NottBot

Yash Suresh,
School of Computer Science,
University of Nottingham,
Nottingham, NG8 1BB
psyys5@nottingham.ac.uk

Word Count = 2285

PART 1, INTRODUCTION

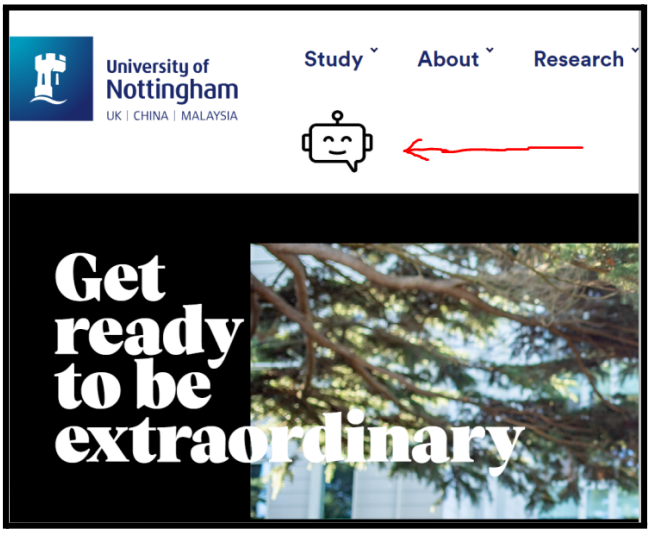
(1.1) What it is

Chatbots continue to be an integral part of businesses and organisations. There can be *so much information* available on say, a company's website that it can be *overwhelming* to users to break it down into meaningful chunks.

I have created **NottBot**, an AI based chatbot which will ***answer questions related to the University of Nottingham***.

Topics can range from *library opening times, to health services, to even the weather & time in Nottingham*, providing a one stop answer to any question without having to scour through the university's website for information.

(1.2) Suggested Use



If deployed to production, NottBot should be an instantly visible icon on the university's main page, so that it is instantly visible to all users of the page, providing ***easy and quick information***.

This will also be visible when accessed via mobile devices, so can be used to provide useful information, in times of emergency.

Clicking / tapping the chatbot will open up a chat room, and when the user types '*bye*', return to its '*icon*' state, which can be tapped / clicked again, repeating the process

.

(1.3) Motivation

ResearchGate revealed the existence of an idea proposed by professors in an Engineering University in India **[1]**. It suggests creating a chatbot (no name was given, let's call it 'CollegeBot') which answers questions regarding admissions, fees, departments etc, which can quickly provide students and faculty key information without having to search the website for it.
I would like to create something like this, providing UoN Students accurate advice and help. Benevolence, ultimately should be the goal of an AI based system like this.

(1.4) Tools

My chatbot is written entirely in **Python**; '**numpy', 'pandas' , 'scikit-learn' and 'nltk'.**

