Custom Chatbot Project

I chose the character descriptions dataset to demostrate a question and answer chatbot. This dataset contains trivial knowledge that is well suited for Q&A sessions. The questions asked would be basic who, what, where, and when questions where the answer would be contained in the dataset.

Data Wrangling

TODO: In the cells below, load your chosen dataset into a `pandas` dataframe with a column named `"text"`. This column should contain all of your text data, separated into at least 20 rows.

In [13]:
```
from pathlib import Path
import pandas as pd
import numpy as np
```

In [27]:
```
df = pd.read_csv("data/character_descriptions.csv")
df.head()
```

Out[27]:

	Name	Description	Medium	Setting
0	Emily	A young woman in her early 20s, Emily is an as...	Play	England
1	Jack	A middle-aged man in his 40s, Jack is a succes...	Play	England
2	Alice	A woman in her late 30s, Alice is a warm and n...	Play	England
3	Tom	A man in his 50s, Tom is a retired soldier and...	Play	England
4	Sarah	A woman in her mid-20s, Sarah is a free-spirit...	Play	England

In [28]:
```
df['text']=df['Name']+' '+df['Description']+' '+df['Medium']+' '+df['Setting']
df.head()
```

Out[28]:

	Name	Description	Medium	Setting	text
0	Emily	A young woman in her early 20s, Emily is an as...	Play	England	Emily A young woman in her early 20s, Emily is...
1	Jack	A middle-aged man in his 40s, Jack is a succes...	Play	England	Jack A middle-aged man in his 40s, Jack is a s...
2	Alice	A woman in her late 30s, Alice is a warm and n...	Play	England	Alice A woman in her late 30s, Alice is a warm...
3	Tom	A man in his 50s, Tom is a retired soldier and...	Play	England	Tom A man in his 50s, Tom is a retired soldier...
4	Sarah	A woman in her mid-20s, Sarah is a free-spirit...	Play	England	Sarah A woman in her mid-20s, Sarah is a free-...

In [29]:
```
df.drop(['Name','Description','Medium','Setting'], axis=1, inplace=True)
df.head()
```

Out[29]:

	text
0	Emily A young woman in her early 20s, Emily is...
1	Jack A middle-aged man in his 40s, Jack is a s...
2	Alice A woman in her late 30s, Alice is a warm...
3	Tom A man in his 50s, Tom is a retired soldier...
4	Sarah A woman in her mid-20s, Sarah is a free-...

Custom Query Completion

TODO: In the cells below, compose a custom query using your chosen dataset and retrieve results from an OpenAI `Completion` model. You may copy and paste any useful code from the course materials.

In [15]:
```
import openai
openai.api_base = "https://openai.vocareum.com/v1"
openai.api_key = "API-Key"
```

In [30]:
```
EMBEDDING_MODEL_NAME = "text-embedding-ada-002"
batch_size = 5
embeddings = []
for i in range(0, len(df), batch_size):
    # Send text data to OpenAI model to get embeddings
    response = openai.Embedding.create(
        input=df.iloc[i:i+batch_size]["text"].tolist(),
        engine=EMBEDDING_MODEL_NAME
    )

    # Add embeddings to list
```

```
    embeddings.extend([data["embedding"] for data in response["data"]])

# Add embeddings list to dataframe
df["embeddings"] = embeddings
df
```

Out[30]:

	text	embeddings
0	Emily A young woman in her early 20s, Emily is...	[-0.018143609166145325, -0.010685139335691929,...
1	Jack A middle-aged man in his 40s, Jack is a s...	[0.004705960862338543, -0.018227731809020042, ...
2	Alice A woman in her late 30s, Alice is a warm...	[0.0036221633199602365, -0.006725931074470282,...
3	Tom A man in his 50s, Tom is a retired soldier...	[0.015647219493985176, -0.018980588763952255, ...
4	Sarah A woman in her mid-20s, Sarah is a free-...	[-0.016781488433480263, -0.020888015627861023,...
5	George A man in his early 30s, George is a cha...	[-0.0212309081107378, -0.014355886727571487, 0...
6	Rachel A woman in her late 20s, Rachel is a sh...	[-0.006127864122390747, -0.012139437720179558,...
7	John A man in his 60s, John is a retired profe...	[0.013520938344299793, -0.014437167905271053, ...
8	Maria A middle-aged Latina woman in her 40s, M...	[-0.010075375437736511, -0.013227262534201145,...
9	Caleb A young African American man in his earl...	[0.001797058735974133, -0.028986914083361626, ...
10	Tyler A white man in his mid-30s, Tyler is a t...	[0.01303091086447239, -0.051223184913396835, -...
11	Sonya A white woman in her late 20s, Sonya is ...	[0.0011662724427878857, -0.03033289685845375, ...
12	Manuel A middle-aged Hispanic man in his 50s, ...	[-0.01588134840130806, -0.0201431792229414, -0...
13	Will A white man in his early 40s, Will is a s...	[-0.0028468179516494274, -0.04675408825278282,...
14	Mia A young Australian woman in her mid-20s, M...	[-0.013416170142591, -0.01895279437303543, 0.0...
15	Lucas A middle-aged Australian man in his 40s,...	[-0.0009651120635680854, -0.014063061214983463...
16	Tahlia A young Indigenous Australian woman in ...	[-0.020322801545262337, -0.011860371567308903,...
17	Max A white Australian man in his late 20s, Ma...	[0.002378160832449794, -0.03563299402594566, -...
18	Ava A middle-aged Australian woman in her 50s,...	[-0.0032959587406367064, -0.004549728706479073...

