

Artificial Intelligence

STAGE 1: CHATBOT WITH RELATED AND SIMILARITY-BASED COMPONENT

STAGE 2: ADD IMAGE CLASSIFICATION COMPONENT

STAGE 3: ADD A KNOWLEDGE BASED COMPONENT

STAGE 4: CHATBOT EXTENDED WITH CLOUD-BASED MULTI-LINGUAL COMPONENT

RISHI SINGH 2018 (N0834113)

Table of Contents

Stage 1: Technology Stocks Chatbot.....	3
System explanation and goals	3
System requirements	3
Explaining the employed AI techniques	4
Rule-based component.....	4
Similarity-based component.....	5
Bag of Words	5
TF-IDF.....	5
Cosine Similarity	5
Explanation of the program.....	6
AIML	6
Similarity-based component.....	6
Web scraping.....	7
System Design Flowchart	8
Stage 2: Technology Stock Chatbot.....	9
System explanations and goals.....	9
Added System Goals.....	9
Added System requirements	10
Must:.....	10
Should:.....	10
Could:	10
Explaining AI employed techniques.....	10
Dataset	10
Convolutional Neural Networks	11
Extra AI techniques.....	12
VGG16.....	12
ResNet50.....	13
Explanation of program	13
Convolutional Neural Network.....	14
News API.....	15
Extra AI techniques	16
Stage 3: Technology Stock Chatbot.....	17
Added System Goals.....	17
Added System requirements	18
Must:.....	18
Should:.....	18
Could:	18
Knowledge-based component:	18
Extra AI techniques.....	21
Stage 4: Chatbot extended with cloud-based multi-lingual component.....	23
Added System requirements	23
Must:.....	23
Should:.....	23
Could:	23
Added System Goals.....	24

Cloud-based Multi-lingual component	24
Detection of language	25
Translation of language.....	26
Limitations	26
Translation design diagram	27
Additional Features	28
Text to speech.....	28
Brand detection.....	28
Sentimental Analysis	28
<i>Bibliography</i>.....	29

Stage 1: 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.
	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.

$$\begin{aligned}tf &= \frac{\text{number of occurrences of a word}}{\text{total word length of sample}} \\ \text{Inverse Document Frequency:} \\idf &= \log \frac{\text{number of samples}}{\text{number of samples that contain the word}} \\tfidf &= tf \cdot idf\end{aligned}$$

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.

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

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>
```

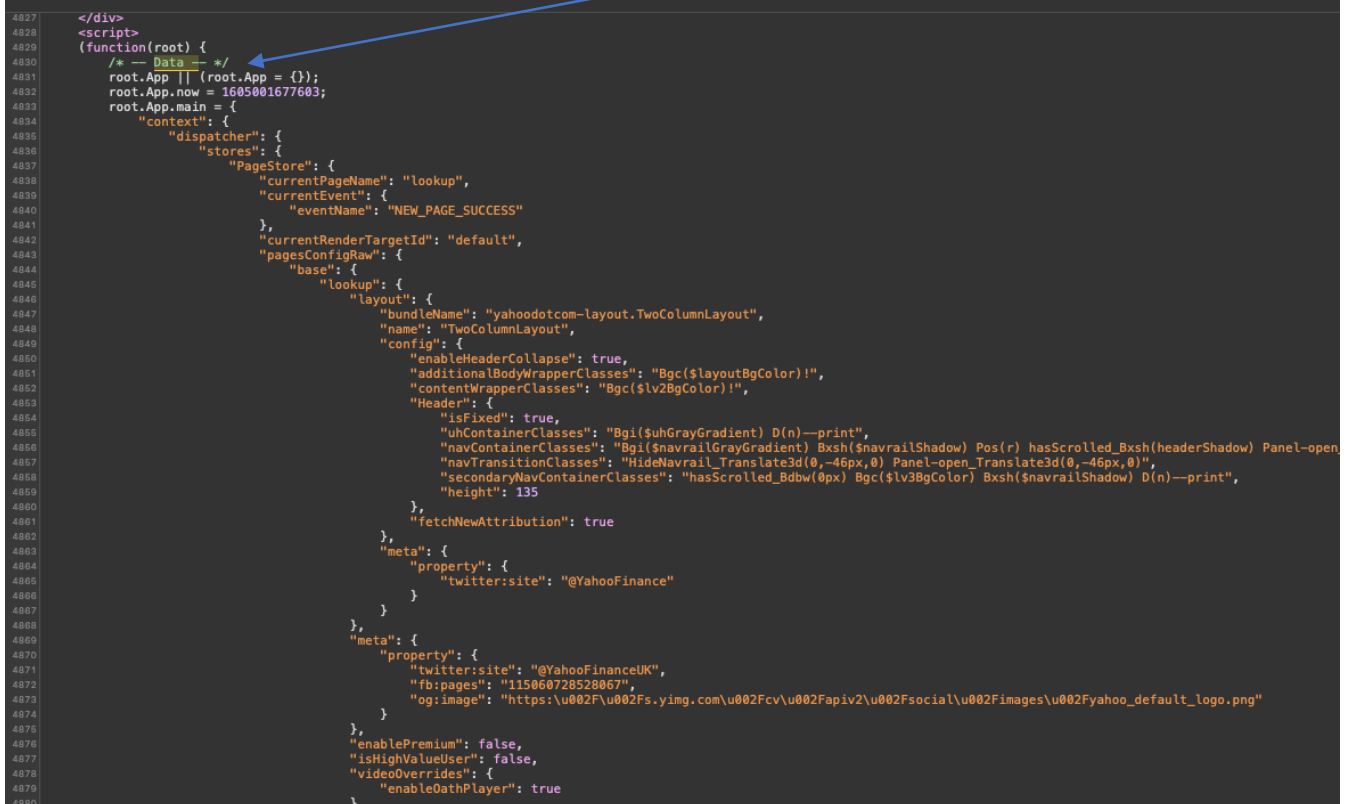
The **#0\$** indicates the point of execution of a piece of which has been developed to output in a certain way to the user or even to be pre-processed so a correct result can be given to the user. In this case, when the user enters ‘bye’ into the chatbot, a response is provided by the chatbot, and the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



