2020

# Artificial Intelligence

STAGE 1: CHATBOT WITH RELATED AND SIMILARITY-BASED COMPONENT

RISHI SINGH 2018 (N0834113)

## Table of Contents

# Technology Stocks Chatbot

## System explanation and goals

For this project, I have created a technology stocks chatbot that should answer specific questions about technology stocks and give relevant information about companies, including the summary of their profile and their current share value. The chatbot's main aim is to provide users with pertinent information to make the correct decision to buy shares in the stock market. Currently, the chatbot has been programmed with information about different technology companies' profiles and their live share price. Additional information a variety of extra information has been stored in a CSV file holding questions and answers for this submission's similarity-based component.

The system's primary goal is to allow the user to enter specific questions about stocks. The chatbot should be able to answer using a related based or similar based component. A related based component involves using pre-defined rules that lead to a specific answer that has been programmed using AIML (Artificial Intelligence Markup Language).

To simplify the fetching of data in this chatbot, I have web scrapped yahoo finance to get accurate data about the profile of a company and the live prices of stocks. Overall the main goal is to inform users of live share prices and their faces.

## System requirements

Must

- The system must allow the user to enter a message.
- The system must return a message to the user.
- The system must have a specific domain/purpose.
- The system must be able to visualise a chat.
- The system must be able to understand the user's input and answer accordingly.
- The system must be able to state any confusion in sentences clearly.
- The system must not crash.
- The system must be able to hold user information, such as name.
- The system must provide information about stock prices and profiles.

Should

- The system should have a user-friendly user interface.
- The system should be able to reply with non-textual data where it is suitable.
- The user must be able to enter several different stock names and get profile information on the stock.

Could

- The system could use text to speech.
- The system could have more variable features and a broader scope regarding the domain.

## Explaining the employed AI techniques

In this chatbot, several AI techniques will be implemented to provide an appropriate experience for the user, using rule-based and similarity-based components to create a conversing chatbot.

### Rule-based component

In the rule-based component of the chatbot, I will be using AIML (Artificial Intelligence Mark-up Language). AIML is used to create a human interface that allows the implementation of a program much simpler and more accessible using XML-based mark-up language. Below I have highlighted an example of how small components of which an AIML file consists of. 'AIML was developed by the Alicebot' (Unkown, 2020), allowing the creation and customisation of chatbot applications founded on A.L.I.C.E (Artificial Intelligent Linguistic Internet Computer Entity.'

```
<category>
        <pattern>WHAT CAN WE DISCUSS ABOUT </pattern>
               <template>
                      We can discuss about technology stocks and their prices
               </template>
</category>
```

Above are basic AIML tags, which can be used to create the rule-based component of this chatbot. The **category tag** signifies the 'unit knowledge in Alicebot's' knowledge, followed by the **pattern tag,** which shows the pattern resembling the user's input. The **template tag is** illustrating the response of the user's input (Unkown, 2020).

*Table 1: Tags and explanations*

| Tags | Explanation |
|---|---|
| **<condition>** | Aids ALICE to respond to matching input. |
| **<think>** | Used to store a variable from the user's input. |
| **<get>** | Gets the value stored in an AIML variable. |
| **<li>** | Lists the number of output choices there can be to a specific pattern. |
| **<random>** | Random responses are generated. |
| **<set>** | Sets values of AIML variables. |

| | |
|---|---|
| **<srai>** | Calls and matches other categories. |
| **<star>** | Matches wild card characters in a pattern. |
| **<topic>** | Stores context so later, the conversation can be upheld and based on that context. |
| **<that>** | Used to respond based on context. |

Similarity-based component
For the similarity-based component, there are three main models I will be using to find answers from a corpus that are similar to the questions and answered pre-defined. A CSV will hold predefined questions & answers which undergo the three models.