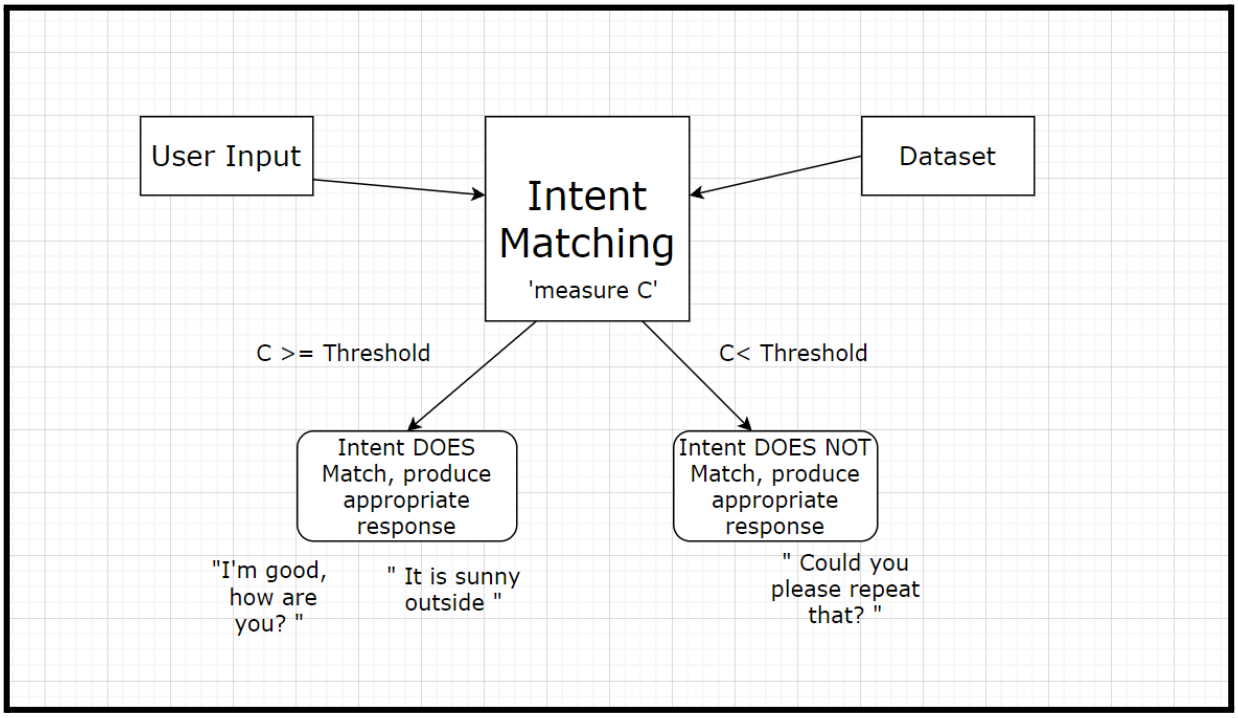
I have created the chatbot on VS Code, and hence should ideally be run on the same IDE. Jupyter was used for *debugging, and testing*.

PART 2, UNDER THE HOOD

(2.1) Basic Workflow

As the diagram below illustrates, it is very much a matter of matching the **similarity** of a query to the correct piece of text in the dataset, which can produce an appropriate response.

Since the interpreter cannot understand text as intrinsic values, we represent the *input* and *dataset* as numbers in a vector space, and use a complex mathematical (explained further below) function to obtain their '**Cosine Similarity**', which is a metric of how similar they are.



2.a Cosine similarity is represented by 'C'

If the value of C *equals or exceeds a certain threshold* (which we set after empirical testing), an appropriate response is fetched from the dataset and returned.
If not, the user is prompted to re-enter the questions.

(2.2) The AI powered Engine behind NottBott

The formula for finding the Cosine Similarity is the core of our engine.

TF-IDF (Term Frequency-Inverse Document Frequency), the formula born after combining the concepts of **Term Frequency** and **Inverse Document Frequency**, by the following reasoning:

- A token/word which appears frequently in a line of text is *significant to that text*.
- **But**, the significance of a word/token *decreases the greater its appearance across different pieces* of text, meaning it is relevant to *either all documents (lines of text) or none* **[2]**

$$tfidf(w,d,D) = tf(w,d) * idf(w,D)$$

2.b

Let's take for instance, "How are you?". We compare this string with existing datasets in our code base.
After using sklearn's TD-IDF Vectoriser, it splits the string into a an array of characters, ['how', 'are', 'you'] and returns a grid like matrix of TD-IDF weights of each character against a table of similar questions.

```
X = vectorizer.fit_transform(dataframe['patterns']).toarray()

print(X)
```

```
[[0.         0.         0.         0.         0.         0.
  0.         0.70710678 0.         0.         0.         0.
  0.         0.         0.         0.70710678 0.         0.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         1.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.        ]
 [0.68150194 0.         0.         0.         0.         0.
  0.         0.         0.58984883 0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.43316679 0.        ]
```

2.c

Using further manipulation of the pandas dataframe, we now have our xTrain (Dataset questions) and xTest Values (user input).

```
xTrain = pd.DataFrame(X, columns = vectorizer.get_feature_names_out())
xTest = vectorizer.transform([str]).toarray()
```

2.d

Pairwise distance is the distance between the arrays in vector space. *Greater the distance, less the similarity*. So the Cosine Similarity can be obtained by: ***1 - pairwise_distance(XTrain, XTest)***.

```
In [26]: cos = 1 - pairwise_distances(xTrain, xTest, metric = 'cosine')
         print(cos)
```

```
[[0.        ]
 [0.        ]
 [0.5355551 ]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.23235949]
 [0.23235949]
 [0.        ]
 [0.        ]
 [0.        ]
 [0.23235949]
 [0.4425554 ]
```

2.e

There we have it, the **Cosine Similarity** between our query and all questions in the dataset, represented by a 1-d array.

