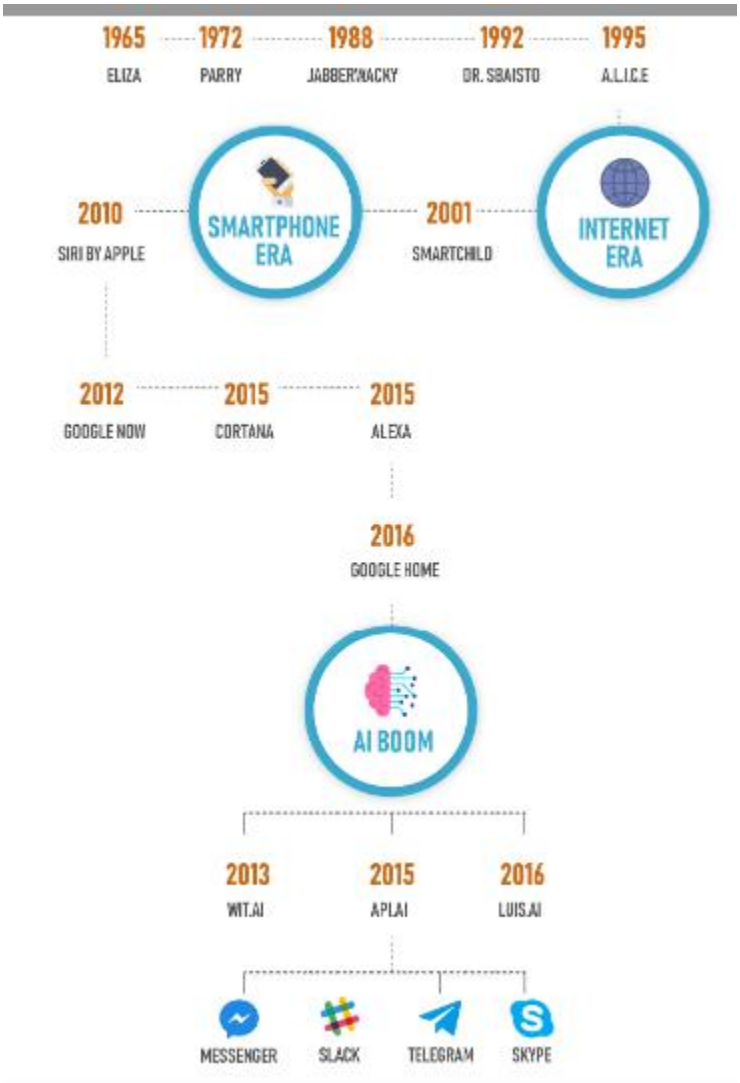
CHATBOT DEVELOPMENT USING PYTHON

Introduction

Chatbot definition and history:

A chatbot is a piece of technology that allows a computer program to communicate with people just like conversing through text messaging using a natural language, say English, to accomplish specific tasks. A chatbot is also known as an artificial conversational entity (ACE), chat robot, talk bot, chatterbot or chatterbox.



The above picture shows a brief history of chatbots.

Project scope:

Chatbot is an AI chatbot that receives questions from users, tries to understand the question, and provides appropriate answers. It does this by converting an English sentence into a machine-friendly query, then going through relevant data to find the necessary information, and finally returning the answer in a natural language sentence.

The main objective is creating a Web API, and sample web and text messaging interfaces that demonstrate the use of the API.

Description

Project perspective:

Most of the search engines today, like Google, use a system (The Pagerank Algorithm) to rank different web pages. When a user enters a query, the query is interpreted as keywords and the system returns a list of highest ranked web pages which may have the answer to the query. This Chatbot, however, will try to understand the query and provide a definitive answer.

There will be four main units to the system working together to understand the question and return an appropriate answer:

- Generic question construction - capable of taking a natural language question and making it more generic.
- Generic answer construction - capable of taking a generic question template and providing a generic answer template
- Generic answer population - capable of taking a generic answer template and populating it with information from the database to form an answer
- Information extraction - capable of finding information through structured or unstructured websites, and storing that information in a database.