Bag of Words
The Bag of Words model is used by tokenising a piece of text and incrementing the number of times a word appears in that specific sentence or, more specifically, in a piece of text. NLP text is converted into numbers through the process of vectorisation using libraries from sklearn.  The reference sentence in this example could 'It was spectacular to see such an event.' As shown below, the bag of model words keeps the count of the frequency of words being repeated. Using stop words in the bag of words model is also essential to filter out words that occur in most sentences or a piece of text.

| It | Was | Amazing | To | See | Everyone | Yesterday |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |

TF-IDF
The TF-IDF model stands for the 'Term Frequency Inverse Document Frequency' (MonkeyLearn Blog, 2019) and derives the importance of a word to a sentence or a piece of text. By signifying the importance of a word in a specific sentence or extract of text allows the model to understand the context of the text, which is derived from the proportional increase of the number of times a word is repeated. The mathematical calculations calculate this stated below.

$$tf = \frac{\textit{number of occurences of a word}}{\textit{total word length of sample}}$$

Inverse Document Frequency:

$$idf = \log \frac{\textit{number of samples}}{\textit{number of samples that contain the word}}$$

$$tfidf = tf \cdot idf$$

*Figure 1:TF-IDF Calculation*

Cosine Similarity

The cosine similarity model uses two TF-IDF vectors to calculate the similarity between both vectors utilizing a formula to calculate the angle between the two vectors. For this submission, a CSV file will contain several predefined questions and answers; a user's input into the chatbot will be vectorised and compared to the CSV file text to calculate the cosine similarity value.

$$similarity(p, q) = \cos\theta = \frac{\mathbf{p}\cdot\mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|}$$

*Figure 2: Cosine Similarity calculation*

However, there are problems with cosine similarity which have been highlighted by Han & Li as they found the cosine similarity model is 'overly biased by features of higher values' disregarding the number of features shared by two vectors (Li & Han, 2013). Thus, the efficiency and accuracy of the model can be questioned.

## Explanation of the program.

In this section, I will be highlighting several parts of the program, which I believe are essential parts of the program. I have created a flowchart to show how the system should work, which has helped me implement the chatbot.

AIML

I have used various AIML tags, which are used to provide a base for communication between the user and the chatbot. As shown in the previous section, an XML file has been created, which is full of AIML tags; a specific part of the AIML tags I would like to highlight is the index value given to different AIML tags.

```
<category><pattern> BYE </pattern>
        <template>#0$Bye! Nice talking to you. You take care now. </template>
</category>
```

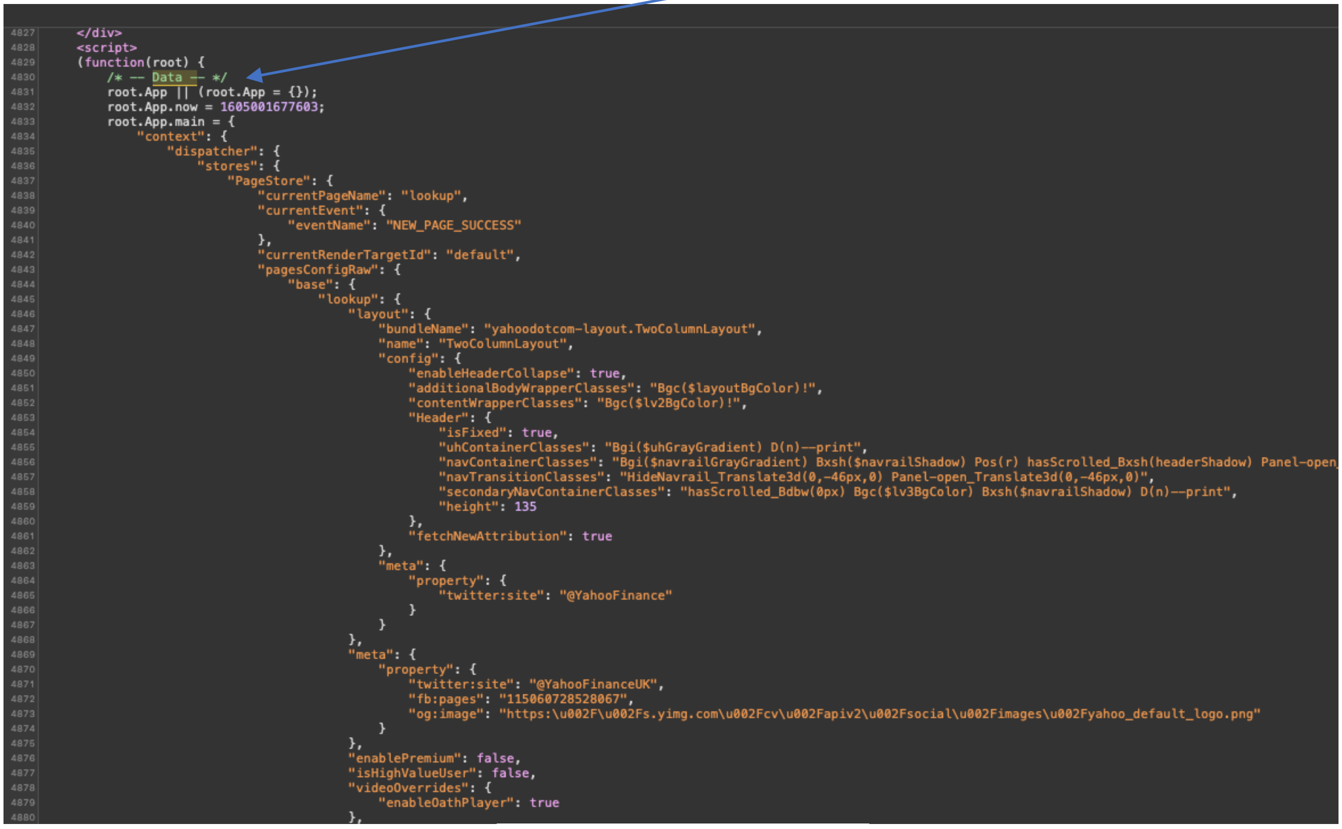
The **#0$** indicates the point of execution of a piece of which has been developed to output in a certain way to the user or even to be pre-processed so a correct result can be given to the user. In this case, when the user enters 'bye' into the chatbot, a response is provided by the chatbot, and the code in the main application is implemented to close the application.

