X-o-Bot: A Generic Dialog Framework for ConversationalAgents - Chatbot for Comp9444

Group Name - SeungBot

Members:

Subham Anand
Z5151878
s.anand@student.unsw.edu.au
Role : Scrum Master and Developer

Vivek Arunagiri
Z5141013
v.arunagiri@student.unsw.edu.au
Role : Developer

Jiaying Guo
Z5129971
jiaying.guo@student.unsw.edu.au
Role : Developer

SheHoi Yeung
Z5130257
z5130257@student.unsw.edu.au
Role : Developer

Table of Content

Introduction

*A **chatbot** (also known as a smartbot, talkbot, chatterbot, Bot, IM bot, interactive agent, conversational interface, Conversational AI, or artificial conversational entity) is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods.*

Chatbots were first introduced in 1966 - Eliza which used to the mimicked human conversation by matching user prompts to scripted responses - it was able, at least for a time, to pass the Turing artificial intelligence test. From there it continues to evolve with time and we saw some chatbots like PARRY, ALICE, SIRI, ALEXA, GOOGLE Assistant and CORTANA.

Chatbots are introduced in many fields nowadays and one of them is education which helps students and other professionals as well. Bostify is an education chatbot which gives a specific topic to students in the form of text, images, videos. After learning the topic students take quizzes, assignments and submit the results to their teachers. This way teachers can easily track student's performances.

In universities, help desks are set up to solve or answer any small queries of students which can be answered using chatbots. So we decided that we will be working on a student - tutor support system which will extract all the content from the specific course and answer the key questions related to the course.

Existing System and Problems in it

In the existing system, the available chatbots answer more generalized questions and cannot answer more specific questions on a subject which makes little difference to students who will have to go through the complete lectures or search the web thoroughly to learn the answers. Also, the response time of chatbots is quite high and not instant. Integration of the chatbot with the different application is not yet configured.

How our system will solve the problems in existing system

Our chatbot application will be designed to answer all questions on a given subject by getting data from the forums and StackOverflow to train our chatbot. This will make our chatbot answer students questions on a subject in depth, with more precision and lower response time. Our chatbot is designed as an API that can be integrated into different applications like the facebook messenger.

AIM

The answers provided by our chatbot will be precise and reliable. Our chatbot will answer in depth questions regarding the subject. The response time will be low and the user does not have to wait to get the reply. The chatbot can be plugged into multiple application and used as enquiry assistant.

1. Content related information.
    a. The chatbot will provide answers related to the content of the course.
    b. Learning outcome from the course.
    c. Overview of the course.

2. Lecture related information.
    a. Class time table and professor details.
    b. Lecture and lab location details.

3. Assignment and exam related information.
    a. Questions related to assignments, quizzes and projects.
    b. Questions related to the mid-term exam and the final exam.