```
4827 </div>
4828 <script>
4829 (function(root) {
4830     /* -- Data -- */
4831     root.App || (root.App = {});
4832     root.App.now = 1605001677603;
4833     root.App.main = {
4834         "context": {
4835             "dispatcher": {
4836                 "stores": {
4837                     "PageStore": {
4838                         "currentPageName": "lookup",
4839                         "currentEvent": {
4840                             "eventName": "NEW_PAGE_SUCCESS"
4841                         },
4842                         "currentRenderTargetId": "default",
4843                         "pagesConfigRaw": {
4844                             "base": {
4845                                 "lookup": {
4846                                     "layout": {
4847                                         "bundleName": "yahoodotcom-layout.TwoColumnLayout",
4848                                         "name": "TwoColumnLayout",
4849                                         "config": {
4850                                             "enableHeaderCollapse": true,
4851                                             "additionalBodyWrapperClasses": "BgC($layoutBgColor)!",
4852                                             "contentWrapperClasses": "BgC($lv2BgColor)!",
4853                                             "Header": {
4854                                                 "isFixed": true,
4855                                                 "uhContainerClasses": "Bgi($uhGrayGradient) D(n)—print",
4856                                                 "navContainerClasses": "Bgi($navrailGrayGradient) Bxsh($navrailShadow) Pos(r) hasScrolled_Bxsh(headerShadow) Panel-open",
4857                                                 "navTransitionClasses": "HideNavrail_Translate3d(0,-46px,0) Panel-open_Translate3d(0,-46px,0)",
4858                                                 "secondaryNavContainerClasses": "hasScrolled_Bdbw(0px) Bgc($lv3BgColor) Bxsh($navrailShadow) D(n)—print",
4859                                                 "height": 135
4860                                             },
4861                                             "fetchNewAttribution": true
4862                                         },
4863                                         "meta": {
4864                                             "property": {
4865                                                 "twitter:site": "@YahooFinance"
4866                                             }
4867                                         },
4868                                         "meta": {
4869                                             "property": {
4870                                                 "twitter:site": "@YahooFinanceUK",
4871                                                 "fb:pages": "115060728528067",
4872                                                 "og:image": "https://u002F\u002Fs.yimg.com\u002Fcv\u002Fapiv2\u002Fsocial\u002Fimages\u002Fyahoo_default_logo.png"
4873                                             }
4874                                         },
4875                                         "enablePremium": false,
4876                                         "isHighValueUser": false,
4877                                         "videoOverrides": {
4878                                             "enableOathPlayer": true
4879                                         }
4880                                     }
4881                                 }
4882                             }
4883                         }
4884                     }
4885                 }
4886             }
4887         }
4888     }
4889 }
4890 )
```

Figure 3: Web Scraping image

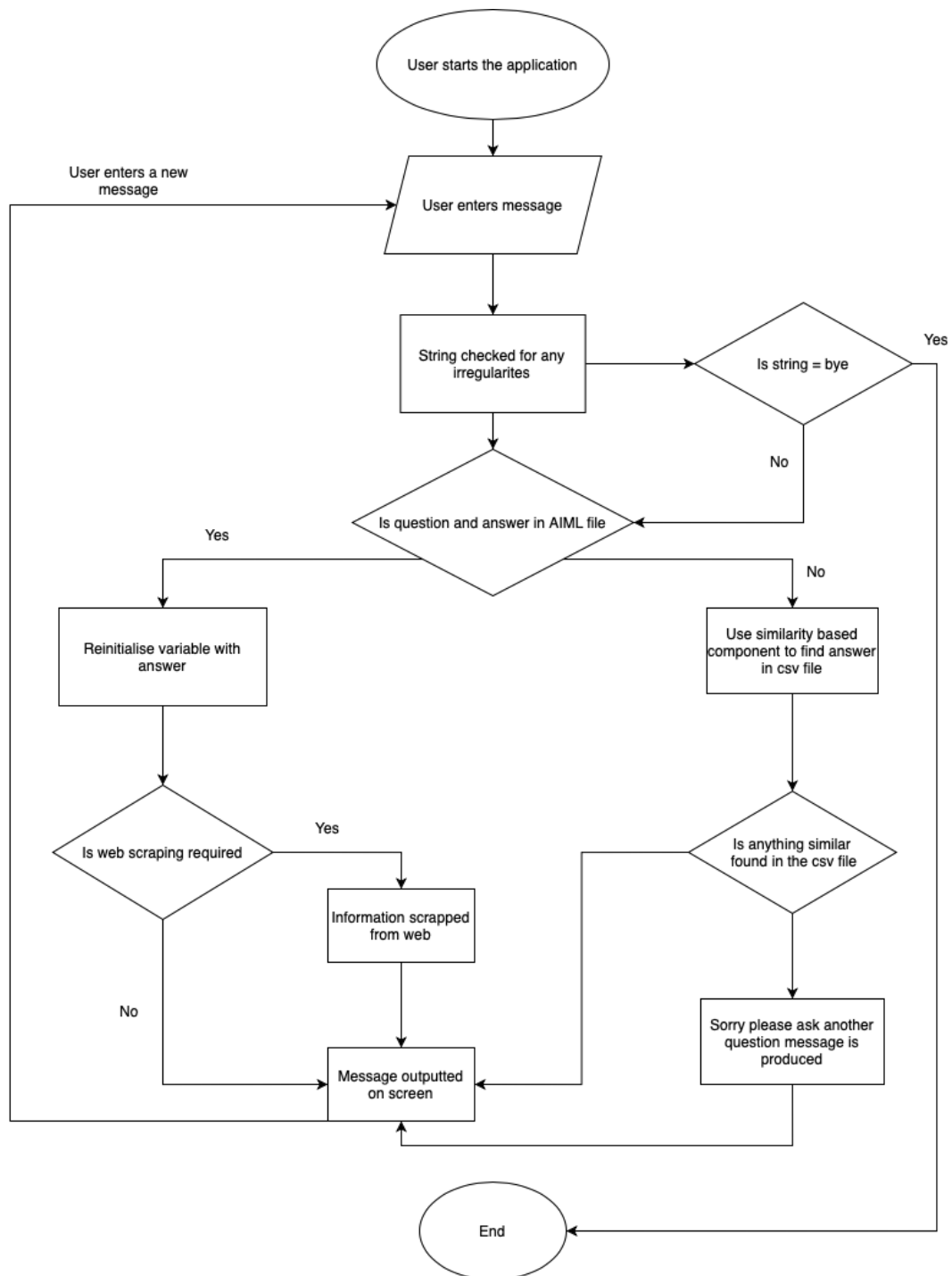
I have similarly done this to isolate the current price of a stock; the pseudocode below represents my thinking behind the extraction of the data.

