**Conversational Chatbot**

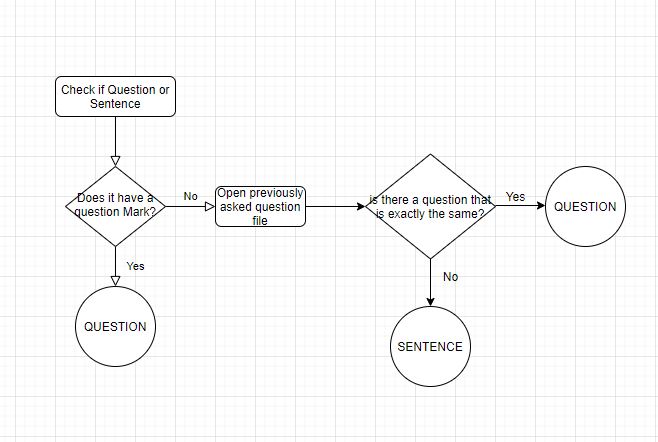
The conversational Chatbot is an algorithm that the user can talk to, as if it was a rea human. It will not be an “assistant”, like “Siri” or “Alexa”, but it will be a companion, for everyday conversations written on python.

In order to make this project successful, I had to start learning the basics of machine Learning, and the different models that are used in the real world.

For my project, I am going to use two of these a lot:

* Classification Model
* Regression Model

This is because, these models should be able to identify whatever the user says, and use it to give an exact answer. One use of the classification model is to identify whether the user input is a question, or a sentence.

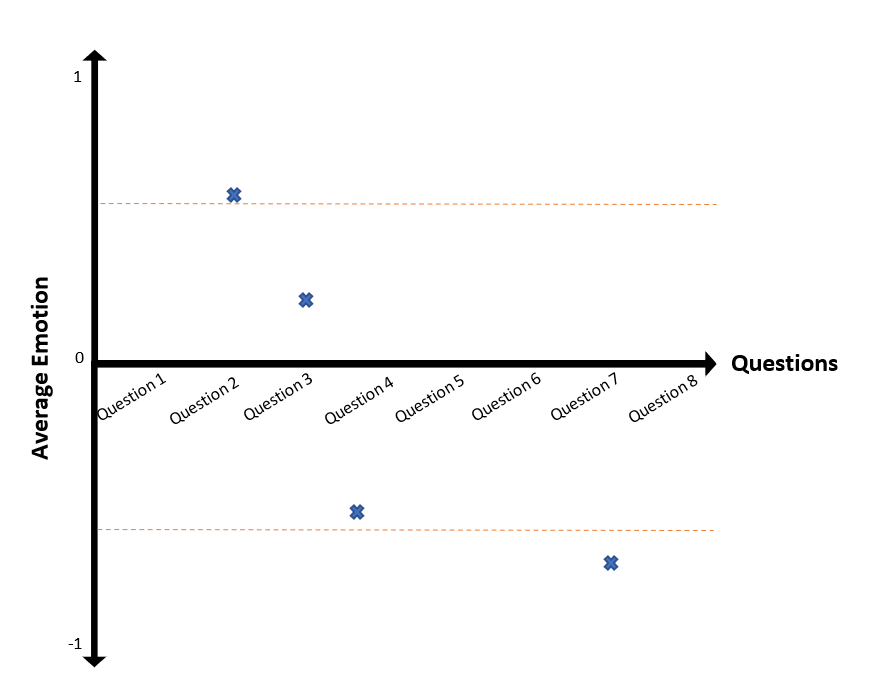


With a given input, by the user, the algorithm should be able to classify the input as being either a question, or a sentence (hence the classification model).

**Recognising user’s emotions:**  
Being a “virtual friend”, my chatbot should be able to roughly identify whether or not the user is sad or happy through the words they use. All of the sentences will be saved, and analysed word for word.

I will create a dictionary of the 1000 most used words in the English Language. An “emotion” value ranging between -1 and 1 will be given. And the program will create an average emotion value ranging between -1 and 1, which will roughly determine whether the user is saying something sad or not. To find the exact boundary values where we don’t consider a sentence being sad, but happy or vice versa, would be impossible. However, I am confident that, with enough trials, I will find the boundary values, which will have the least compromises.