Product feature:

The major features of the products are:

- **Web API:** An API call will include a question in the form of a query string URL parameter and the service will reply in JSON.
- **Natural Language Processing:** The system will take in questions written in standard English

- **Natural Language Responses:** The answer to the question will be written in standard and understandable English
- **Information Extraction:** There will be a database containing all the information needed, populated using information extraction techniques.

Constraints:

- Creating a chatbot able to answer every single question about Drexel is not possible to implement with current technology and within the duration of the project, so the system will be able to answer questions about limited topics.
- The system will understand only English language.

FUNCTIONAL REQUIREMENTS

Client system:

- The client will send a GET request to the Web API with the question as a URL parameter.
- Client will specify the header Content-Type: application/JSON in their requests as convention.

- A valid API query is a single URL parameter containing one sentence that is a question in standard English.
- The server will reply with either data or an error, the client will be able to parse the JSON and determine if there was an error.

Server System:

- The server will send all API data in JSON response documents with the header Content-Type: application/JSON.
- The server will respond with a 200 OK status code if a request has the header Content-Type: application/json and is a valid API query.
- The server will respond with a 400 Bad Request status code if a request does not specify the header Content-Type: application/json OR is a malformed API query

Response Structures:

- API responses are defined in JSON.
- A JSON object will be the root of every API response.
- Data: the document's "primary data," in this case, the response to the client's query

- Errors: an array of error objects stating what went wrong with the client's request, should any issues arise.
- The top-level members specified in R4.1.3.3 will not coexist in the same JSON document. If data is present, errors will be absent and vice versa.

Generic question construction:

- This unit will receive a text string from the URL parameter.
- This unit will identify important words in the sentence and replace them with generic representations preceded by an escape character.
- This unit will output the sentence as a string.
- This unit will output a map of generic representations to the words they replaced.
- This unit will have a list of generic words related specifically to potential queries.
- If there was an error here, then the unit failed to create a generic answer given a generic sentence. In this case, simply fallback to the error handling described in

Generic answer construction:

- This unit will receive as input a mapping from the Generic Question Construction unit.
- This unit will receive as input a generic answer from the Generic Answer Construction unit.
- This unit will query the database for data about the elements in the mapping.
- This unit will replace the representations in the generic answer with data.
- This unit will output the answer to the original question.
- If there was an error here, then the unit failed to create a generic answer given a generic sentence.

Generic answer population:

- This unit will query the database for data about the elements in the mapping.
- This unit will replace the representations in the generic answer with data.
- This unit will output the answer to the original question.
- If querying the database did not provide an answer, the system will say that it does not have an answer and provide appropriate website link where the user could find the answer.

- If the system could not find appropriate website associated with the question, the system will return a generic error message such as “Sorry, I couldn’t find an answer to that.”

Database:

- A MySQL database will be used to store all information required to answer questions.
- The database is populated by the information extraction unit before the rest of the system is available, so that all information is readily accessible.
- The database will use the generic representations from as table names or column headers for easier retrieval of data.
- The database will be updated periodically and the API will be unavailable during the update.

Structured input:

- These are highly organized data sources, such that including the data into our database is simple.
- Structural data sources will have their data stored in our database.