```
url_StockProfile = "URL {}"
response = get(url.profileData(userChoice))
BeautifulSoup(response, 'HTML parser')
Variable = regular expressions('/* -- Data -- *')
Json.load(variable)
print(json.data())
```

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

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



Stage 2: Technology Stock Chatbot

System explanations and goals

For this project, I have developed this project further as the new chatbot includes three different types of convolutional neural networks (CNN) which allow the user to enter an image of a brand. The CNN consists of a large dataset that includes 29 different classes and a total of (certain images). The dataset has been taken from Kaggle (<https://www.kaggle.com/momotabanerjee/brand-logo-recognition-dataset>) adapted and some extra images have been added to the dataset to provide better results. A larger dataset of images would have been more suited for this project however due to the specific domain the project has had to go outside of the domain slightly and classify different types of brands. Initially, the chatbot was created to give users information about technology stocks, this initial aim is still carried on in the development however, the CNN includes some brands which are not technology companies. Users will be able to enter new images into the chatbot which will classify the brand.

The primary aim of this part of the project is to classify images that consist of brands that have been done using three different CNN architectures. The first is a custom CNN architecture which has been adapted to the dataset using TensorFlow documentation. The second CNN architecture is a pre-trained model using Keras: VGG16 and the third CNN architecture is a ResNet50 architecture. Further explanation of both of these models have been given in later in the document.

I have been able to update the previous stage and I have incorporated the use of News API to give more information about specific companies. From the CNN output, I have been able to merge both of these ideas as the CNN classifies the brand in an image and then passes the name of the brand to the News API where I have been able to find the top headlines of the specific brand. By classifying images and gaining extra information, users will be able to get extensive information about a specific brand that has been classified.

Added System Goals

- A functioning CNN classifies images of brands into correct categories.
- Integration of chatbot and CNN.
- Extra information about brands that have been classified.
- The system saves the CNN model in a .h5 format.
- The system uses a pre-trained and custom CNN architecture.
- Error-free chatbot.
- Successful chat between chatbot and user.
- Ability to send non-textual data to the chatbot.

Added System requirements

Must:

- The system must allow the user to upload an image using the directory name.
- The system must allow users to upload an image through the browse file.
- The system must include a CNN.
- The system must be able to read a .h5 file containing CNN.
- The system must have a CNN integrated with the chatbot.
- The system must have an output from CNN.
- CNN output must be in line with user input.
- CNN must adhere to the chatbot domain.

Should:

- The system should allow images with a .jpg extension.
- The system should consider other CNN models to get the best accuracy for optimal accuracy.
- The system should allow users to enter a different type of questions to get to the CNN.
- The system should classify unknown images as none.
- The system should have an accuracy of above 50%

Could:

- The system could output relevant news headlines and articles related to CNN output
- The system could have a larger dataset.
- The system could have pre-trained models such as ResNet or VGG16. The system could have an accuracy of 75% or above.
- The system could have a new GUI colour scheme.

Explaining AI employed techniques

Dataset

For this project, I have been able to get a dataset from Kaggle of around 1000 images with 29 different classes of brands. These brands include Adidas, Apple, BMW, Citroen, CocaCola, DHL, FedEx, Ferrari, Ford, Google, HP, Heineken, Intel, McDonald's, Mini, Multiple, NBC, Nike, None, Pepsi, Porsche, Puma, Redbull, Sprite, Starbucks, Texaco, Unicef, Vodafone, Yahoo. The dataset is split into 80% for training and 20% for testing data made of 874 to 219 images respectively. Due to the small dataset, there will be an impact on the accuracy but this size of data is showing promise in classifying the correct brands.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a 'class of artificial neural networks' (Yamashita, 2018) which has become very popular in recent times attracting a lot of attention in different domains such as the medical sector, financial sector, and many others. CNN is a deep learning model that processes image data, designed to classify the contents of an image that are provided. There are two important types of classification CNN models can have, the first is a binary image classifier and the second is a categorical image classifier.

A binary image classifier can be used to categorised images into yes or no, good or bad categories where there are two distinct answers. For example, a dataset could be provided of ultrasound images which might consist of good ultrasound images with an appropriate angle along with a set of bad ultrasound images with irregular angles. Here the CNN can categorise new ultrasound images into two binary categories good represented by 1 and bad represented by 0. On the other hand, a categorical image classifier can have many different classes where new images can be classified into certain classes. The main difference between a binary and categorical image classifier is that in a binary image classifier there are only two categories available whereas a categorical image can more than two different classes for image classification.

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

Figure 4: Convolution Mathematical Formula

In a convolutional neural network, there are four main stages which include a convolutional operation, ReLU layer, Pooling, Flattening, and a Full connection. The convolution operation can be represented by a mathematical formula that represents the integration of two functions as shown above. The convolution process is where a small matrix of numbers is taken passed over an image and is transformed depending on the values from the filter. f is denoted by the input image into the CNN and g represents the kernel. After the application of the filter on the image, the values from the kernel are multiplied by the pixel values from an image, and a feature map is created as a result. To obtain a convolutional layer several feature maps are created and compiled after the application of several filters and multiplication of the kernel.

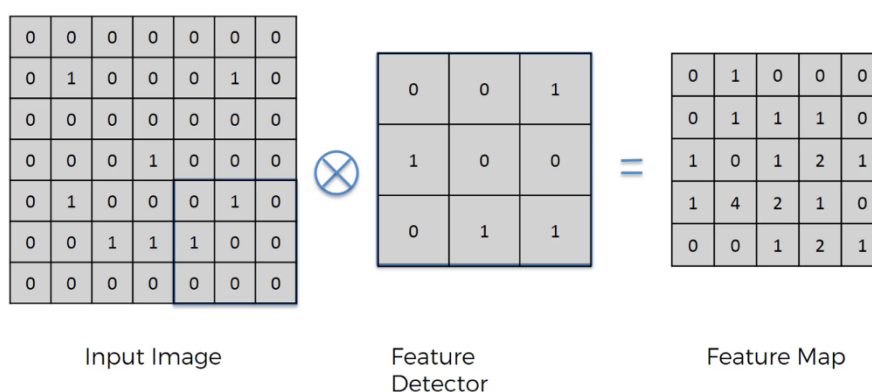


Figure 5: Kernel multiplication

A ReLU layer is applied in the next stage which is known as a rectified linear activation function as this layer makes it easier to train the model and increase the performance of the model. From the feature map produced a pooling operation is applied, this is done by moving a small box over the input image and a maximum value in that small box is chosen. Hyperparameters such as filter size and stride can also be varied to perform this pooling stage. After the pooling stage, the pooled feature map is then flattened which is a stage where the pooled feature maps are converted into a long vector and is used as input layer of a future artificial neural network (ANN).

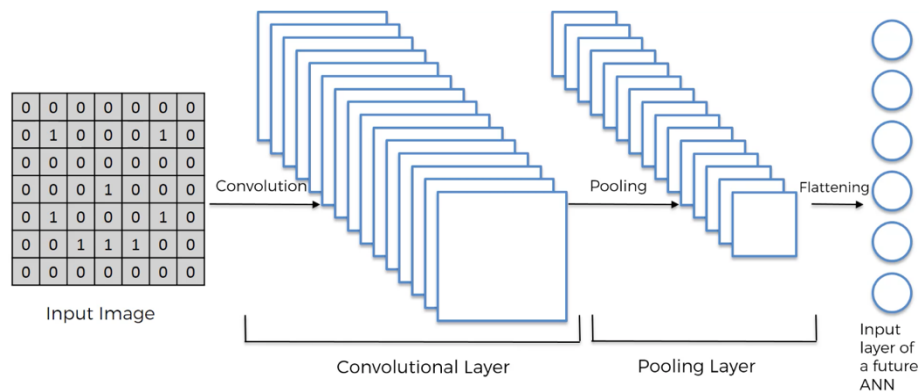


Figure 6: CNN representation

Finally, a full connection is produced which is an ANN consisting of three layers: an input layer, a hidden layer, and an output layer. Each layer has a different number of neurons which reduce or increase after each layer specifically in the hidden layers and finally produce an output layer that can be used to classify a final output.

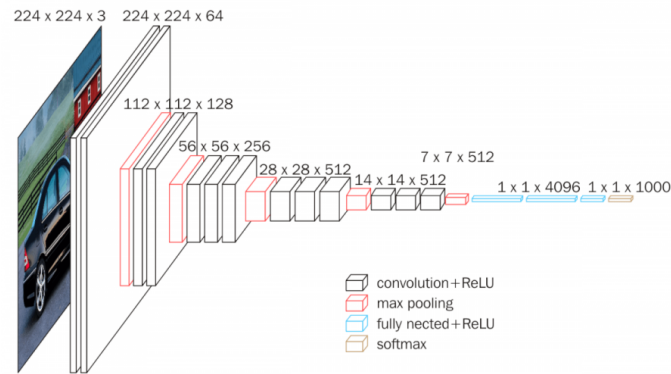
Extra AI techniques

For this project, I have researched CNN models which have already been trained and an architecture has been provided. There are several pre-trained models some of the most popular ones include: ReNet50, VGG16, Inceptionv3 otherwise known as GoogLeNet, and EfficientNet. I have chosen to use the Keras library applying the ResNet50 CNN model and the VGG16 model for my current data set.

VGG16

VGG16 is a pre-trained CNN model that has been created by the Visual Geometric Group. This network contains 16 layers with a standard number of parameters used to get the best accuracy. VGG16 can take input images that have an image size of 224X224 with three channels which are RGB (Red, Green, and Blue). The Vgg-16 model can reach an accuracy of 92% with a top-5 test accuracy in ImageNet exceeding the abilities of AlexNet consisting of multiple 3X3 kernel-sized filters that replace the large kernel that the AlexNet has. The architecture can be seen below the use of the softmax function at the end of the model. The softmax function is an activation layer that returns a vector consisting of a list of potential outcomes shown by a distribution of probabilities.

Figure 7: VGG16 visualisation



ResNet50

A Residual Network known as a ResNet is a deep neural network consisting of more than 150 layers. ResNet introduces a layer known as the 'skip layer' which allows you to take the activation of one layer and feed it to a layer that is deeper in the neural network in most ResNets this is skipping to weighted layers. By skipping these layers, there is a directly proportional relationship between the training error and the number of layers; as the number of layers increases the training layer decreases. In the mathematical equation below x represents the input into the model x then skips two weighted layers which in this case is a ReLU activation. Without these skip connections, some large neural networks with a large number of layers can have problems selecting the correct parameters. The skip layer can be seen in the diagram below.

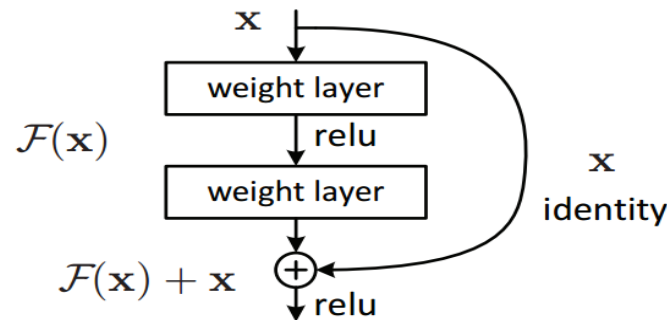


Figure 8: Skip Connection

Explanation of program

In this section, I will be explaining the functionality I have added to the previous chatbot. The main new features are the brand image classification I have added using deep learning methods such as CNN which allows the user to enter an image into the system and the model can return the type of brand present in the image along with the accuracy percentage. I have highlighted some key new features in my program which I would like to explain.

Convolutional Neural Network

I have used a dataset that contains around 1100 images of 29 different brands which have been split into test and training sets for my convolutional neural network. With the use of Keras and TensorFlow, I have been able to load this data which can be seen below by my data/image pre-processing. Here I have successfully been able to load my dataset and split the data into a training set that contains 80 % percent of the data and the test which holds 20% of the data. The program sets the image size as 180x180 and has identified 29 different classes and which are ready to be pre-processed and have been displayed before the commencement of the model.