All responses are picked *on the basis of Cosine Similarity*.

(2.3) Datasets

Every good Machine Learning needs good training data. Due to the specific nature of NottBot, I have created the datasets on my own.

0	greetings
1	best_academic
2	best_phd_student
3	student_service
4	nott_clubs
5	nottbott
6	david_ross
7	best_indian_restaurants_notts
8	best_chinese_restaurants

0	hi hello hey good morning evening afternoon
1	best greatest lecturer professor teacher
2	best phd assistant
3	where student service find uni university
4	nightclubs night out clubs nottingham nightlif...
5	name your call you
6	sport facility gym football pitch
7	best highest rate tasty restaurant eat food in...
8	best highest rate tasty chinese food takeaway ...
9	when libary open closed opening hours study sp...
10	catch hopper 903 stop time timings hours frequ...
11	mynottingham app broken working
12	exams schedule timetable time day location

0	welcome to Notts, where the party bops!!\nHow ...
1	It has to be Monsieur Docteur Jeremie Clos
2	It has to be Monsieur Benerradi
3	Portland Building, in the main Campus
4	Nottingham has some good clubs: \nPryzm\nInk\n...
5	My name is NottBot, here to answer all your Uo...
6	We have the mighty David Ross
7	I would recommend Maharaja's Retreat
8	Hot 9, just outside Jubilee Campus, is the one...
9	All UoN libaries will be open 24 hours till 31...
10	Hopper buses run Mon-Fri every 15 mins.\nEvery...
11	Please speak to Student Services. Here's their...
12	Should be on your MyNottingham app
13	The answer is obvious: Computer Science!

2.e

Each of the responses are *specific to the questions asked*, and when matched by similarity, the algorithm will choose that particular response.

There are 2 datasets, one for **UoN QA**, and the other dedicated for **Small Talk**.

Both use JSON, but what differentiates them is the **preprocessing** that precedes.

(2.4) Pre-Processing

Similar to CollegeBot, Preprocessing here as been done in the following steps:

1. lower():
2. Remove punctuation using **regexTokeniser**
3. Removal of stopwords
4. Stem the words

The patterns in my UoN-QA dataset have been *manually preprocessed*. I have only used keywords for each question; the kind of unique keywords which have a very high chance of matching similarity with the user question. For instance, a question relating to libraries across campus will look like the following:

[‘library opening close djanogly hallward hours’]

I have deliberately removed words like ‘**university time campus**’ as these are words likely to appear in other questions being related to the university, increasing Cosine Similarity with other dataset questions*, risking a false positive Similarity Match.*

I have created the dataset this way as it makes it *computationally quicker* to preprocess *just the test data* rather than the training data as well, and have carefully chosen words which have a very high chance of appearing in queries.

For **small-talk, weather and name**, I chose NOT to remove stopwords. These topics use words which are common to everyday language, so it is not feasible to sieve through questions when most of the words have been removed.

To ensure accurate similarity matching in small-talk, the threshold value has been set high, so that only a question which is very similar will match.

(2.5) Threshold Testing

To determine an appropriate similarity threshold for each feature of the chatbot, let's test the Cosine Similarity values empirically. Using Jupyter Notebook, I've created some dummy queries to be tested against the datasets.

Small Talk:

userInput	**Closest 'pattern' in smalltalk.json**	**Cosine Similarity**
'How are you?'	'how are you'	1.0
'How you been?'	'how have you been'	0.826
'Thank you''	'thank you'	1.0
'Thanks a lot!!'	'thanks'	1.0

The similarity levels are very high, with a few perfect matches.
0.7 has been chosen as the threshold.