Once the classification of the user’s emotion has been made, I can have different outcomes for the chatbot.

**Recognising the user’s questions**

The graph shows how the classification roughly will work. Once the algorithm has identified that what the user has written is in fact a question, it will try and find out what the user is asking. To do that, I will be using another csv file, which will contain a long list of questions that the user might ask the AI. The algorithm will find which question is the closest to the inputted question, and choose that one, and depending on the “average emotion value”, the chatbot will output an answer. This method will obviously not be able to take into account things like “rhetorical questions”, as it will assume that each question has to have an answer; this abstraction simplifies my code by so much, but still be able to retain the realism of talking to chatbot.

***Correcting the program***

Whenever there will be a question that the AI misunderstands (because it will happen), the user will have the option to correct the AI, and give an appropriate answer for the question they have just asked. The new question and answer will be added to the csv file containing all of the questions, which is part of the machine learning mechanism.

I will also try and find conversations between people (online), and try to automate this process, by adding the example questions in the file myself, so that the algorithm has more examples for its classification, to get better over time.

**Recognising user’s sentences**

Whenever the algorithm has decided that the user wrote a sentence, it will treat it as such.

First, it will open a csv file where different sentences with their appropriate responses are stored. And it will try and check if what the user has written is similar to any of these examples. And if it is, it will output the response.

However, it will occur many times that the sentence typed by the user is not similar to any of the examples. When that is the case, if the sentence is a really long sentence and it is using many words, the algorithm will assume the sentence as being a story/anecdote. And for this specific classification, the “emotion value” is really important, as the chatbot has to cheer up the user, or asking more about the story, if what they said represents something sad.

If it’s a shorter sentence, a shorter answer will be outputted such as “awesome” or “great”, or “I am so sorry”, depending on the “emotion value”.

**Creating questions**

There is two ways that the algorithm can create questions. The chatbot will randomly choose between the first and second option, however there will be a slightly bigger weight on the first option, as it is more secure.

1. First, the algorithm will open the csv file with all of the questions, and ask a question to the user that the user hasn’t previously asked the chatbot.
2. The algorithm will also be able to create questions on its own. By using a set of grammar rules, the chatbot will take a random verb, a random noun etc… and construct a random question. There will be different structures for these questions, meaning that there will be many types of questions the chatbot will create. Part of the machine learning process consists in correcting the chatbot: whenever the combination is not desirable, the user will be able to send feedback, so that the chatbot will never make that combination ever again.

Asking an unrelated, random question can be alright at times, but it is necessary that the chatbot asks questions related to the topic the user has been talking about. To do this, every word saved in the “dictionary” csv file will also have a value of “conversation”. If a word has such value, it means that, whenever the user uses it in that sentence, it would be quite easy for the chatbot to ask a question with that word in it.

**Creating sentences**

Similarly, to how I will create questions, sentences will also be created from scratch, given some grammar rules, or chosen from the file with all the sentences. However, all of the sentences the chatbot will say, should be all positive, so the algorithm will have to pick the right sentences, depending on the “average emotion” value each sentence retains.

**GUI of the program**

Being a chatbot, the user needs to feel as if they are chatting with someone, so the interface for the user is very important. A window will be created, and it will have the interface of a chatting app.

Once the user has typed and sent their message, they will not be able to send another one, until the chatbot responds. I am including this feature, as it massively simplifies the way the conversation flows, because human conversation is much more complex; it should be, however, sufficient enough to make the conversation realistic and immersive for the user.

Adding to that, features like the “three dots” to wait for the response of the chatbot, the ability to correct the algorithm if there’s any wrong combinations and the immersive experience, will all contribute to the realism of the program.

**Program Organisation**

The code will be all written in OOP, with inheritance being the most dominant in the program, but also retaining some level of aggregation as well. Every class will be saved on a different file, which will be then imported in the main program. This is to ensure that the program is encapsulated perfectly, as I am only allowed to import variable/function/class from the other files into the main program, and not vice versa, as collisions would occur during run-time.