```
In [21]: img_height = 180
img_width = 180
batch_size = 32

In [22]: train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "LogosNew/Train",
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)

Found 1122 files belonging to 29 classes.
Using 898 files for training.

In [23]: val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "LogosNew/Train",
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size)

Found 1122 files belonging to 29 classes.
Using 224 files for validation.

In [24]: class_names = train_ds.class_names
print(class_names)

['Adidas', 'Apple', 'BMW', 'Citroen', 'Cocacola', 'DHL', 'Fedex', 'Ferrari', 'Ford', 'Google', 'HP', 'Heineken', 'Intel', 'McDonalds', 'Mini', 'Multiple', 'Nbc', 'Nike', 'None', 'Pepsi', 'Porsche', 'Puma', 'Redbull', 'Sprite', 'Starbucks', 'Texaco', 'Unicef', 'Vodafone', 'Yahoo']
```

Figure 9:Image Pre-processing

I have then been able to then visualise the data showing 29 different image classes followed by configuring the data set using the Dataset.cache() and Dataset.prefetching() function to keep images in the memory after running an epoch. Dataset.prefetching() overlaps the pre-processing and the model execution while training, by doing this the data is read faster and the speed of training is increased. This has been followed by standardising the data and rescaling the RGB channels leading to data augmentation which is an important aspect. The data augmentation uses images from the dataset and generates an additional number of images that have been transformed so more similar images can be yielded and prepared to make the model stronger. Finally, the CNN model is created which includes several layers such as the convolution, pooling, dropout, flatten and the dense using a ReLU activation throughout. I have adapted a convolutional neural network from the TensorFlow website which can include three convolutions with increasing filters. In-between these three convolutions I have included three max-pooling functions. After the final max-pooling, I have included a dropout to the network to work around any overfitting the parameters I have used for this is 0.2 which means 20% of the output units will be dropped randomly. I have then compiled and fitted the model running 49 epochs as I found my best accuracy at this point of 0.9404. I have then plotted the validation and training accuracy along with the validation and training loss which can be seen underneath.

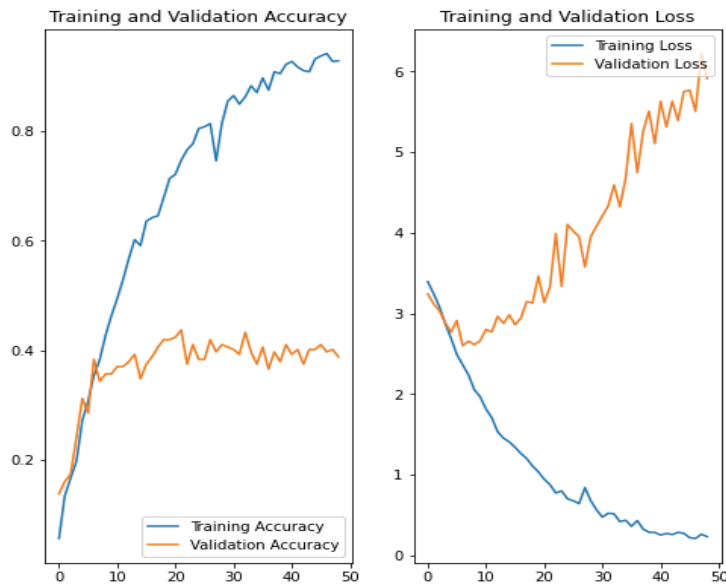


Figure 10: Training and Validation

Pseudocode Convolutional Neural Network:

Classes = 29

```
Model = Sequential ([
    Image.preprocessing.rescaling(1./255),
    Conv2d(filters=16,kernal=3,padding = same, activation = relu),
    MaxPooling(),
    Conv2d(filters=32,kernal=3,padding = same, activation = relu),
    MaxPooling(),
    Conv2d(filters=64,kernal=3,padding = same, activation = relu),
    MxPooling()
    Dropout(0.2)
    Flatten()
    Dense(128, activation = relu)
    Dense(Classes)
])
```

Model.summary()

Model.compile(adam, SparseCategoricalCrossentropy, accuracy)

Model.fit(x_train, y_test, epochs = 49)

News API

I have also added a new feature from stage 1 which is using the News API to retrieve information about a certain brand which has been classified by the CNN. I have done this using the News API and generating an API key then taking in the answer produced by the CNN and searching that specific brands headline. This can be seen by the pseudocode produced below:

Pseudocode Convolutional Neural Network:

```
def gen_brandNews():
```

```
    urlNews = http://newsapi.org/v2/everything? + CNNClass + "&from=2020-12-11&sortBy=publishedAt&apiKey=6c2ceae5f916435183851b58debd63de"
    open_page = requests.get(urlNews).json()
    Article = open['article']
    resultsURL = []
    webBrowser.open(resultsURL[1])
```

Extra AI techniques

I have also incorporated some extra AI techniques such as using pre-trained CNN models which can provide similar or even better accuracy. This gave me a wider reading of the CNN architecture and I was able to compare existing CNN architectures with pre-trained models. I have used the Keras library to use the pre-trained models from *tKerasas.applications.VGG16*. It is pivotal to make sure the input shape in the model has a size of 224 and a number of channels 3.

A model summary can be seen below:

```
In [24]: x = Flatten()(vgg.output)
prediction = Dense(len(Folders), activation='softmax')(x)
model = Model(inputs=vgg.input, outputs=prediction)
model.summary()

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359008
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359008
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359008
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359008
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359008
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
Flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 29)	727581

Total params: 15,442,269
Trainable params: 727,581
Non-trainable params: 14,714,688

Figure 11: Model Summary

Stage 3: Technology Stock Chatbot

For this project, I have developed previous technology stock chatbot to a further extent which includes the implementation of a Knowledge base chatbot, the use of Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) to highlight the prediction of Google stock prices from previous historical data and finally I have been able to incorporate another CNN architecture carrying on from my research into which CNN architecture is the best to use for a brand logo recognition/classification.