Similarity-based component

Before I can run the models, I have specified in the previous section. It is necessary to pre-process the data before I can pass it through the model. I have learned two new processes for data pre-processing during my research into the three models: stemming and lemmatization. Stemming is used to find the base form of a word such as 'process,' 'processing,' 'processed' the stem of these three words is a process. On the other hand, lemmatization creates a group of lemmas which connect similar words together, rather than creating non-essential words. Firstly, I have read the CSV file containing predefined questions and answers and stored them in a corpus. Next, I have used the NLTK library to tokenise the text into lists of sentences and words, creating a bag of words. I have then lemmatised the tokens and normalised the text as part of the data pre-processing. I then get the user input and append the input into the variable containing the tokenised sentences. Using the TF-IDF and cosine similarity libraries, I have been able to find a similarity value and extract the correct answers from my CSV file. This code and methodology have been adapted by an external source (Pandey, 2018)

Web scraping

For this chatbot, I have implemented web scraping to fetch data from yahoo finance, such as companies' profile information and live stock prices. For this, I have inspected the elements of a yahoo finance page and understood how the page has been built. I have used the beautiful soup library to pull data from the HTML code and store it into a JSON format. In the screenshot below, I have been able to identify where the data is being held in the page source and use regular expressions; I have been able to extract the data I need, such as the company's profile information.



```
</div>
<script>
(function(root) {
    /* -- Data -- */
    root.App || (root.App = {});
    root.App.now = 1605001677603;
    root.App.main = {
        "context": {
            "dispatcher": {
                "stores": {
                    "PageStore": {
                        "currentPageName": "lookup",
                        "currentEvent": {
                            "eventName": "NEW_PAGE_SUCCESS"
                        },
                        "currentRenderTargetId": "default",
                        "pagesConfigRaw": {
                            "base": {
                                "lookup": {
                                    "layout": {
                                        "bundleName": "yahoodotcom-layout.TwoColumnLayout",
                                        "name": "TwoColumnLayout",
                                        "config": {
                                            "enableHeaderCollapse": true,
                                            "additionalBodyWrapperClasses": "Bgc($layoutBgColor)!",
                                            "contentWrapperClasses": "Bgc($lv2BgColor)!",
                                            "Header": {
                                                "isFixed": true,
                                                "uhContainerClasses": "Bgi($uhGrayGradient) D(n)--print",
                                                "navContainerClasses": "Bgi($navrailGrayGradient) Bxsh($navrailShadow) Pos(r) hasScrolled_Bxsh(headerShadow) Panel-open_
                                                "navTransitionClasses": "HideNavrail_Translate3d(0,-46px,0) Panel-open_Translate3d(0,-46px,0)",
                                                "secondaryNavContainerClasses": "hasScrolled_Bdbw(0px) Bgc($lv3BgColor) Bxsh($navrailShadow) D(n)--print",
                                                "height": 135
                                            },
                                            "fetchNewAttribution": true
                                        },
                                        "meta": {
                                            "property": {
                                                "twitter:site": "@YahooFinance"
                                            }
                                        }
                                    },
                                    "meta": {
                                        "property": {
                                            "twitter:site": "@YahooFinanceUK",
                                            "fb:pages": "115060728528067",
                                            "og:image": "https:\u002F\u002Fs.yimg.com\u002Fcv\u002Fapiv2\u002Fsocial\u002Fimages\u002Fyahoo_default_logo.png"
                                        }
                                    },
                                    "enablePremium": false,
                                    "isHighValueUser": false,
                                    "videoOverrides": {
                                        "enableOathPlayer": true
                                    },
```

