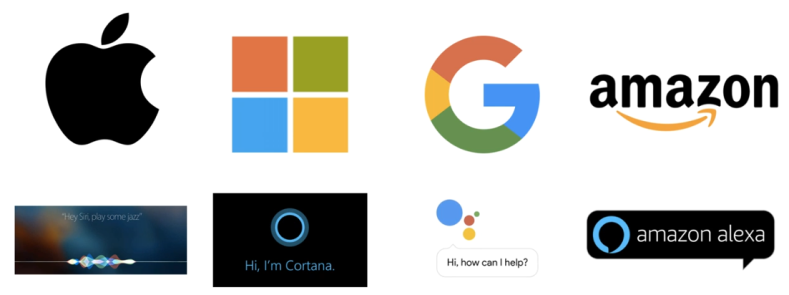
**EE769 Introduction to Machine Learning Course Project Report, EE Dept, IIT Bombay, June 2020 (Course Instructor: Prof. Amit Sethi)**

**Chatbot**

**Siddhesh Parab** (Roll No. 193074015) 193074015@iitb.ac.in

A **chatbot** is an artificial intelligence-powered piece of software in a device (Siri, Alexa, Google Assistant etc), application, website or other networks that try to gauge consumer’s needs and then assist them to perform a particular task like a commercial transaction, hotel booking, form submission etc . Today almost every company has a chatbot deployed to engage with the users.



**Introduction:**

**How does Chatbot work?**

There are broadly two variants of [chatbots](https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa" \t "_blank): **Rule-Based**and**Self-learning.**

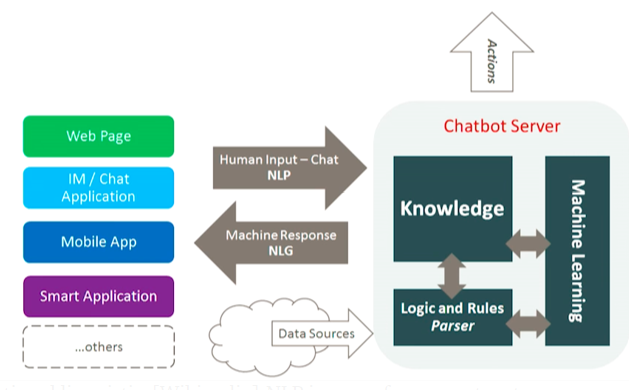
In a**Rule-based approach**, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.

Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be of further two types: **Retrieval Based**or **Generative**

i) In **retrieval-based models**, a chatbot uses some heuristic to select a response from a library of predefined responses. The chatbot uses the message and context of the conversation for selecting the best response from a predefined list of bot messages. The context can include a current position in the dialogue tree, all previous messages in the conversation, previously saved variables (e.g. username). Heuristics for selecting a response can be engineered in many different ways, from rule-based if-else conditional logic to machine learning classifiers.

ii) **Generative**bots can generate the answers and not always replies with one of the answers from a set of answers. This makes them more intelligent as they take word by word from the query and generates the answers.

**Block Diagram of Chatbot**



**Steps of Chatbot:**

1. **Data Preprocessing**
2. **Importing libraries**

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, a Python libraries do not pertain to any specific context in Python. Here, a 'library' loosely describes a collection of core modules.

1. **Corpus**

For our example, we will be using the Wikipedia page for [chatbots](https://en.wikipedia.org/wiki/Chatbot" \t "_blank) as our corpus. Copy the contents from the page and place it in a text file named ‘chatbot.txt’.

1. **Read data**

Read in the corpus.txt file and convert the entire corpus into a list of sentences and a list of words for further pre-processing.

1. **PreProcess**

A function called LemTokens will take as input the tokens and return normalized tokens.



1. **Building & Training Model**
2. **Keyword Matching**

Next, we shall define a function for a greeting by the bot i.e if a user’s input is a greeting, the bot shall return a greeting response.ELIZA uses a simple keyword matching for greetings. We will utilize the same concept here.