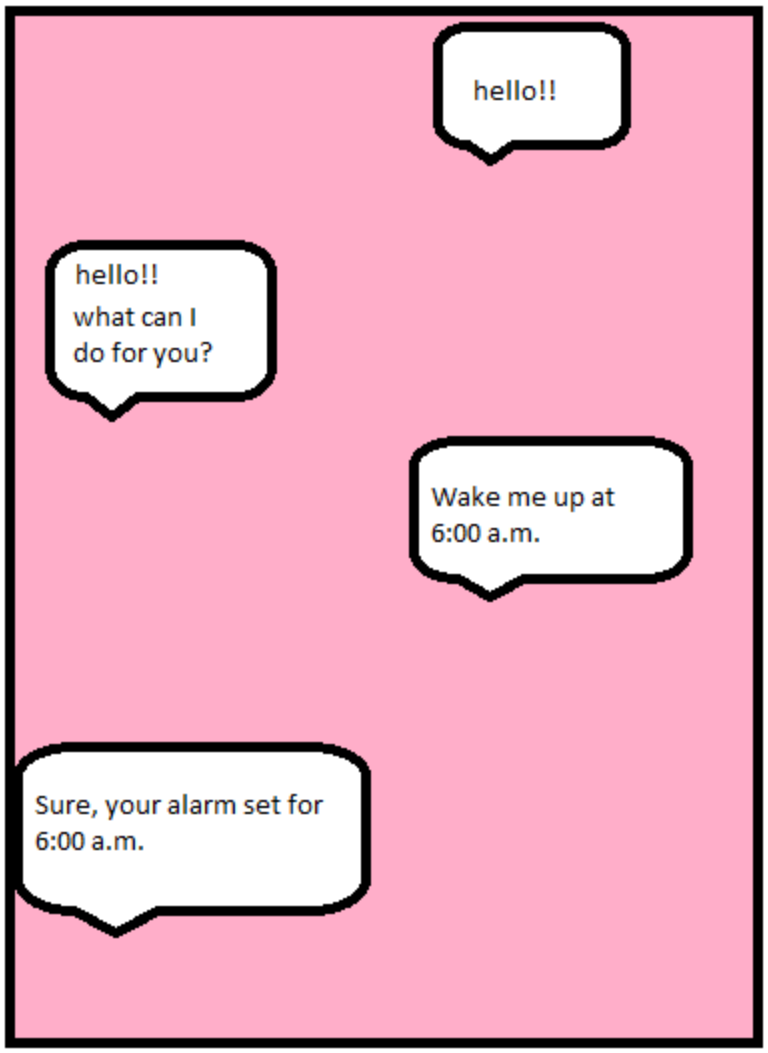
Semi structured and unstructured input:

- Semi-structured data sources are data sources with some organization, but the structure is not rigid enough to assure easy extraction of data.
- Unstructured data is data with no organization, so extracting information is very hard to do programmatically.
- Extraction of information from semi-structured and unstructured data sources can be handled in 3 possible ways:
    1. For data with enough structuring, web scraping will be used to programmatically extract all the data needed and store it in our database.
    2. For data with enough structuring, web scraping will be used to programmatically extract all the data needed and store it in our database.
    3. Data that is especially hard to extract, for whatever reason, will be manually extracted and added to the database by a developer.

User Interface:

- The GUI will have a textbox that will accept inputs from a keyboard.

- Text box will originally contain a suggestive text question, to guide the user to the format of an appropriate question.
- The GUI will have a “Send” button which sends text from the textbox to the API when clicked.
- The GUI will have a chat window displaying questions sent to the system and responses from the API.
- The chat window will contain all questions and answers from the current session, with a scroll bar if all messages can’t fit on the screen.
- If there is a network issue, the chat window will display an error message.



A chatbot conversation example

Non Functional Requirements

API:

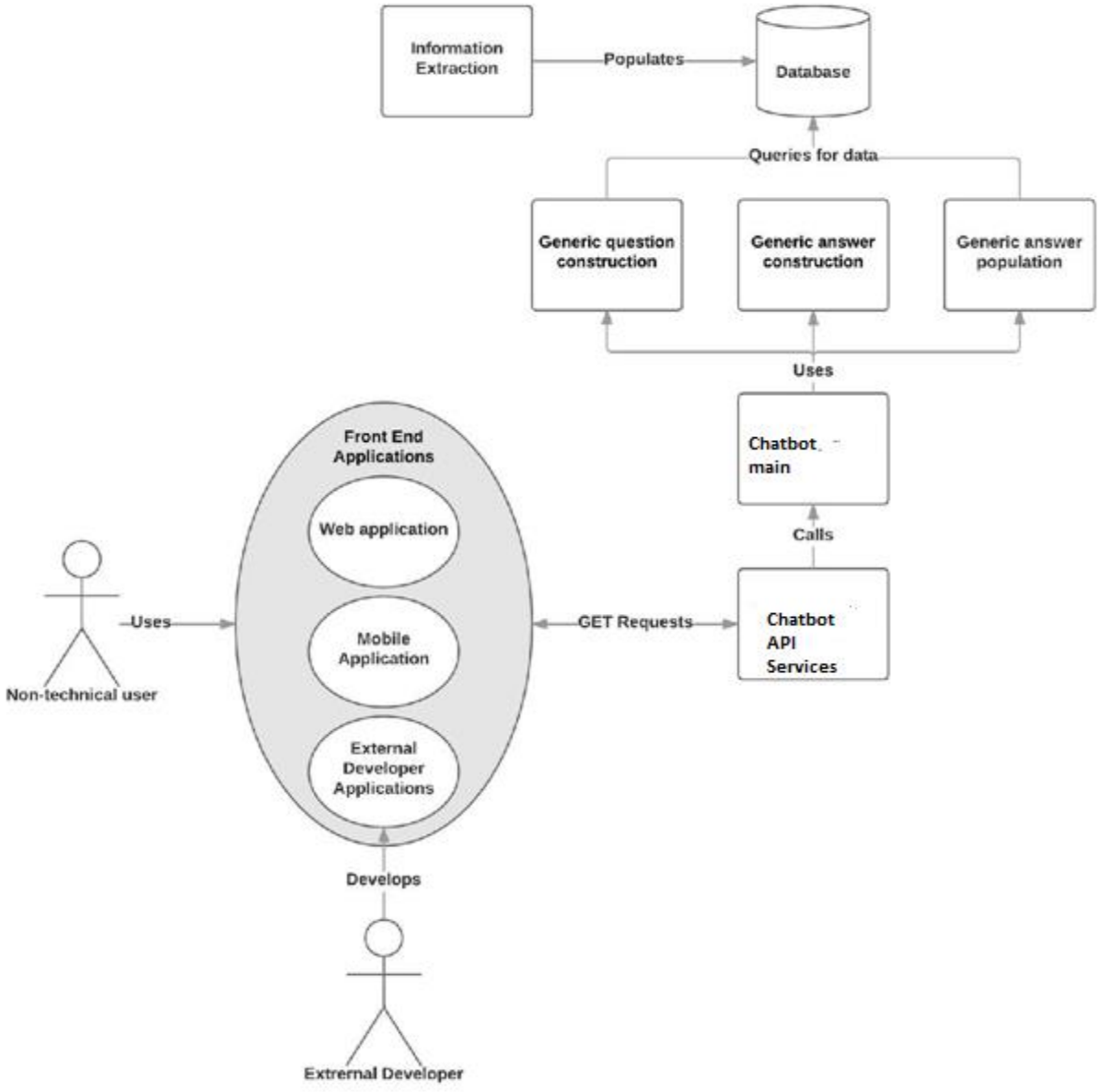
- The system will be designed in such a way that the algorithms for the four main units will be able to be easily swapped out.

- The overall accuracy of the Web API's response will be measured using a developer-made testing set.
- The overall accuracy is calculated by dividing total number of correct answers by the number of questions asked.
- The accuracy of the Generic Question Constructionpart will be close to 80%.
- The accuracy of the Generic Answer Constructionunit will be close to 70%.
- The accuracy of the Generic Answer Populationunit will be close to 70%.
- The average time for the server to respond, over the question testing set, will be less than or equal to 2 seconds.
- The connection between the Web API and the programs will use HTTPS, for security.

Web Interface:

A new user will make less than 3 mistakes in 5 minutes after 5 minutes of use.

Use Case Diagram:



Activity diagram