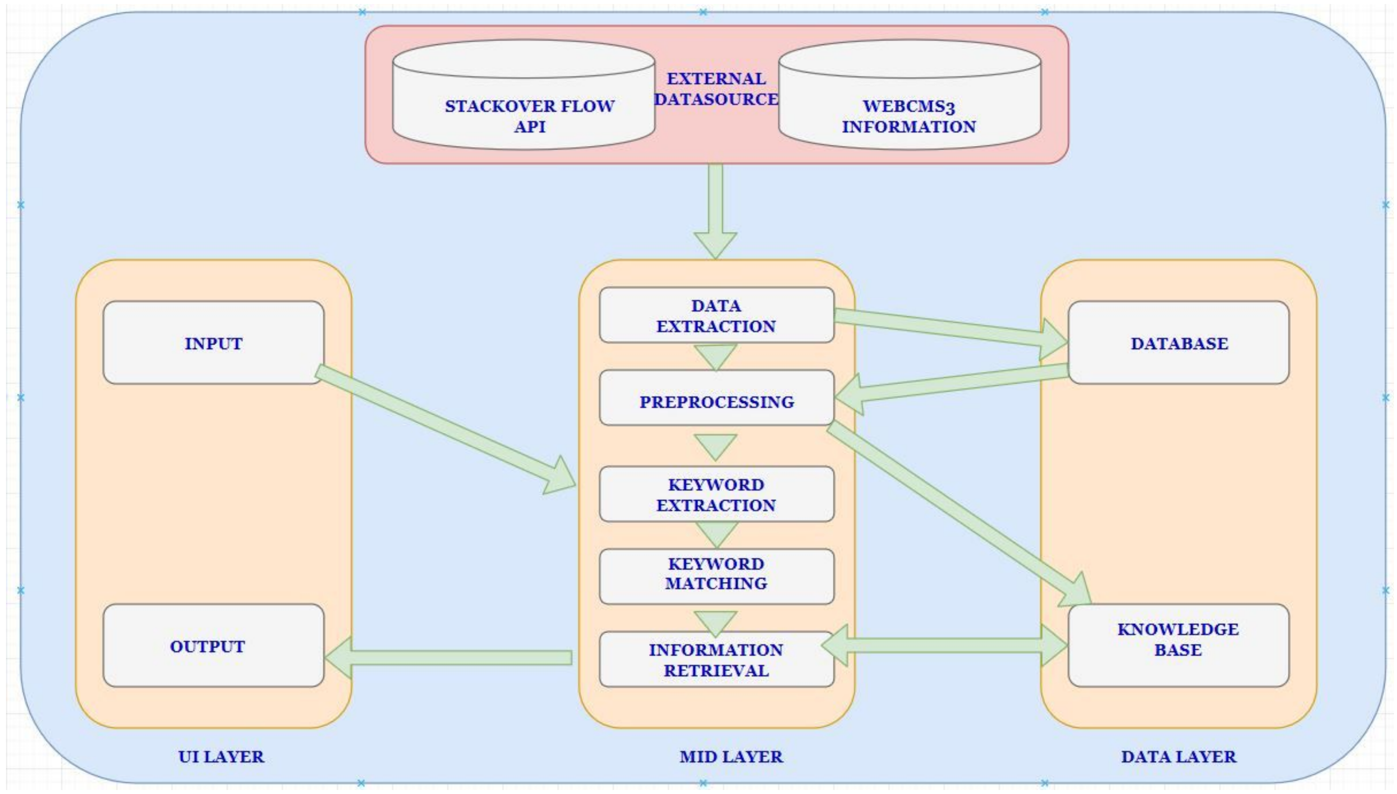
SCOPE

We will be building a chatbot that simulates the interaction between a student and a tutor, where the student has questions about the details and structure of a course offered by UNSW and the chatbot can answer the question or link the student to the appropriate page.
We will be using COMP9444 - Neural Networks and Deep Learning to show how our system would work, but this can be easily modified for any other course. We intend to extract all the relevant information from the **course website** and also all the questions and answers in the **forums**, we will also extract data from **StackOverflow** and store them in a cloud-based database. We will then convert this data into a knowledge base we can use to answer questions answered by the student. Based on the question asked by the student we will use NLP and ML algorithms to provide the most accurate answer. We will be also implementing a feedback system where the student can provide feedback saying whether the answer was helpful and this will be used to tune the algorithm.
Our chatbot returns more accurate and precise answers and the feedback system will keep improving the system as it is being used.

BACKGROUND

The domain of our chatbot is the student-tutor support system, which will focus on a specific subject in our case is Neural Networks and answer in depth questions on the subject. The users of our service will be students taking that specific subject. Our system will have a simple user interface, which can also be integrated into other applications like Facebook messenger, where students can ask questions and the system will process the question and provide a precise, accurate answer in low response time.

System Architecture



Our system architecture consists of three layers UI layer, Middle layer and Data layer.

UI layer interacts with the user by getting the input question passing it to the middle layer and getting the response from it and showing it to the user. The visual look of the UI layer is simple with just three things input, output box and a button. Whereas we are integrating our UI to facebook messenger so that anyone can access the chatbot if the person is connected to the chatbot facebook page.