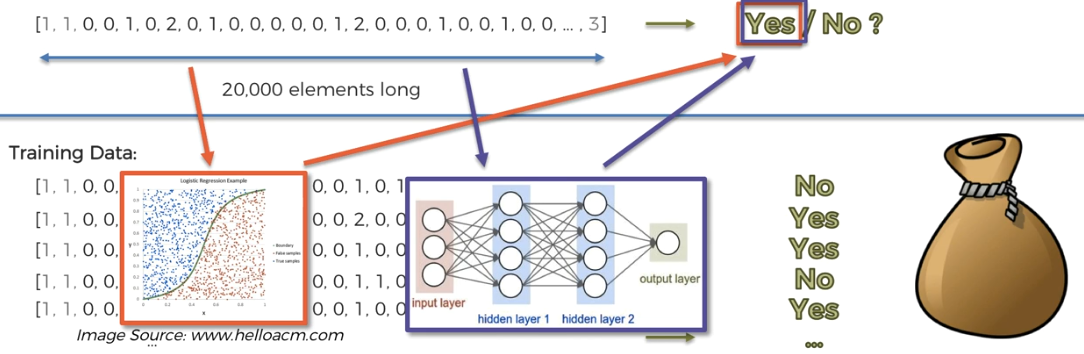
1. **Generating Response**

To generate a response from our bot for input questions, the concept of document similarity will be used. So we begin by importing the necessary modules.

From scikit learn library, import the [TFidf vectorizer](http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html" \t "_blank) to convert a collection of raw documents to a matrix of TF-IDF features.

Also, import[cosine similarity](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html) module from scikit learn library

This will be used to find the similarity between words entered by the user and the words in the corpus. This is the simplest possible implementation of a chatbot.



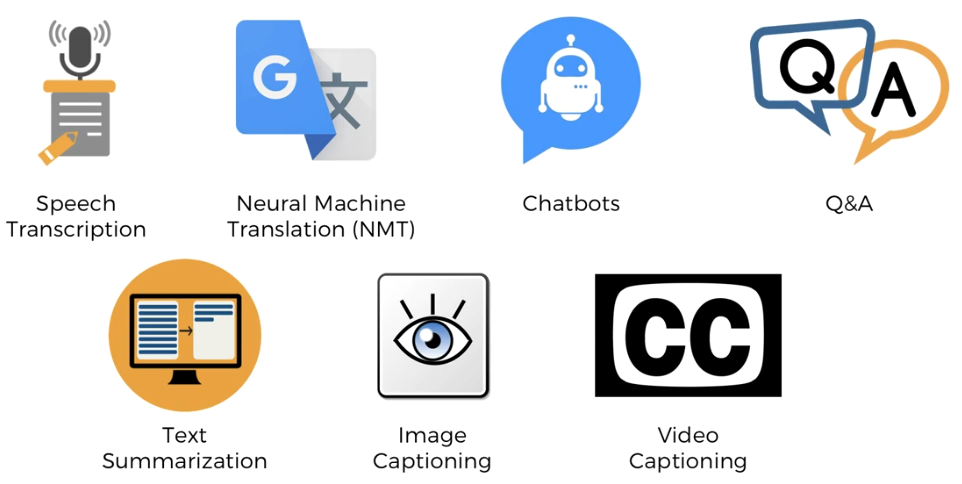
We define a function**response** which searches the user’s utterance for one or more known keywords and returns one of several possible responses. If it doesn’t find the input matching any of the keywords, it returns a response:” I am sorry! I don’t understand you”

1. **Testing Model**

Finally, we will feed the lines that we want our bot to say while starting and ending a conversation depending upon the user’s input.

**Applications:**

1. E-commerce bots: from ordering pizza to ordering shoes.
2. Handling minor daily task, for example Poncho is a great [local weather bot](http://poncho.is/) and source of early morning humour.
3. Content delivery – look at the Wall Street Journal and Tech Crunch!
4. Event reservation – restaurant reservations, doctor appointments, movie tickets etc.
5. Agenda/Scheduling – we already use to do lists, calendars and similar, but bots could do this for us as well.
6. Conference bots – they are like an extra organiser. I worked on bot that does something like this, he’s name is [Sava Bot.](http://savabot.com/)
7. Personalized helpers: bots that help you manage your health, your grocery list, your social life. Just about anything.
8. Travel bots – from saving you money on booking flights to helping you connect with locals and tourist guides.



**References:**

1. Wikipedia Chatbot Text
2. [shala2020.github.io/](https://shala2020.github.io/)
3. Automate the Boring Stuff with Python Programming by Udemy