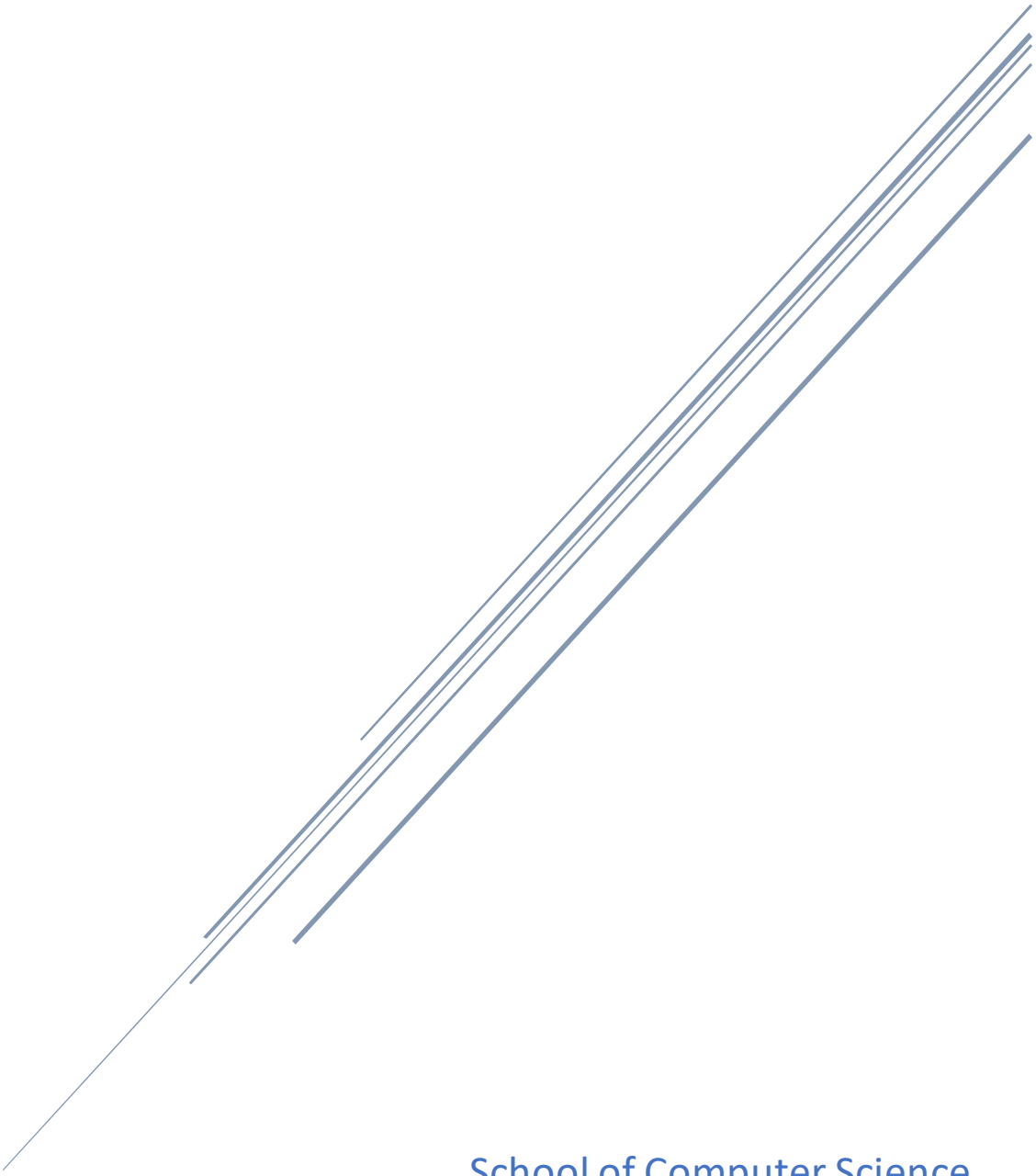
CHATBOT REPORT



School of Computer Science

COMP3074

Introduction

I built a modest bot which aims to perfect the functionality of the implementations added. The bot is capable of intent matching between multiple intents. I implemented these features as it was sufficient for the Chatbot to be interactive and engaging.