Middle layer performs all the operations to provide the answers with high accuracy and precision. It starts with data extraction from different sources such as previous year forum questions of the course and collection of question answers from StackOverflow and different other sources to train our model with more questions. To get more accuracy, preprocess the data which includes data cleaning and removing the noises from the data. Then we extract the keywords from the cleaned data which is stored in the knowledge base. When a user asks a question to the system we need to extract keywords from the question and send it to the data layer to get the response. Keyword matching and Information retrieval are the processes to get back answer from our corpus in the database to provide it to the user. Keyword matching compares the questions in the database and knowledge with the user input using cosine similarity and processes the answer which has the highest similarity.

Data layer will consist of two types of data corpus-based and knowledge-based. All the data will be preprocessed and clean already and stored in the database. The database to be used is AWS RDS Microsoft SQL Server.

Data Extraction

We have implemented a web scraper and provided the web url for the course website. The web scraper extracts the questions and answers from the forums that are available to the public using BeautifulSoup. It also extracts the basic information about the course such as the timetable, the lecturer, the course outline etc. The challenges we faced in scrapping the data from the course website was that many of the forums were locked behind a signup page for students enrolled in the course. This restricted the amount of data we could extract from the course website.
To circumvent this, we decided to look at public forums, which dealt with the subjects our chatbot was dealing with. Using the stackoverflow api, we extracted all the questions and answers relating to all the keywords we could think of relating to our topic “Neural Networks and Deep Learning”. The stackoverflow api returns a json formatted text which contains all questions and answers for a given keyword. We then process the json text to extract the question, answer pairs and stored in the database. This way we increase our available data corpus to train our chatbot.

Example: Following is the example of returned json file from stackoverflow api.

```
"tags": [
"python",
"neural-network",
"backpropagation",
"feed-forward"
],
"owner": {
"reputation": 21,
"user_id": 4256575,
"user_type": "registered",
"profile_image": "https://i.stack.imgur.com/zVDT9.jpg?s=128&g=1",
"display_name": "Nek",
"link": "https://stackoverflow.com/users/4256575/nek"
},
"is_answered": false,
"view_count": 157,
"answer_count": 0,
"score": 0,
"last_activity_date": 1528671055,
"creation_date": 1528670541,
"last_edit_date": 1528671055,
"question_id": 50788677,
"link": "https://stackoverflow.com/questions/50788677/python-xor-back-propagation-neural-network-converges-to-0-5",
"title": "python - XOR Back Propagation Neural Network converges to 0.5"
},
```

The json file consists all the questions related to the keyword searched. This is file is again parsed to open each and every link provided for the particular question. After opening each link it scan the page for the question description and select the most relevant answer. The most relevant answer is selected by observing the post with most upvotes and if any answer is marked as right one. After extracting the questions and answer it is saved into a csv file to provide it as a input to the next step. Below is the format of details stored in csv file.

Question	**Answer**
What is neural network.	Artificial neural networks (ANN) or connectionist...
What is deep learning.	Deep learning is part of a broader family of machine learning...
What is reinforcement learning.	Reinforcement learning is an area of machine learning...

Keyword Extraction

Once our database is filled with data, we then go through the database and extract the keywords from the Questions using nltk (Natural Language ToolKit) and rake (Rapid Automatic Keyword

Extraction) libraries. The extracted keywords are pre-processed to make sure they are cleaned using the re (regular expression) library. These are then ranked in order of importance and are stored in our knowledge base along with the answers.

Example:

Question	**Keywords**
How to train an artificial neural network to play Diablo 2 using visual input?	['play diablo 2 using visual input', 'artificial neural network', 'train']
Training a Neural Network with Reinforcement learning	['reinforcement learning', 'neural network', 'training']
What is the difference between value iteration and policy iteration?	['value iteration', 'policy iteration', 'difference']
What is the difference between Q-learning and SARSA?	['sarsa', 'q', 'learning', 'difference']
Support Vector Machines -- Better than Artificial Neural Networks in which learning situations?	['support vector machines -- better', 'artificial neural networks', 'learning situations']
Good implementations of reinforcement learning?	['reinforcement learning', 'good implementations']
When to use a certain Reinforcement Learning algorithm?	['certain reinforcement learning algorithm', 'use']
What is the way to understand Proximal Policy Optimization Algorithm in RL?	['understand proximal policy optimization algorithm', 'way', 'rl']
How can I apply reinforcement learning to continuous action spaces?	['continuous action spaces', 'apply reinforcement learning']
What is the difference between reinforcement learning and deep RL?	['reinforcement learning', 'deep rl', 'difference']