Firstly, in this project, I have been tasked to implement a knowledge-based component that will support the chatbot in human-like decision-making. In this project, I have been able to create a CSV file with several rules that the chatbot has learned and uses the NLTK library which provides several functions and classes for First-Order Logic and specifically syntax structures for representing logical statements. These logical statements include the use of negation, conjunction, disjunction, and implication. Within this implementation, I have highlighted key sections of code that deal with any contradictions the chatbot may face along with any false detection which will be explained later in this document.

Secondly, as an additional artificial intelligence technique, I have created a stock predicting model using an LSTM approach which is a variation of the Recurrent Neural Network. In this implementation, I have used various libraries and obtained a dataset from the Yahoo Finance website for Google. The data in this dataset have a time date from August 2004 to February 2021 (<https://uk.finance.yahoo.com/quote/GOOG/history?p=GOOG>). During this implementation, I have been able to create which explores the use of LSTM, Dropout, Dense, and the use of sklearn's MinMaxScaler. I have given more details regarding these specific attributes later in this documentation.

Thirdly, to carry on from my previous implementation I have been able to implement another CNN architecture in pursuit to find the best architecture for the brand logos data set I currently have. In the previous stage, I was able to implement the VGG-16 model and the Resnet-50 model. In this stage, I will be exploring the implementation of the MobileNet model along with hyper-tuning the learning rate and decay.

Added System Goals

- Functioning knowledge-based chatbot.
- Ability to add new knowledge to the chatbot.
- Chatbots ability to provide relevant and accurate answers
- System uses LSTM for the prediction of stock prices for Google
- Error-free chatbot.
- Successful chat between chatbot and user.
- Exploration of the Inception CNN architecture.

Added System requirements

Must:

- System must allow the user to enter a knowledge-based or statement.
- System must be able to check the statement.
- System must be able to inform the user regarding contradictions.
- System must be able to provide false detection for statements.
- System must allow multiple inputs for logic statements.
- System must inform the user if statements already exist.

Should:

- The system must allow users to view the results from the LSTM stocks prediction.
- System should make use of a real data set from yahoo finance.
- System should use a scaler factor for the LSTM.
- System should produce a graph of real stock prices and predicted stock prices

Could:

- System could be improved with the use of MobileNet CNN architecture
- System could have a large dataset for LSTM.
- MobileNet could have higher accuracy than at least 75% for comparison.
- System could analyse the accuracy and value loss of CNN architecture.

Knowledge-based component:

For this stage of the coursework, I have been able to implement a Knowledge-Based (KB) component to my chatbot which conversates about stocks. I have created a knowledge base which is an organised set of information regarding stocks specifically I have created a CSV file that contains the knowledge itself but as well as the inference engine which is the set of rules that connect the knowledge-based data. During my research into KB, I have come across an 'Expert System' which has a very detailed design of a KB implementation.

The figure below is an example of a design that I have adapted to create the KB agent for my chatbot (Heidenreich, 2018). The figure below is an example of an Expert System which is 'computer applications developed to solve complex problems in a specific domain. An Expert System consists of components that have been highlighted in the figure below: a Knowledge Base, Inference Engine, and a user interface. A similar design of the KB system can be adopted from this design as the user can enter KB data into the chatbot which would acknowledge whether the statement entered already exists in the system or whether the statement contradicts the knowledge that is already present in the system or whether the data entered can be appended to the system. The crucial element of this system is the Inference Engine itself as the defined rules are required to be accurate through logic used.

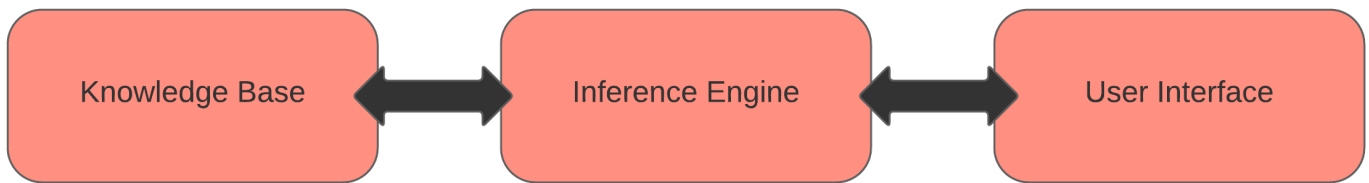


Figure 12: Knowledge base Design (Heidenreich, 2018)

Symbols

To create an Inference Engine, it was necessary to understand the propositional logic used to create the set of rules which help in making decisions for the chatbot. Using the NLTK documentation (Nltk.org, 2019), propositional logic is a 'logical language used to make reasoning formally explicit'. NLTK has defined five main Boolean operators which the system will use to create a KB component in the chatbot. In the table below are the four main Boolean operators that I have used to create the KB component of the chatbot.

BOOLEAN OPERATOR	SYMBOL	MEANING
NEGATION	-	NOT THE CASE
CONJUNCTION	&	AND
DISJUNCTION		OR
IMPLICATION	→	IF...THEN....

Example:

An example of one of these Boolean operators can be seen by the use of a conjunction operator and an implication operator. In the example below if there is a company and it is not overvalued then the statement buy must be correct. A worded example for the understanding of this example could be: Amazon is a company and its stock price is not overvalued therefore Amazon stocks could be bought.

Company(r) & -overvalue(r) -> buy(r)

Contradiction & False Detection.

In this system, it is crucial for the KB component to be able to inform the user about any contradicting statements and statements which have a false detection. False detection is crucial as statements that returns a FALSE statement cannot be interpreted as an incorrect statement. Thus, it is important to follow up with a negation of the original statement as the original statement could be true. To avoid any confusion and for the chatbot to be accurate it is important for the system to be able to carry out a false detection. Below I have written some pseudocode to demonstrate how the chatbot can carry out a contradiction and false detection.

The code below demonstrates a contradiction when the expression extracted from a statement is passed through the ResolutionProver() if a true statement is returned then a

contradiction has occurred this can be seen by the figure below which shows the exact point of a contradiction occurring.