Background

Chennai Chatbot[1]

The academic report discusses the designing of a NLP Chatbot which utilised the Logistic Regression Classification algorithm. It inspired my system's structure and use of Logistic Regression; also the speech recognition input was a solid feature in their bot along with multi-layered similarity-matching.

Kuki AI Chatbot[2]

'Kuki is the world's most popular English language social chatbot', it's a bot designed for engagement. It inspired the use of an input cleaner which would only return a user's name, as well as my date / time and weather retrieval functions. It utilised a Wikipedia API for question answering and a rating system for the response accuracy, both features would have been good additions to my project.

[1] https://www.ijrte.org/wp-content/uploads/papers/v8i2S11/B12600982S1119.pdf
[2] https://chat.kuki.ai/chat
https://www.kuki.ai/research

Methodology

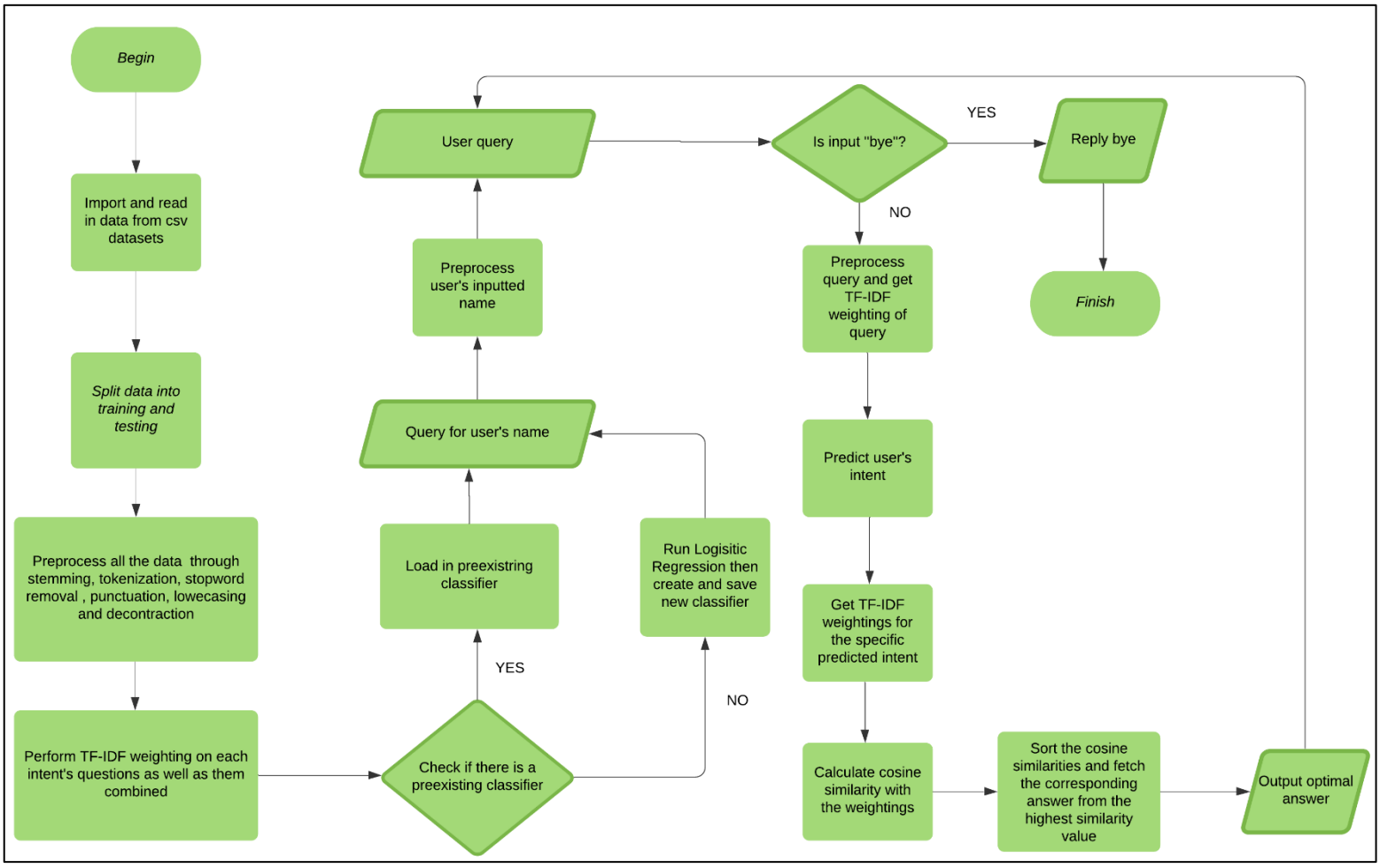


Figure 1: Flowchart showcasing how the system functions

1. Data Importation

When the bot is first run all datasets are read in, the question answering dataset uses both the CW and SQuAD[3] dataset, the small talk dataset uses the Microsoft's Chit-chat[4] dataset. Two intent sets are fed into two dictionaries, one is a dictionary of dictionaries where each dictionary is an intent, holding the questions and answers for the corresponding intent. The 'key' represents the question and the 'value' links to the response. This dictionary is used to get the optimal response for the user. The other dictionary is a dictionary of lists, each list is a unique intent, storing the questions from the corresponding intent. This dictionary is split into training and testing data to be used by the classifier which predicts intent. Finally, I import a name dataset which contains a large set of common names used for name processing.

[3] https://www.kaggle.com/ananthu017/squad-csv-format

[4] https://github.com/microsoft/botframework-cli/blob/main/packages/qnamaker/docs/chit-chat-dataset.md

2. TF-IDF Weighting

I run both intent question lists through a Count Vectorizer which contains an analyser. The analyser performs pre-processing: lowercasing, removing punctuation and contractions, stemming and for question and answering, removing stop words. The Count Vectorizer, will fit to the list's vocabulary and return the counts for the amount of terms in the list as a term-document matrix.

Next, I apply a TF-IDF Transformer to the term-document matrix. This will transform the matrix into a normalised TF-IDF matrix. This will reduce the impact of common tokens as uncommon tokens are more empirically informative. Now, I will have two TF-IDF matrixes based on each intent's questions, these will be used for Cosine Similarity matching. I also create a separate TF-IDF matrix which contains weighting for my feature training data which is a list of questions from both intents. This matrix will be used to train the classifier to predict intent.

