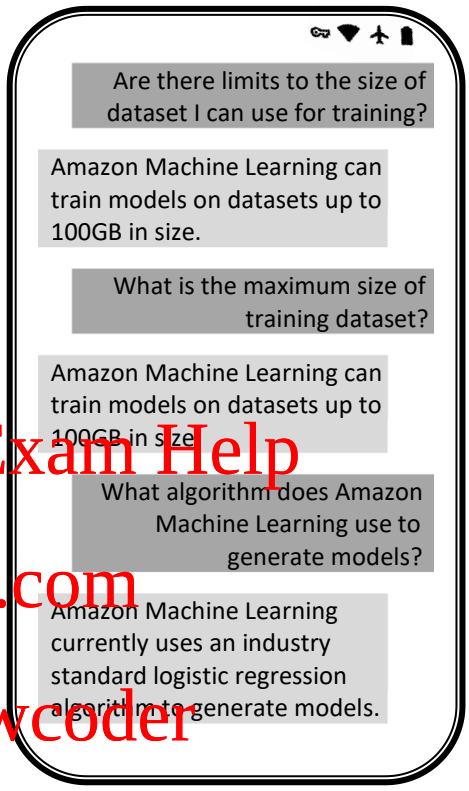
# COMP 10261: The *FAQ Bot*, Phase 0

Sam Scott, Mohawk College, January 2022

## Overview

An **FAQ Bot** answers questions about a particular topic. It is a conversational interface to a stock set of questions and answers.




When an FAQ Bot receives an **utterance**, it determines the user's **intent** by matching that utterance to one of its stored question and answer pairs. If it succeeds in determining intent in this way, it uses the answer as its **response**. In the example on the right (from Vajjala et al.'s *Practical Natural Language Processing*) the FAQ Bot has determined that the first two utterances have the same intent and has responded with the same text in both cases.

If an FAQ Bot fails to determine intent, it usually outputs a standard message to let the user know that it does not know the answer.

This is Phase 0 of the project—getting started.



## The FAQ Domain

In this phase, the goal is to make sure you have a basis to start from. You will prepare the data for your FAQ Bot and implement a very simple string-matching FAQ bot using **faq_bot_skeleton.py** on Canvas.

1. Determine your FAQ Bot's knowledge domain and prepare a set of 20 question and answer pairs. One easy way to do this is to find a long Wikipedia page and copy sections of 1 to 3 sentences as each answer (the more text the better), then generate a question to go with each answer. Make sure you reference all online sources in comments. (Reminder: The topic "Mohawk College" is not allowed, but any other topic is fair game.)
2. Store your questions and answers in text files.
3. Create a python program that loads the answers and questions from your file(s), then allows the user to make utterances. Use string matching to find a match between for user's utterance in your list of questions, then output the corresponding answer as a response. You can adapt **faq_bot_skeleton.py** and **file_input.py** for this purpose, or you can write your code from scratch.
4. The bot should also respond to "hello" by greeting the user, and "goodbye" or "quit" by ending the program.
5. If the bot fails to match an utterance, it should politely let the user know that it did not recognize their question.

6. You should incorporate some basic string processing (see the **Strings** handout and **strings.py** on canvas) to make your intent matching a bit more forgiving. At the very least, try to be forgiving with case matching, whitespace, and punctuation.

## Discord Integration

Once the bot is working well in the Python shell, you should make it available as a Discord bot and include a link to add the bot to a server when you hand in. You can use the **simple_discord_faq_bot.py** file on Canvas as a starting point to turn your chat bot into a discord bot.

### Optional: Hosting and Keeping the Bot Alive

If you hand in your Discord token along with your code, the instructor will be able to fire up the bot, so there's no need to do anything further. But one challenge of bot development is how to host and keep the bot alive and running all the time.

The solution is to find a server that is always running, start up the bot on that server, and then monitor to keep it running. If you have access to a server of your own and the ability to set up python on that server, you might be able to use it for this. If not, there are cheap hosting services that specialize in discord bots. For example, SparkedHost charges only $1 (US) per month for its hosting services.

https://sparkedhost.com/discord-bot-hosting



## Handing In

You should start your bot and make sure it's accessible from Discord. Then place all the following into a single project folder, then zip it up and hand it in on Canvas.

1. A folder containing all the code and supporting files for the Python shell and Discord versions of your bot. It should be possible to run the bot from this folder (both standalone and Discord versions).
2. A test file called "instructions.txt". This file should contain the link to the discord version of your bot along with any special instructions required to talk to it (prefixes, etc.). If your bot does anything clever or unusual that you want me to notice, you should write about it here.

See Canvas for the due date.

## EVALUATION

This assignment will be marked out of 5 using the following Rubric. It will be worth 5% of your grade.

| Category | Level 4: 100% | Level 3: 75% | Level 2: 50% | Level 1: 25% |
|---|---|---|---|---|
| **Content (2 points)** | Content consists of 20 or more questions and answers. Each answer consists of 1 or more complete sentences. The questions are a good match for the answers. | Content consists of 20 or more questions and answers. Most answers consist of 1 or more complete sentences. Most questions are a good match for the answers. | Content consists of some questions and answers. Most answers consist of 1 or more complete sentences. Most questions are a good match for the answers. | Content consists of some questions and answers. |
| **String Matching (1 point)** | Use string matching effectively to answer all questions identified by the developer. Offer meaningful responses to other utterances ("hello", "goodbye", "I don't know the answer to that", etc.) | | Use string matching to answer some questions correctly. | |
| **Discord Hosting (1 point)** | Bot can be added to a discord server and functions as well as the Python shell version. | | Bot can be added to a discord server and responds to utterances. | |
| **Internal Documentation (1 point)** | Commenting and naming conventions are consistent with course standards (based on the PEP-8 and PEP-257). All files contain a docstring with a description, author information, and links to original sources. All functions contain a docstring with description of behavior, parameters, and return values. | | Commenting and naming conventions are somewhat consistent with course standards and/or docstrings are missing or incomplete for some files. | |