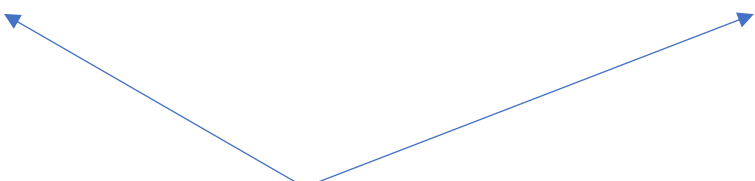
Pseudocode for contradiction:

```
statementCheck = False
object,subject = statementEntered.split
expression = read(subject(object)) ##Company(Amazon)
answer = ResolutionProver().prove(expression)
if expression exists in row:
    check = True
    print("Statement already exists")

if answer = False
KnowledgeBase.append(expression)
Else:
    print("this contradicts our knowledge")
```

Figure 13: KB contradiction

Company(Llyods)	
[1] {-Company(Llyods)}	A
[2] {Company(Llyods)}	A
[3] {-Company(z103), Technology(z103)}	A
[4] {-Company(z104), overvalue(z104), buy(z104)}	A
[5] {-Company(z106), -Sales(z106), buy(z105)}	A
[6] {-Company(e), -Technology(e), Fang(e)}	A
[7] {Strategy(z107)}	A
[8] {Bluechip(z108)}	A
[9] {Technology(Onepus)}	A
[10] {Technology(Amazon)}	A
[11] {Technology(Apple)}	A
[12] {Technology(IBM)}	A
[13] {Company(Reliance)}	A
[14] {Amazon(Bezos)}	A
[15] {Google(Pichai)}	A
[16] {Facebook(Zuckerburg)}	A
[17] {Company(Onepus)}	A
[18] {}	(1, 2)



Line 18 shows a the ResolutionProver returning an empty curly bracket which means a contradiction has occurred. This can be seen by statements lines and 2 contradicting each other.

Pseudocode for False Detection

```
Answer = ResolutionProver(expression)
If Answer = True:
    Print("Answer is correct")
Else
    Answer = ResolutionProver("¬" + expression)
If Answer:
    Print("This may be correct")
Else:
    Print("This is incorrect")
```

For a false detection, if the first answer is true then the chatbot can confirm the logic being entered is correct otherwise the statement might be true. To distinguish between if the statement is completely incorrect or partially true a negation of the original statement is required. If the answer to that statement is false then the statement can be concluded as incorrect. This can be seen by the pseudocode displayed above.

Extra AI techniques

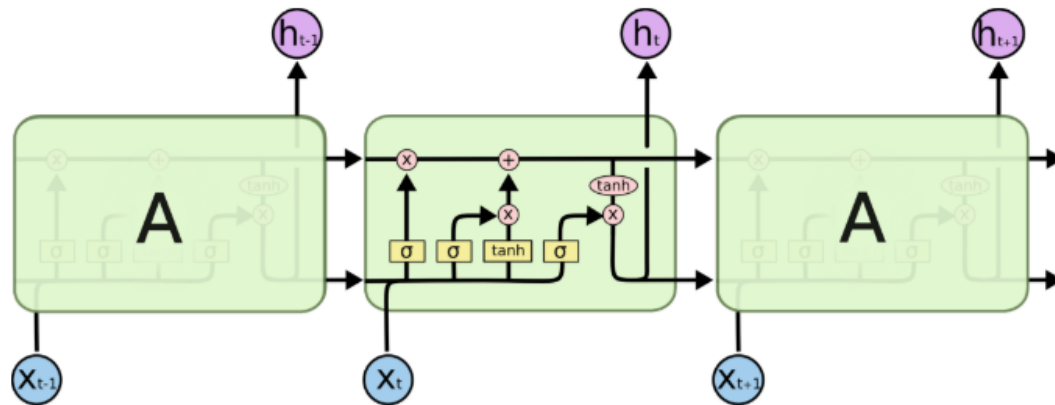
Stock prediction LSTM

In this stage of the project, I have also been able to implement and integrate a stock prediction component-based on historical data into the chatbot. The user will be able to view a graph concluding the actual stock price of Google and seeing if the LSTM model was able to predict the correct prices of stock. The dataset I have used is from the Yahoo Finance website where I was able to download historical data of Google from August 2004 to February 2021.

Long Short-Term Memory is a solution that can be used to solve problems with Recurrent Neural Networks. Some key problems with RNN's are the vanishing gradient problem where when the gradient becomes smaller and smaller, updates from the parameters are insignificant, and hence the rate of learning decreases, hampering the ability of the network. LSTM's can solve this problem by getting RNN's to remember information for a long time. LSTM's were introduced by Hochreiter et al who suggest the use of memory cells and unit gates' can solve the problem of backpropagation and the vanishing gradient (Hochreiter, Rovelli and Winckler, 1997). Hochreiter was able to design a memory cell using these unit gates (linear and logistic) to solve the problem. Each LSTM contains three cells, the input, forget and output gates which allow the cell to be updated, memory set to 0, and show the visibility of the current information. The output of these three gates or layers of cells can be described as a hidden state. The figure below shows a visual representation of an LSTM cell.

In this implementation, it was important to scale the features using the sklearn library and the MinMax scaler, however, there is debate regarding the scaling feature as stock prices can increase to whatever amount hence giving a bound problem so it is suggested that a

standard scaler is used instead. Although, for this implementation, I have used the MinMax scaler along with within regressor model I have defined the **LSTM units** as 50, a **dropout** of 0.2, and **Dense units** of 1. Along with this, I have used the Adam optimizer running 5 epochs and a batch size of 32.



The repeating module in an LSTM contains four interacting layers.

MobileNet

Carrying on from the second stage where I had introduced the use of Convolutional Neural Networks I have carried on working and trying to understand the variety of CNN architectures that have already been implemented. In the second stage, I successfully implemented a ResNet architecture and a VGG16 architecture from which the VGG16 architecture outperformed the ResNet. I wanted to continue this comparison and introduce another architecture that could have similar or even better results. Therefore, I have been able to implement a MobileNet architecture using the Keras library.

Compared to a VGG16 model a MobileNet has a size of 17MB and 4,200,000 parameters whereas the VGG16 is much greater consisting of a size of 553MB and 138,000,000 parameters. MobileNets have been described to be smaller and faster than VGG16 models however they are also described to be less accurate than bigger models. MobileNets are 'streamlined architectures' (Wang et al., 2020) that comprise of depthwise separable convolutions which are lightweight and provide an efficient model. The diagram below shows the design of the model.

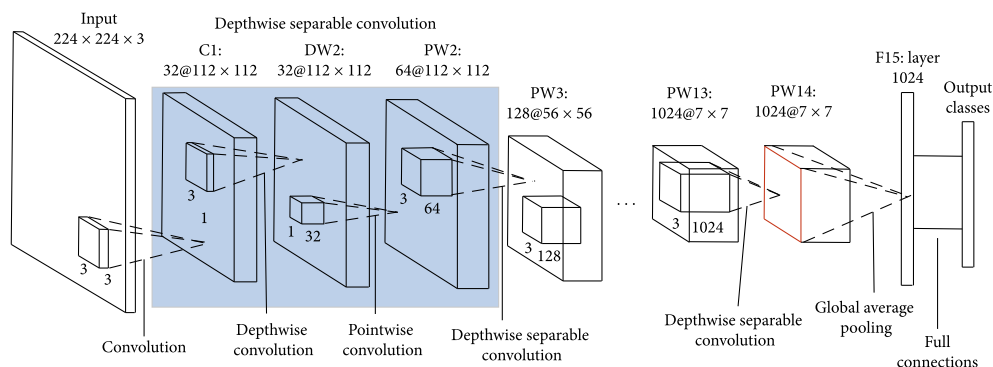


Figure 14: MobileNet Architecture

Stage 4: Chatbot extended with cloud-based multi-lingual component

For the final stage of the chatbot project, I have created a cloud-based multi-lingual component that has been added on top of the developments made in the previous stages. In this stage, I have used the NLP services provided by Microsoft Azure cloud. The chatbot is now able to support up to 70+ languages and with the use of these languages' different parts of the previous implementation can be run. However, during the implementation of the NLP services using the Microsoft Azure cloud, there were some limitations that have been discussed later in this document.