Name Change/Weather

There are really just a few common phrases which indicate the user's desire to change names, so we need to choose a high threshold.

userInput	**Closest 'pattern' in intents.json**	**Cosine Similarity**
'I want to change my name'	'change name please amend wrong'	0.749
'Please change my name'		0.801
'What is the weather outside in Nottingham?''	'What weather conditions temperature outside'	0.647
'My name is wrong. Change it please!!'		0.906

There is high similarity here as well. To maintain that, *Stopwords have not been removed*.
A value of 0.6 has been chosen as the threshold, for 'name' and 'weather'

Question Answering

This is where we need to be *more flexible*. As there are many different possible questions, we can't always expect a high match. Not to mention, stopwords are being removed.

userInput	**Closest 'response' in intents.json**	**Cosine Similarity**
I want a doctor'	'We have a range of health services..'	0.212
'Best phd student?'	"Of course Monsieur Benneradi" (or the one marking this)🙂	0.545
'Best chinese takeaway?''	'Spice House.. Near Jubilee Campus'	0.518

A value of 0.15 has been chosen as the threshold.

(2.6) Features

To summarise, here are all the features implemented:

Question Answering: Provides quick UoN information based on commonly asked questions.

Name Recognition, Name Change: NottBot displays and changes the username every time it's their turn to speak, giving an old-school chat room feel.

Small Talk: NottBot detects everyday phrases to respond in a single-turn style.

Weather + Time: A quick, easy way to find out the local weather and time in Nottingham.

Smooth UX: By rigorous testing, and the 'while' loop which powers the chatbot, the user will always be returned a response, even if the chatbot needs to **escalate** and **prompt.**

PART 3, Evaluation:

(3.1) User Testing

We tested the individual features previously to set similarity threshold values.

As for **User Testing**, we can manually measure the **Accuracy** by calculating **correctness of responses against expected responses.**

Lets group questions into complexity levels:

Easy: The similarity should easily be detected by the system.

Medium: the question's wording has a higher chance of similarity mismatch or not passing the threshold.

Hard: Convoluted wording; the question can really be expected to be understood only by a live human.

Question	Expected Response	Actual Response	Pass/Fail
How are you?	'All good'	'All Good'	✅
So, tell me how you are..	'All good'	'All Good'	✅
Please inform me of your condition	'All good'	"My name is NottBott, here to answer all your UoN questions"	❌ Intent Mismatch
Who is Head of the School of Geography?	*head of school name	*Sorry, please repeat"	❌ *not in dataset
Inform me of the weather conditions outside	*weather and local time	*weather and local time	✅
Kind Sir, please tell me the humidity in Nottingham	*understands the question is about weather, Still returns weather and time	"Sorry, please repeat"	❌
Best lecturer at Nottingham?	*Dr Jeremy Clos	Dr Jeremy Clos	✅
I'm looking for Dr Jeremie Clos in the whole campus, would you know where to find him.?	*Department of Computer Science	"Sorry, please repeat"	❌

For easy questions: the accuracy rate, 2/3 **= 67%**

For medium questions: the accuracy rate, 2/2 **= 100%**
For hard questions: the accuracy rate, 0/3 **= 0%**
Overall, (2+2+0)/(3+2+3) **=** 4/8 **= 50%**

Naturally**, testing will be biassed** as the creator of NottBot is also testing it, so will tend to use questions which are covered in the dataset.
My friend (who doesn't study computer science) tested it. Results of his questions were:

For easy questions: the accuracy rate : 1/3 **= 33%**
For medium questions: the accuracy rate : 1/4 **= 25%**
For hard questions: the accuracy rate : 0/4 **= 0%**
Overall = 18.2%

NottBot wouldn't pass the Turing Test. It simply cannot understand the context of the question like a human does, if the wording doesn't overlap with the dataset, even if the object of the question is the same and has been covered by easier questions.
NottBot, like other chatbots, is **very dataset and language dependant**

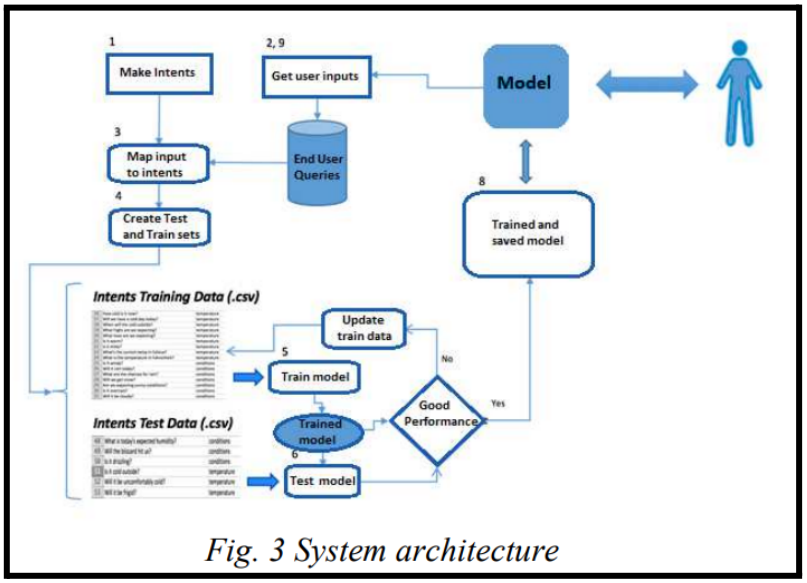
When asked to describe his UX with NottBot in 3 words, my friend said, '**Smooth, friendly, but limited.'**