3. Run Classifier

If the bot is running for the first time, I will pass in my feature training data TF-IDF matrix through my classifier with the training labels. Training the classifier makes sure the algorithm is learning generalisable knowledge. After being trained the classifier is saved, allowing the model to be reused.

4. Name Processing

Now the bot can run, the user can enter their name. The input is pre-processed: lowercasing, removing punctuation and contractions, lemmatising, and removing stop words. It's passed through a function which removes all redundant words surrounding the user's name. It first removes common words used. Then will run the cleaned input through the name list and if there's a match it will return the name, else it will return the cleaned input.

5. Querying

The user is now able to query the bot, if the user wants to exit, they can type 'bye' which breaks the query loop.

Intent Matching Complexity

There are multiple levels of complexity in my intent matching, not all the intents are predicted by the classifier, therefore I use 'simple text matching' for the date / time and weather features. While for small talk and question answering, I use a 'machine learning-based approach' using Logistic Regression to predict intent. Identity Management comes under the small talk dataset.

a) Text-based Matching

i. Get Date / Time

If the user input matches a phrase from the string list relating to the date / time feature, the bot will fetch the current date time using the 'datetime' library and output it.

ii. Get Weather

If the user input matches a phrase from the string list relating to the weather feature, a weather function is called which will ask the user for what city they live in, then output the temperature and weather stats at that location through the 'python_weather' library.

b) ML-based Matching

The user's query undergoes TF-IDF weighting to form a TF-IDF matrix, the weighting utilises the same Count Vectorizer and Transformer as the feature training data matrix created. This is passed into the classifier which outputs a predicted label stating the intent.

i. Small Talk and Question Answering

Depending on the predicted intent, two TF-IDF arrays will be needed. The first array will be formed based on the user's query and will utilise the same Count Vectorizer and Transformer as the corresponding intent's question matrix created. The other will be the corresponding intent's question matrix converted to an array. These arrays are used to perform Cosine Similarity.

