are not available in a structured machine-readable format, the techniques and technologies used are insufficient to extract the dates of specific exams. However, the system can direct the user to the calendar of final exams.

TA-TEST-006 Academic calendar responder

Passed.

The system can direct users to the academic calendar.

TA-TEST-007 Department info responder

Passed.

The system can list staff and personnel of a department.

TA-TEST-008 Opened courses responder

Passed.

The system can present the list of opened courses.

TA-TEST-009 User acceptance testing, TA-TEST-010 Open beta

An user acceptance test or open beta has yet to be conducted as of the submission of this report.

References

[1] D. Akşimşek, E. Akmil, G. Kurtcuoğlu, M.B. Şenol, “Final Project Proposal—TEDU Assistant,” 2022. https://tedu-cmpe491-group10.netlify.app/reports/proposal.pdf.

[2] WebAIM, “The WebAIM Million,” 2023. https://webaim.org/projects/million/#errors.

[3] W3C, “Web Content Accessibility Guidelines (WCAG) 2.1,” 2018. https://www.w3.org/TR/WCAG21/.

[4] D. Akşimşek, E. Akmil, G. Kurtcuoğlu, M.B. Şenol, “TEDU Assistant—Test Plan Report,” 2023. https://tedu-cmpe491-group10.netlify.app/reports/testplan.pdf.