(3.2) Comparison with 'College Enquiry Bot'

Compared to the CollegeBot proposal we met before, here is how NottBot compares

ChatBot	Uses TensorFlow	Uses NLP for similarity/intent matching	Cosine Similarity	Stemming	Lemmatisation	JSON dataset file
College Bot	✔	✔	✘	✘	✔	✔
NottBot	✘	✔	✔	✔	✘	✔

ChatBot	Deep Learning	Aesthetic UI	Fully Fledged Application
College Bot	✔	✔	✔
NottBot	✘	✘	✘



3a. College Enquiry Bot has a somewhat similar architecture to NottBott

(3.3) Future Development

It is worth breaking down some of NottBot's issues, and possible measures.

Issue	Example	Type	Possible Measures
Accuracy. Test results above show NottBot is not very conversationally deep, and can be stumped by off-topic or convolutedly-worded questions.	***"Please tell me what to call you"*** will result in different answer from ***"What is your name?"*** Although their intent is the same	Technical	-> Change stemming to lemmatization in preprocessing. This will result in real English words, not shortened non-English words of stemming, increasing Cosine Similarity -> Increase the size of datasets, with a large number of possible questions.
Nature of datasets. If we use 'unique to a topic/question' words in datasets, we can soon run out of such words	The word 'professor' can be used for different professors across the university, and cannot be stored as separate questions for each professor	Technical	Possibly use Neural Networks for a more intelligent algorithm to figure out sequence of words rather than single words
Data Bias, (one of the 2 main forms of bias) **[3]** Being a UoN student, much of the datasets are generated by me.	One of my responses states a certain lecturer from Toulouse as being the best at UoN. This is an opinion of	Social/ Ethical	Reviewing and creating larger datasets can be done by multiple people from a wide background (**religions,**

My perceptions of the university's facilities, staff etc will very likely affect the neutrality of the dataset.	mine, and may not be shared by others (I'm sure that's not really the case🙂)		**ethnic groups, genders and interests**) to ensure not only factfulness and minimal bias, but also that only a certain type of questions don't dominate NottBot's functionality.
User Testing Bias	I will test the bot in a way I am familiar with. Even my friend (who shares interests with me) will probably ask similar questions.	Social/Ethical	To ensure it serves a wide variety of people with minimal bias, we need to get a **more diverse testing control group**.

PART 4, CONCLUSION:

Final Thoughts

'Smooth, friendly but limited'.

3 words which my friend uttered as he looked at the screen, after testing NottBott.

NottBott is a **smooth** talker. It will make you laugh. It will help you find a taxi after a night out. It will save you the hassle of asking people the whereabouts of the library.
It's well mannered, **friendly** and pleasant like all virtual assistants should be.

But it is not Human. Its functionality is only **limited** to its creator's own university experience, his intelligence, and his biases.

References

[1], VNR,VJIET, 2022
https://www.researchgate.net/publication/359166891_An_Intelligent_College_Enquiry_Bot_using_NLP_and_Deep_Learning_based_techniques

[2], Marius Borcan, 2020

https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

[3]
https://www.lexalytics.com/blog/bias-in-ai-machine-learning/

Appendix

When running the Python file, please '**cd**' into the NottBott folder or else the json files won't be linked.

Place the NottBot folder in your IDE (ideally VS Code) and run the '**main.py**' file. **This is the entry point for the program.**

Sample questions:

What is your name?
Best Indian takeaway in Notts?
Good chinese restaurant in Notts
Djanogly Library opening hours?
Yo!
What are some good nightclubs in Nottingham?
What sport facilities have you got?
Hopper bus times?