Cosine-Similarity

I run a loop where each question in the TF-IDF intent question array has it's similarity checked against the TF-IDF query array. These values are saved to a dictionary if the similarity is above a threshold. The cosine number is saved as a value where the 'key' is the intent question. After every index has been iterated through, the dictionary is sorted from highest to lowest. The dictionary of dictionaries is used to get corresponding optimal responses for the user based on the highest value in the dictionary. If there are multiple responses for one query, a random output will be printed.

ii. Get / Set Name

If the highest dictionary value is equivalent to a getter / setter response within the small talk dataset, the bot will either print the user's current name or request a new one.

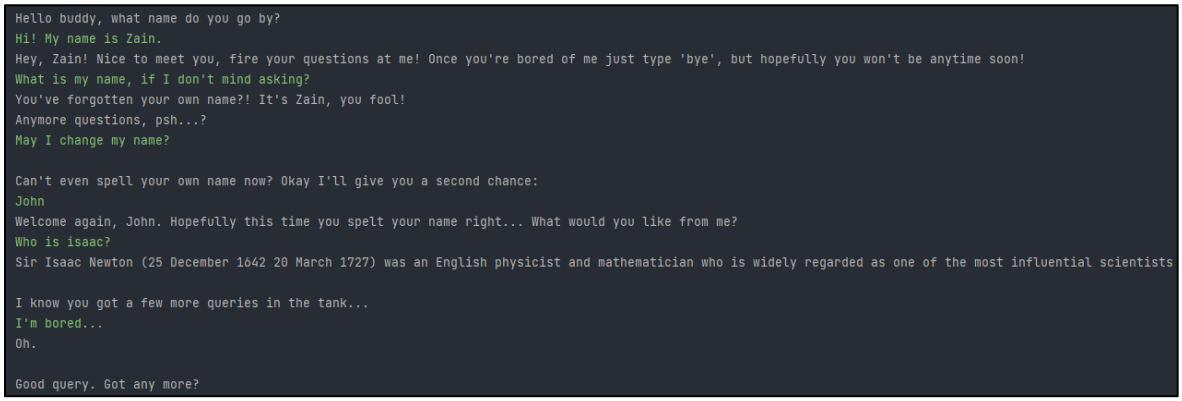


Figure 2: Example user interaction with bot

Zain Rashid – PSYZR2 – 20172929 – WORD COUNT (1992)

Benchmarking

Small Talk

Testing Query	Matching Query	Cosine Similarity	Potential Outputs
Hello, how are you doing?	Hello	0.7321365	Hey.
	How are you doing?	0.68115795	Living the dream.
whats your name?	What's your name?	0.9367621	What's in a name? Not much, apparently, because I don't have one.
whats you're name?	Whats my name?	0.75467605	You've forgotten your own name?! It's Zain, you fool!
	What's your name?	0.73717741	What's in a name? Not much, apparently, because I don't have one.
What's your nmae?	What's your name?	0.71501451	

Question Answering

Testing Query	Matching Query	Cosine Similarity	Outputs
rocket engine	Who engineered you?	0.8667075	People created me. But not the way people created you.
	I'm an engineer	0.7980253	Okay.
how does a rocket engine function?	how a rocket engine works	0.70887661	A rocket engine, or simply "rocket", is a jet engine that uses only stored propellant mass for forming its high-speed propulsive jet .
	What is the function of a "beer engine"?	0.57759007	to dispense beer from a cask or container in a pub's basement or cellar
who is isac nwton?	Who is the city Manager?	0.41207107	Jim Couch
isaac newton			I'm afraid I haven't understood you exactly, maybe you would like to ask another query?

Identity Management

Testing Query	Matching Query	Cosine Similarity	Outputs
whats name?	What was Plymouth's original name?	0.46352948	Sutton
	What was Riba's first name?	0.45433472	Institute of British Architects in London
	Who named the city Boston?	0.44931574	Puritan colonists from England
can you please tell my name, thank you!	Tell me my name	0.62042874	It's Zain, you fool!
	Name please	0.60966528	What's in a name? Not much, apparently, because I don't have one.
	Can you tell me your name?	0.60913011	
tweak name	Tweak my name	0.93078143	Can't even spell your own name now?
Hello! My name is Zain. Nice to meet you, hope you had a great day!	Zain	----------	Welcome again, Zain.
Hi nice to meet you! My name is xenliqo, thanks for asking!	Xenliqo	----------	Welcome again, Xenliqo.