Keyword Matching

Once the keywords are extracted from the question asked by the user. This set of keywords is compared to each and every set of keywords in our knowledge base to find the most similar one using cosine similarity. The answer corresponding to the most similar keyword match will be returned to the user.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

where A_i and B_i are components of A and B respectively

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

Example:

User Input - What is neural network

The cosine similarity will find the most relevant answer in the database. It depends on the score of similarity and gets the answer with the highest score.

Cosine Similarity score from database - 0.75

Database

We used Amazon’s AWS RDS Microsoft SQL Server. We created a database instance in AWS and connected it to Microsoft SQL Server Management Studio. Our data corpus and knowledge base are stored in a table in this database. The database is secured with the master username and password and it uses sql server authentication.

Web Server

To connect the user interface and database we have used flask. We have divided our user interaction to two parts. One will connect the database to the web interface and the other will connect with the facebook messenger. We gave the connection string of our database which consists of db instance, master username and password. It will connect the server to the database and ready to retrieve answers. We have used facebook developer to create a facebook app which will be attached to a facebook page. This page will be used to interact with the user or admin of the page. Facebook developer uses messenger and webhooks to attach the server to our flask. We use ngrok which is a multiplatform tunnelling, reverse proxy software that establishes secure tunnels from a public endpoint such as internet to a locally running network service while capturing all traffic for detailed inspection and replay. The flask server contains the methods to get message from messenger and send back the reply.

Voice Command

We are using google's speech to text recognition. Whenever a user holds the record button in the messenger application to send a voice command it will be send to the flask server. Server receives the message and extract the link to the .mp4 file. Our server opens the link and download the .mp4. It converts the .mp4 file to .wav file which is the better input to the speech to text api. It will convert the speech to text and process the answer similar to the text input.

Web UI - React + Redux

For the web UI, we used React and Redux. React is really helpful for page design, it treats each page element in the web app as a component, its one-way data flow mode makes maintenance and management of UI component state easier and clearer.

When we developed with React, each page element is a component, the implementation of the page is equivalent to the assembly mode, especially for the business with high similarity for most of the pages, it will be particularly efficient and fast. For the add, modified or delete operation, it is very convenient because each component is independent, we do not have to worry about affecting other components, especially after we delete a component unless some components update state based on this component's data.

During the development, we used Redux as our data controller, the one-way data flow helps us troubleshoot problems easier. The re-render of a component is based on data changes, that means changing the data at the reducer, which in turn triggers the update of the component.

Imagine that the store is a library, the data is a book, and the reducer is a borrowed book record. Every time we update the data, it is equivalent to borrowing books or returning books. We need to update the borrowing records. When there are new users who want to perform certain actions, such as borrowing books, we need to check the borrowing record, if the book is in the library, we would allow the borrowing action, otherwise, we would reject the request.

Based on this logic, any problem occurs, as long as the data changes at each point are monitored via Redux DevTools, the problem can be easily identified.

Finally, as an app, it has a large store which contains multiple stores. Each store belongs to exactly one component. All actions for this component should be saved at its store and its data should be kept at reducer. For easier debugging, we have a constant file under the store directory which used for unifying the names of actions.

Traditionally, when a single file contains a lot of code and the code becomes messy and unreadable. However, using Redux to separate the logic from the presentation, so that the code is clearer and easier to maintain. The code is divided into pieces and each piece is very detailed. To change the logic, instead of focusing the React JSX, just pay attention to the actions in store. Every time we modify the data, we will have a specific action to do it. If there is a functional modification, it will be solved very easily. Most of the time, we do not need to change other actions of the same component.