	text	embeddings
19	Donna A seasoned performer with a larger-than-...	[-0.02835903689265251, -0.015874752774834633, ...
20	Johnny A young up-and-coming performer, Johnny...	[-0.03403228893876076, -0.024665603414177895, ...
21	Sable A sultry and dramatic performer, Sable e...	[-0.03202955797314644, -0.01830260455608368, 0...
22	Dolly A bubbly and vivacious performer, Dolly ...	[-0.03174883872270584, -0.02019905485212803, 0...
23	Vixen A fierce and competitive performer, Vixe...	[-0.03632688522338867, -0.02695518359541893, 0...
24	Karma A chameleon-like performer, Karma is kno...	[-0.010391872376203537, -0.02039184421300888, ...
25	Crystal A quirky and imaginative performer, Cr...	[-0.009465857408940792, -0.018026456236839294,...
26	Olivia A confident and charismatic marketing e...	[-0.011716819368302822, -0.015397356823086739,...
27	Marcus A charming and successful entrepreneur,...	[-0.016546497121453285, -0.040593720972537994,...
28	Maya A kind and compassionate nurse with a hea...	[-0.031968966126441956, -0.02310340106487274, ...
29	James A handsome and athletic personal trainer...	[-0.02719157375395298, -0.015734152868390083, ...
30	Sophia A fun-loving and adventurous travel blo...	[0.008179083466529846, -0.018317947164177895, ...
31	Noah A quirky and creative graphic designer, N...	[-0.02496894635260105, -0.019916342571377754, ...
32	Chloe A driven and ambitious attorney, Chloe i...	[-0.011569351889193058, -0.014301279559731483,...
33	Jake A laid-back and easygoing firefighter, Ja...	[-0.01694687269628048, -0.023319942876696587, ...
34	Prince Lorenzo A charming and handsome prince ...	[-0.0075529501773417, -0.013316872529685497, 0...
35	Signora Rosa A mysterious and alluring woman w...	[0.004425266291946173, -0.006385211832821369, ...
36	Baron Gustavo A wealthy and arrogant nobleman ...	[-0.025302523747086525, -0.021291470155119896,...
37	Francesca A fiery and passionate young woman w...	[-0.01653730683028698, -0.024414049461483955, ...

	text	embeddings
38	Don Carlo A charming and charismatic young man...	[-0.01340782456099987, -0.0117326769977808, 0....
39	Duke Orsino A pompous and self-important noble...	[-0.007580036763101816, -0.04048707336187363, ...
40	Lady Olivia A wealthy and beautiful noblewoman...	[-0.018834782764315605, -0.019713563844561577,...
41	Sir Toby Belch A drunken and lecherous knight ...	[-0.003336531575769186, -0.0320097990334034, -...
42	Malvolio A pompous and self-righteous steward ...	[-0.013383149169385433, -0.03094157949090004, ...
43	Viola A plucky and resourceful young woman who...	[-0.011567608453333378, -0.04079786688089371, ...
44	Sir Andrew Aguecheek A dim-witted and gullible...	[-0.025076858699321747, -0.026693427935242653,...
45	Bianca Lady Olivia's cunning and quick-witted ...	[-0.015172109007835388, -0.02168578840792179, ...
46	Sebastian Viola's twin brother, who is also sh...	[-0.010931655764579773, -0.036083560436964035,...
47	Feste A jester and musician who works in Lady ...	[-0.008233584463596344, -0.02486688829958439, ...
48	Antonio A sea captain who rescues Sebastian fr...	[-0.0067506334744393826, -0.03637021407485008,...
49	Abigail A plucky and resourceful young woman w...	[-0.02940809540450573, -0.022513125091791153, ...
50	Thomas A good-natured and affable young man wh...	[-0.014632567763328552, -0.014632567763328552,...
51	Reverend Brown The pious and stern minister of...	[0.0004625700530596077 4, -0.0320437066257, 0.0...
52	Captain James The charismatic and dashing capt...	[-0.008437766693532467, -0.019885266199707985,...
53	Mrs. Mercer The matriarch of the wealthiest fa...	[-0.024833254516124725, -0.012128020636737347,...
54	Mr. Mercer The bumbling and absent-minded patr...	[-0.015459359623491764, -0.01682836003601551, ...