*Figure 3:Web Scraping image*

I have similarly done this to isolate the current price of a stock; the pseudocode below represents my thinking behind the extraction of the data.

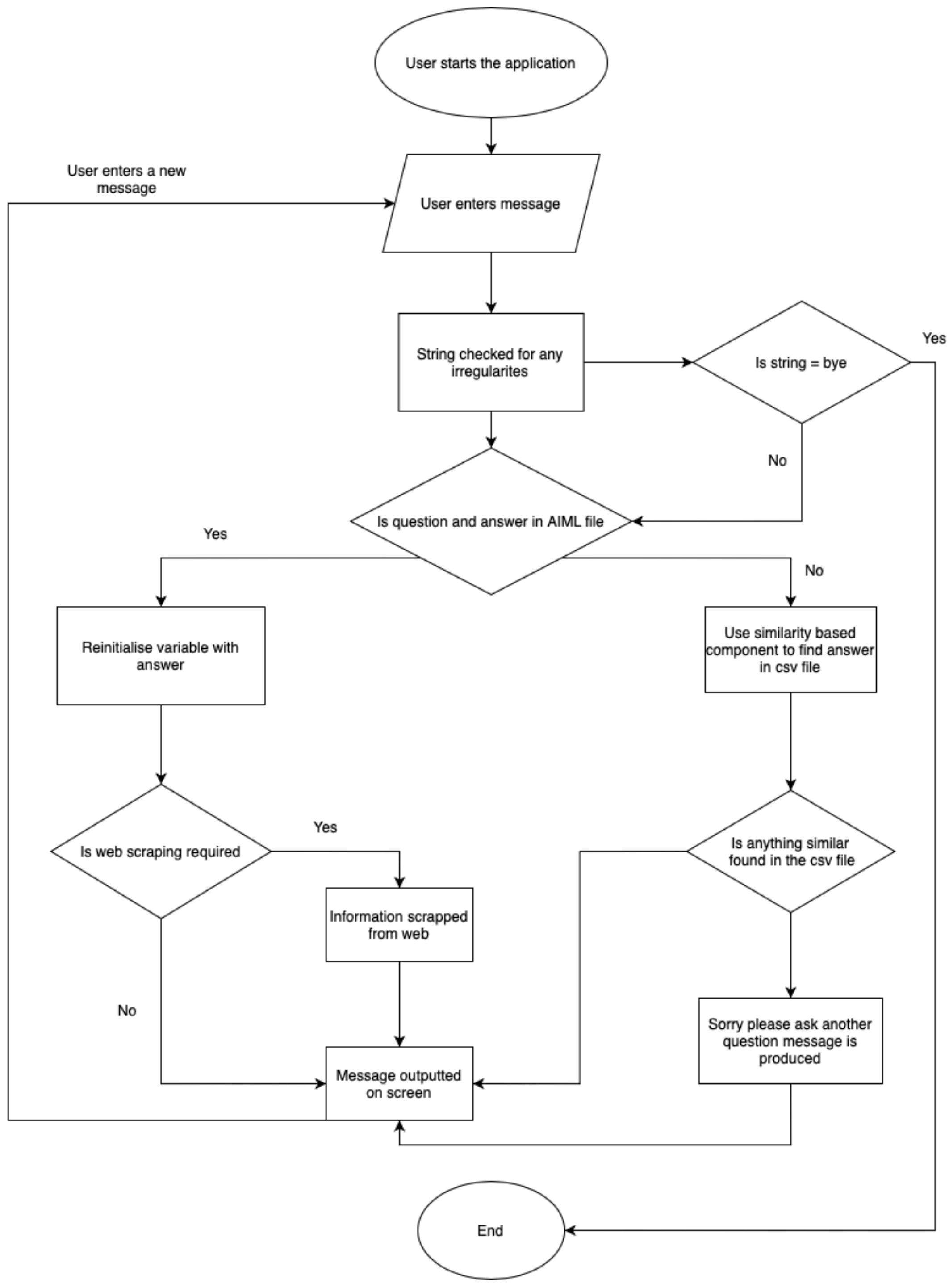
```
url_StockProfile  = " URL{}"

response = get(url.profileData(userChoice))
```