User documentation

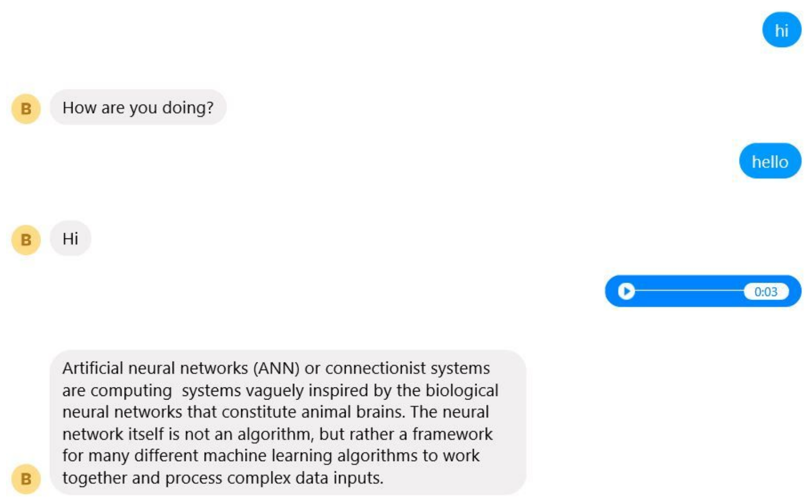
1) Chatbot

Set up:

1) Clone from the master branch and extract all the folders.
2) Run the flask.py file.
3) Run ngrok and provide the port name.
4) Feed the ip address and security key to the facebook developer webhooks.
5) Open facebook and start chatting as an admin to the bot.

Dependencies:

1) Install all the libraries mentioned in the readme file.
2) Set up a facebook account, page and app from the facebook developer.


hi

B How are you doing?

hello

B Hi

B Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs.

2) Web UI

Set up:

1) Clone from the master branch, then extract all
2) Change the path to the UI folder
3) Under the UI folder, open the terminal, and 'npm install' all dependencies
4) Run 'npm start', open http://localhost:3000 to view it in the browser.

Dependencies:

1) Axios : 0.18.0
2) Immutable : 4.0.0-rc.12
3) React : 16.8.1
4) React-dom : 16.8.1
5) React-loadable : 5.5.0
6) React-redux : 6.0.0
7) React-router-dom : 4.3.1
8) React-scripts : 2.1.3
9) React-transition-group : 2.5.3
10) Redux : 4.0.1
11) Redux-immutable : 4.0.0
12) Redux-thunk : 2.3.0
13) Styled-components : 4.1.3

Conclusion

There were some hurdles on the way to design our chatbot. The first one was how to get as much data so that we can design our model more accurately. We tried to get data from different sources which include stack overflow, webcms for previous years questions and details about the course. The next problem was how to extract questions and answers from stack overflow as it has quite some amount of questions and its manually impossible to gather all the data. So we designed a web crawling technique which will feed the keywords related to neural network to the stack overflow api and get the json file of the data and opens each and every link to extract the question and the most relevant answer. The next big problem was to host our chatbot with facebook messenger. We use flask and facebook developer to create app and page on facebook and to connect it to our server on flask.

There are some improvement which can be done in coming time. It is always better to have more amount of data to train the model and match the response for the user accurately. However cosine similarity gives the matching score almost accurately but it somehow do not care about the semantic of the input. However there are few similarity technique which can be used to take care about the semantic of the input as well. One more major improvement can be the feedback system which can train the model by taking in the inputs of the user over time and try to respond more fast accurately when it is asked again in future.

References:

https://blog.hartleybrody.com/fb-messenger-bot/

MOOC-O-Bot: Using Cognitive Technologies to Extend Knowledge Support in MOOCs

https://en.wikipedia.org/wiki/Chatbot

https://en.wikipedia.org/wiki/Cosine_similarity

https://pcc.cs.byu.edu/2018/03/26/a-brief-history-of-chatbots/

http://flask.pocoo.org/docs/1.0/

https://ngrok.com/docs

https://api.stackexchange.com/docs/questions

https://realpython.com/python-speech-recognition/

https://reactjs.org/

https://github.com/reduxjs/redux-thunk

https://immutable-js.github.io/immutable-js/docs/#/