In [34]:
```
from openai.embeddings_utils import get_embedding, distances_from_embeddings

def get_rows_sorted_by_relevance(question, df):
    """
    Function that takes in a question string and a dataframe containing
    rows of text and associated embeddings, and returns that dataframe
```

```
    sorted from least to most relevant for that question
    """

    # Get embeddings for the question text
    question_embeddings = get_embedding(question, engine=EMBEDDING_MODEL_NAME)

    # Make a copy of the dataframe and add a "distances" column containing
    # the cosine distances between each row's embeddings and the
    # embeddings of the question
    df_copy = df.copy()
    df_copy["distances"] = distances_from_embeddings(
        question_embeddings,
        df_copy["embeddings"].values,
        distance_metric="cosine"
    )

    # Sort the copied dataframe by the distances and return it
    # (shorter distance = more relevant so we sort in ascending order)
    df_copy.sort_values("distances", ascending=True, inplace=True)
    return df_copy
```

In [31]:

```
import tiktoken

def create_prompt(question, df, max_token_count):
    """
    Given a question and a dataframe containing rows of text and their
    embeddings, return a text prompt to send to a Completion model
    """
    # Create a tokenizer that is designed to align with our embeddings
    tokenizer = tiktoken.get_encoding("cl100k_base")

    # Count the number of tokens in the prompt template and question
    prompt_template = """
Answer the question based on the context below, and if the question
can't be answered based on the context, say "I don't know"

Context:

{}

---

Question: {}
Answer:"""

    current_token_count = len(tokenizer.encode(prompt_template)) + \
                          len(tokenizer.encode(question))

    context = []
    for text in get_rows_sorted_by_relevance(question, df)["text"].values:

        # Increase the counter based on the number of tokens in this row
        text_token_count = len(tokenizer.encode(text))
        current_token_count += text_token_count

        # Add the row of text to the list if we haven't exceeded the max
```

```
        if current_token_count <= max_token_count:
            context.append(text)
        else:
            break

    return prompt_template.format("\n\n###\n\n".join(context), question)
```

In [44]:
```
COMPLETION_MODEL_NAME = "gpt-3.5-turbo-instruct"

def answer_question_custom(
    question, df, max_prompt_tokens=1800, max_answer_tokens=150
):
    """
    Given a question, a dataframe containing rows of text, and a maximum
    number of desired tokens in the prompt and response, return the
    answer to the question according to an OpenAI Completion model

    If the model produces an error, return an empty string
    """

    prompt = create_prompt(question, df, max_prompt_tokens)

    try:
        response = openai.Completion.create(
            model=COMPLETION_MODEL_NAME,
            prompt=prompt,
            max_tokens=max_answer_tokens
        )
        return response["choices"][0]["text"].strip()
    except Exception as e:
        print(e)
        return ""
```

In [41]:
```
COMPLETION_MODEL_NAME = "gpt-3.5-turbo-instruct"

def answer_question_basic(
    question, df, max_prompt_tokens=1800, max_answer_tokens=150
):
    """
    Given a question, a dataframe containing rows of text, and a maximum
    number of desired tokens in the prompt and response, return the
    answer to the question according to an OpenAI Completion model

    If the model produces an error, return an empty string
    """

    try:
        response = openai.Completion.create(
            model=COMPLETION_MODEL_NAME,
            prompt=question,
            max_tokens=max_answer_tokens
        )
        return response["choices"][0]["text"].strip()
    except Exception as e:
```

```
        print(e)
        return ""
```

Custom Performance Demonstration

TODO: In the cells below, demonstrate the performance of your custom query using at least 2 questions. For each question, show the answer from a basic `Completion` model query as well as the answer from your custom query.

Question 1

In [45]:
```
answer = answer_question_custom("Whats the name of a man in his 50s?", df)
print(answer)
```

```
Tom, Manuel, James, Tyler
```

In [42]:
```
answer = answer_question_basic("Whats the name of a man in his 50s?", df)
print(answer)
```

```
Some possible names for a man in his 50s could be:

1. John
2. David
3. Robert
4. Michael
5. James
6. Richard
7. William
8. Mark
9. Thomas
10. Paul
11. George
12. Steven
13. Charles
14. Joseph
15. Brian
```

Question 2

In [48]:
```
answer = answer_question_custom("Whats the name of a woman in her 20s?", df)
print(answer)
```

```
Sarah, Emily, Rachel, Sonya, Mia, Tahlia, Chloe, Maya
```

In [43]:
```
answer = answer_question_basic("Whats the name of a woman in her 20s?", df)
print(answer)
```

Some common names for women in their 20s are:

1. Sarah
2. Emily
3. Jessica
4. Emma
5. Ashley
6. Hannah
7. Taylor
8. Olivia
9. Amanda
10. Samantha