The curly brackets represent where the user's choice of stock will be entered so the correct page is opened.

```
BeautifulSoup(response, 'HTML parser')
Variable = regular expressions('/* -- Data – *\)
Json.load(variable)
print(json.data())
```

Extra features I have added in this program are creating a GUI, text to speech reader however this does not fully work with the GUI and requires more development. The chatbot can provide non-textual data for the user such as different charts.

## System Design Flowchart


User starts the application
User enters a new message
User enters message
String checked for any irregularites
Is string = bye
Yes
No
Is question and answer in AIML file
Yes
No
Reinitialise variable with answer
Use similarity based component to find answer in csv file
Is web scraping required
Yes
Information scrapped from web
No
Is anything similar found in the csv file
Sorry please ask another question message is produced
Message outputted on screen
End

## Bibliography

Li, B. & Han, L., 2013. Distance Weighted Cosine Similarity Measure for Text Classification. In: Yin H. et al. (eds) Intelligent Data Engineering and Automated Learning. *14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings,* Volume 8206, pp. 611-618.

Pandey, P., 2018. *Building a Simple Chatbot from Scratch in Python (using NLTK).* [Online]
Available at: https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e
[Accessed 10 11 2020].

Unkown, 2020. *Tutorialspoint.com.* [Online]
Available at: https://www.tutorialspoint.com/aiml/index.htm
[Accessed 10 11 2020].

Izzy Analytics (2020). How to scrape STOCKS and FINANCIALS from YAHOO! Finance with PYTHON. YouTube. Available at: https://www.youtube.com/watch?v=fw4gK-leExw&t=220s [Accessed 10 Nov. 2020].

straight_code (2019). Real Time Stock Price Scraping with Python and Beautiful Soup. YouTube. Available at: https://www.youtube.com/watch?v=rONhdonaWUo [Accessed 10 Nov. 2020].

Ntu.ac.uk. (2020). *Week 2 Slides - ISYS30221: Artificial Intelligence 202021 Full Year*. [online] Available at: https://now.ntu.ac.uk/d2l/le/content/716407/viewContent/5043986/View [Accessed 10 Nov. 2020].

MonkeyLearn Blog. (2019). *What is TF-IDF?* [online] Available at: https://monkeylearn.com/blog/what-is-tf-idf/ [Accessed 10 Nov. 2020].

Izzy Analytics (2020). How to scrape STOCKS and FINANCIALS from YAHOO! Finance with PYTHON. YouTube. Available at: https://www.youtube.com/watch?v=fw4gK-leExw&t=220s [Accessed 10 Nov. 2020].

straight_code (2019). Real Time Stock Price Scraping with Python and Beautiful Soup. YouTube. Available at: https://www.youtube.com/watch?v=rONhdonaWUo [Accessed 10 Nov. 2020].

Ntu.ac.uk. (2020). *Week 2 Slides - ISYS30221: Artificial Intelligence 202021 Full Year*. [online] Available at: https://now.ntu.ac.uk/d2l/le/content/716407/viewContent/5043986/View [Accessed 10 Nov. 2020].

MonkeyLearn Blog. (2019). *What is TF-IDF?* [online] Available at: https://monkeylearn.com/blog/what-is-tf-idf/ [Accessed 10 Nov. 2020].