Accuracy Results

```
Accuracy score:  0.9850662158354466
Question Answer F1 Score:  0.9856834143706104
Small Talk F1 Score:  0.9843934040047113
Question Answer Precision score:  0.9764516992239765
Small Talk Precision score:  0.9946444510562332
Question Answer Recall score:  0.9950913553313335
Small Talk Recall score:  0.9743515010201107
Confusion Matrix:
 [[3649   18]
 [  88 3343]]
```

Discussion

Results Evaluation

Benchmarking

I can decipher that for most test queries I get good outputs. However, the bot still has flaws. When inputting the phrase 'Hello, how are you doing?'. Although the focus of the input is asking 'how are you doing', the bot assigns the query to 'Hello'. This seems to be as both 'Hello' and 'how are you doing' are queries found in the small talk dataset. So when they're combined in a sentence, the bot only matches the query to one of the two. To fix this the Cosine Similarity threshold can be increased, although this could cause less successful outputs as the bot will have a harder time matching.

Another flaw would be when 'rocket engine' and 'isaac newton' were intent matched as small talk although they were actually question answer queries. I believe this to be the case because of the dataset size as when I decreased my small talk size, I got better results for question answering. Even though a greater dataset size should theoretically lead to higher accuracy models. This is proven as I get higher accuracy, f1 and precision/ recall scores when using larger datasets. Which in turns means there could be an issue with train / test data split or the classification model itself.

Finally, broken English or spelling mistakes can lead to decreases in accuracy from the model. Depending on the misspelled word a query can become near unmatchable, this can be seen with 'who is isac nwton?' which gave an unrelated query 'Who is the city Manager?' which only shared the 'who is' string with the test query. This could be as what defines a question for question answering is usually very unique and exact. So if the keyword is slightly misspelt it can completely alter the output. This could be remedied through the use of a spell checker although this still wouldn't solve the issue of misspelt names. In addition, spell checkers can unintentionally change correct spellings to incorrect ones, as well as correct a spelling to the wrong word.

In terms of user interactiveness the bot falls short due to being reply-based. This means it doesn't 'converse' with the user as the bot only questions the user for their name, so it only relies on replying to user queries, instead of being more interactive.

Accuracy Results / Optimisation

All accuracy functions gave solid results, showing a high matching rate. However, I believe the results to not be fully true as although they classify the intent correctly often, it could be due to factors like over-fitting.

The size of the datasets effected accuracy greatly, as variance between the intent sets would cause a big difference in the accuracy of matching per intent. Sets similar in size gave equal weighting of accuracy, giving more reliable results. In addition, larger sets gave better accuracy, so since I used large datasets of similar size, I would be maximising my accuracy output.

The training / testing data split and Random State value were set to 0.35 and 11 respectively. These values gave high accuracy scores; however, these values may not perform as well for other datasets.

To break up words I had to decide between a lemmatiser and stemmer, lemmatising is more accurate it but takes long to compute and resulted in lower accuracy scores. This could be due to the simplicity of the chatbot and how reducing the tokens could lead to more ambiguity within queries. I decided on using the Snowball Stemmer as it's more reliable than the Lancaster Stemmer, while also being an updated model of the Porter Stemmer.

Impact

The bot could have a positive impact if an individual seeks companionship, the bot utilises small talk, so can interact and chat with the user. Additionally, the bot is trained on a 'witty' dataset meaning interactions can be interactive. The bot can share helplines in case a user may feel lost in life. Furthermore, the question and answering allows a user to increase their general knowledge if they interact and ask questions. But, if a user is given an incorrect response to a query, they may believe it's true. This could be bad as they may spread the information throughout their social circles, especially if within a filter bubble.

Bias

The bot can be biased if the dataset doesn't contain a varied source of queries. As it could lead to the user getting similar outputs for any of their inputs. Furthermore, if datasets are split up there could be bias if queries are grouped together based on similarity instead of randomly.

Conclusion

Overall, I have made a balanced and optimised chatbot which near-perfects the implemented features. I thoroughly explained the bot's functionality and implementation, in addition to noting and suggesting fixes for any issues relating to the bot, giving my input on whether they are feasible or not. Performing thorough analysis on the system's behind the bot to test functionality. I also took inspiration from previous chat bots and have recognised important features which could improve on the current implementation I have. Finally, I understood the limitations and negative consequences of the bot as well as how it may benefit users.