I developed this stage further by adding two more additional features that are the text to speech and sentimental analysis of text components. This has been added to develop the chatbot so people can listen to the response of the chatbot and read the response. The sentimental analysis component of this stage is a key feature that has been added to allow users to identify articles that have a positive sentiment or negative sentiment. This can be very helpful specifically for my domain as information about stocks help users make trades. Allowing users to understand the sentiment behind information can help users predict if the price of the stock will go up or down. I have also developed a URL brand detector that detects a specific brand in an image and draws an outline over the logo of the brand. These components have all been developed using the Microsoft Azure cloud and the cognitive services resource.

Added System requirements

Must:

- The system must allow the user to communicate with the chatbot in 70+ languages.
- System must use the Microsoft Azure cloud.
- System must detect the language entered into the chatbot.
- System must translate the detected language into English.
- System must find answer from AIML or CSV file.
- System must translate the appropriate answer from English to language which was inputted into the chatbot.

Should:

- The system should allow the user to retrieve answers from previous stages into the language they were originally inputted into the chatbot.
- System should enter include 10 new conversational pairs in at least two new languages.
- System should be able to perform sentimental analysis on articles stored in a folder.

Could:

- System could implement a text to speech
- System should be able to detect brands.
- System should be able to outline logos in images.

Added System Goals

- Ability to detect the language the user enters in the chatbot.
- Ability to translate a message inputted into the system to English.
- Ability to access the AIML/CSV file developed in stage 1.
- Ability to translate these responses back into the original language which was entered.
- Ability to return an appropriate answer to the user in the language it was sent in.
- Ability to use text to speech.
- Ability to review articles using sentimental analysis.
- Ability to detect brands using Microsoft Azure cloud.
- 10 new conversational pairs are added to the AIML file and few more pairs are added to the CSV file.

Cloud-based Multi-lingual component

In this stage, the chatbot has added the ability to automatically translate messages entered into the chat into English, process the answer for the original questions/statement, and then output an answer translated back into the original language that was entered into the bot. For this component, I have used the Microsoft Azure cloud where I have created a new resource which can be seen in the image below. The Cognitive Services resource is a product created by Microsoft that allows users to access multiple services using a single API key. This service includes multiple services which can help implement machine learning techniques such as Computer Vision, Natural Language Processing, Searching and Speech.

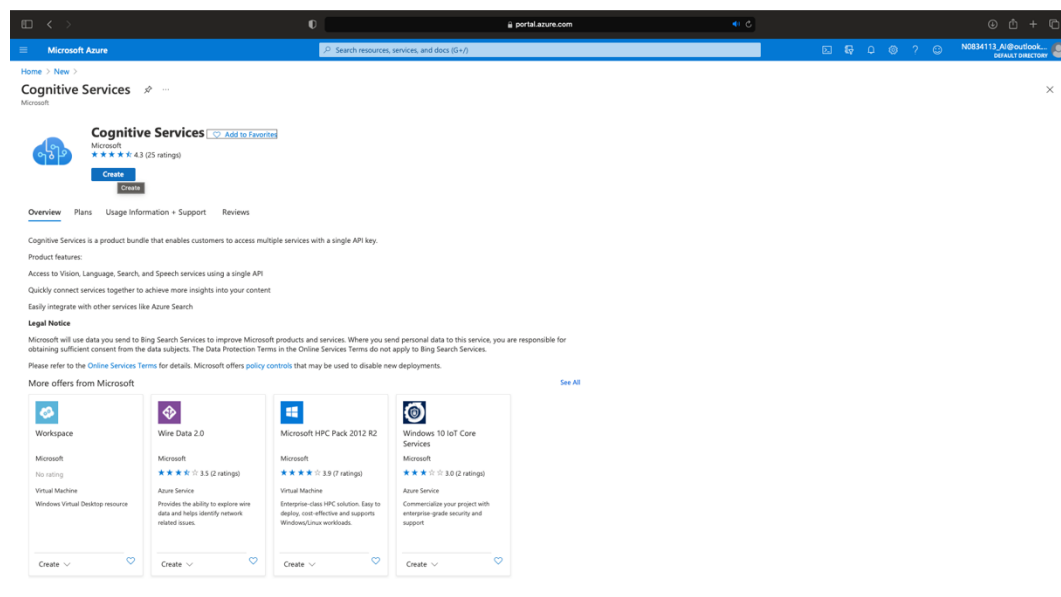


Figure 15: Creating a Cognitive Services Resource

Figure 16: Creation of Cognitive Services

After creating the cognitive services, I aim to use Natural Language Processing (NLP) which deals with written and spoken languages. The Microsoft Azure cloud consists of the cognitive services which will use the Text Analytics service which has the ability to identify key phrases and classify text sentiments of text. This Text Analytics service can be seen later in the document when analysing the sentiment of articles. For the implementation of identifying the language of a specific message from the user and translating this to a given language, it is necessary to use the cognitive service API key, endpoint, and region.

Detection of language

Firstly, it is important to detect the language of the message which is being entered into the chatbot this can be done using the azure cognitive services. The Text Analytics package uses the TextAnalytics client and the Cognitive services credentials import is required. The identification of specific language requires the document which contains the id of a statement and the text to be in a JSON format. The text is extracted and the `text_analytics_client.detect_language` method is used which generates the language of the text, language code, and the score out of 1. The pseudocode below shows the logical steps taken to perform the detection of a language.

Pseudocode for the detection of language

```

DetectLanguage(userInput)
    TextL = []
    Id = "user"
    Texts = {id:id, text:userInput}
    Texts.append(Text)
    Lang_analysis = text_analytics_client.detectLanguage(texts)
    Loop Texts:
        Texts[text_num][id]
        Lang = language_analysis.docs[text].detected_languages[0]
    Return langCode

```

Translation of language

Secondly, after detecting the language the language code is returned which can be used in the translation of the text. Text can be translated using the cognitive Microsoft translator API which again uses the endpoint, API key, and region as parameters. The user input is taken in as a parameter along with the language code from and the language code to. In every case, the language lang_to will be English when the chatbot processes the answer for the question/message the user enters. This can be seen by the pseudocode presented below.

```
Translate_Text(region, key, userInput, lang_to, lang_from)
    Path = Microsoft azure api translator path
    Parameters = (lang_from, lang_to)
    Body = text translated
    Response = request.json()

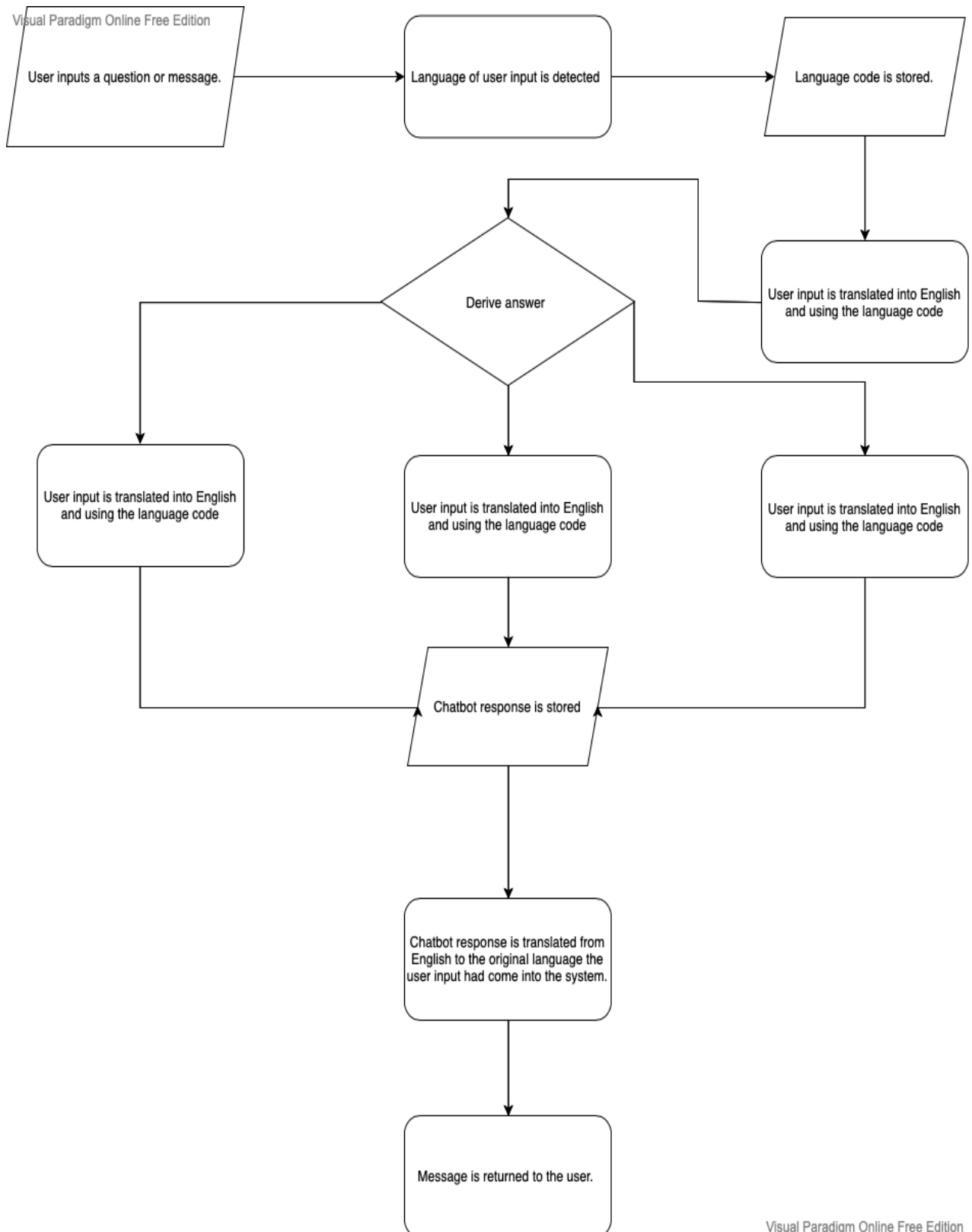
    Return Response
```

Once the response is generated into English the questions/statement can be checked in the AIML/CSV file which then can reply with a suitable answer. If no answer is found then a message is generated by the chatbot. Once the appropriate answer is derived either by using the AIML, CSV, or error message the text is required to be translated back to the language the English response was generated from. Here we still have the code variable storing the language code of the detected language. We use this language code again except this time the parameter lang_from is “en” and the lang_to is the language code of the original detected language. The design diagram below represents the logic used to derive this functionality.

Limitations

However, there are some limitations to the implementation. When translating sentences from one language to another, the order of words can be changed. Below I have given an example of a translation that has caused problems for the overall component. Let's take an example firstly I am translating from Hindi to English. I would like to ask my chatbot “what the price of goog” is in Hindi which is represented by the following क्या है गुग की कीमत. However, the translation of this from Hindi to English is “what is the price of gug” which does not correlate to my conversational pair located in the AIML file. Hence causing an error in my program. Another example of this can be seen by the following Hindi sentence मुझे पता है कि गूगल एक तकनीक है. This should mean “I know that Google is a technology” in English however when using the Microsoft azure translator and translating this sentence from Hindi to English the actual translation is “I know that Google is a tech” which is an incorrect translation. To solve this problem, I can just add the literal translations into the AIML or CSV file.

Translation design diagram



Additional Features

For Stage 4 of I have developed three additional features all using the Microsoft Azure cloud. These three additions help develop my chatbot with additional cloud-based implementation using the Cognitive services resource.

Text to speech

The text to speech component is very simple and uses the cognitive speech services which takes in the user's input and the Microsoft azure API key and region are taken in as parameters for the SpeechConfig method. The AudioOutputConfig class is used and the default speaker is set to true. The text is then transcribed to speech using the speech_synthesiser and finally, the output audio file is played. I have created a text-to-speech function that takes the user input as a parameter and plays the speech when the function comes to an end.

Brand detection

I have also added a brand detection component as an additional feature using the cognitive services computer vision service. This feature allows the user to enter "detect brand" into the chatbot. There are predefined URL links in the code that will open a specific image and detect the brand in the image by drawing a logo on the image. For this implementation, I have used various libraries such as OpenCV, NumPy, and requests. I have used the computer vision client image analysis method which takes in the URL links as a parameter and a features parameter which in this case is 'brands'. This fits in very well with the work I had done in stage 2 when I created a convolutional neural network that was able to classify brands. I have gone a step further using the cognitive services by drawing a rectangle over the logo in the image. Using OpenCV I have read the image URL and defined the colour and thickness of the annotation and created a window that will open for 10 seconds to display the image and annotation. Along with this the chatbot classifies the image and gives a percentage accuracy.

Sentimental Analysis

Finally, I have also implemented a sentimental analysis component to stage 4 which analyses different review articles regarding their sentiment and extracts key phrases from the text. Using a JSON format the id and text of the article are separated into the format; using the TextAnalyticsClient this implementation can extract key phrases and derive the sentiment of the article in a range from 0-1. 0 being a negative sentiment and 1 being a positive sentiment. For my domain, a sentimental analysis feature is key as users can check the sentiment of new news articles and decide whether they want to buy or sell a stock. If sentiment is positive then most likely users will buy a stock and if it is negative then they will either stay away from buying the stock or sell. There is further research into how sentimental analysis is an indicator of the rise or fall of a stock price. In the future, I would like to do some data analysis to find a trend between sentimental analysis and the rise or fall of stock values.

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].
- Yamashita, R. N. M. D. R., 2018. Convolutional neural networks: an overview and application in radiology.. *Insights into Imaging*, Issue 9, p. 611–629.
- 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 2020/21 Full Year*. [online] Available at: <https://now.ntu.ac.uk/d21/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].
- Heidenreich, H. (2018). *An Introduction into Knowledge-Based Agents - DataDrivenInvestor*. [online] Medium. Available at: <https://medium.datadriveninvestor.com/an-introduction-into-knowledge-based-agents-4e5c4ea0e8a> [Accessed 23 Feb. 2021].
- Wang, W., Li, Y., Zou, T., Wang, X., You, J. and Luo, Y. (2020). A Novel Image Classification Approach via Dense-MobileNet Models. *Mobile Information Systems*, [online] 2020, pp.1–8. Available at: <https://www.hindawi.com/journals/misy/2020/7602384/> [Accessed 23 Feb. 2021].
- Nltk.org. (2019). *10. Analyzing the Meaning of Sentences*. [online] Available at: <https://www.nltk.org/book/ch10.html> [Accessed 23 Feb. 2021].