AAI520_Final_Group_Train_Generation

October 16, 2023

0.1 AAI-520

0.2 Final Project - Group 6

0.3 Chatbot for Movie Info utilizing the Cornell Movie Dialogs Corpus

This Jupyter Notebook is used to generate the training dataset that will be used to the train the LLM ChatBot

[82]:
```
#@title 1: Load the related Libraries
from __future__ import absolute_import, division, print_function,␣
 ↪unicode_literals
import argparse
import codecs
import csv
import os
import pandas as pd
import ast
import re
```

##2: Load Movie Data Corpus

[83]:
```
is_on_colab = False

google_drive = "/content/drive/MyDrive/AAI-520/Final/Data"
local_dir = "./Dataset/Cornell_Movie_Dialog_Corpus/"
# local_dir = 'C:/Users/alden/AAI520/Final Project/Dataset'

dataset_dir = local_dir
if (is_on_colab):
    dataset_dir = google_drive

train_dir = dataset_dir + "../Train/"
# train_dir = dataset_dir + '/Train/'
if not os.path.exists(train_dir):
    os.makedirs(train_dir)
```

[84]:
```
#@title 2.1: Load line_data
#Load the Line file
```

```
def loadLines(filePath, fields):
    """
    Args:
        filePath (str): full path to the file to load
        fields (set<str>): fields to extract
    Return:
        dict<dict<str>>: the extracted fields for each line
    """
    lines = {}

    with open(filePath, 'r', encoding='iso-8859-1') as f:
        for line in f:
            values = line.split(" +++$+++ ")

            # Extract fields
            lineObj = {}
            for i, field in enumerate(fields):
                lineObj[field] = values[i]

            lines[lineObj['lineID']] = lineObj

    return lines

# Usage example
fields_to_extract = ['lineID', 'characterID', 'movieID', 'character', 'text']
file_path = dataset_dir + "/movie_lines.txt"
lines_data = loadLines(file_path, fields_to_extract)
```

[85]:
```
#@title 2.2: Load the charachter_data
def loadCharacterMetadata(filePath, fields):
    """
    Args:
        filePath (str): full path to the character metadata file to load
        fields (set<str>): fields to extract
    Return:
        dict<dict<str>>: the extracted fields for each character
    """
    characters = {}

    with open(filePath, 'r', encoding='iso-8859-1') as f:
        for line in f:
            values = line.split(" +++$+++ ")

            # Extract fields
            characterObj = {}
            for i, field in enumerate(fields):
                characterObj[field] = values[i]
            characters[characterObj['characterID']] = characterObj

    return characters

# Usage example
character_fields_to_extract = ['characterID', 'characterName', 'movieID',␣
 ↪'movieTitle', 'gender', 'position']
character_file_path = dataset_dir + "/movie_characters_metadata.txt"
character_data = loadCharacterMetadata(character_file_path,␣
 ↪character_fields_to_extract)
```

[86]:
```
#@title 2.3: Load conversation_data
def loadConversations(filePath, fields):
    """
    Args:
        filePath (str): full path to the conversations file to load
        fields (set<str>): fields to extract
    Return:
        list<dict<str>>: a list of dictionaries representing conversations
    """

    conversations = {}

    with open(filePath, 'r', encoding='iso-8859-1') as f:
        for line in f:
            values = line.split(" +++$+++ ")

            # Extract fields
            conversationObj = {}
            for i, field in enumerate(fields):
                conversationObj[field] = values[i]


            conversations[conversationObj['movieID']] = conversationObj

    return conversations

# Usage example
conversation_fields_to_extract = ['characterID1', 'characterID2', 'movieID',␣
 ↪'utteranceIDs']
conversation_file_path = dataset_dir + "/movie_conversations.txt"
conversation_data = loadConversations(conversation_file_path,␣
 ↪conversation_fields_to_extract)
```

[87]:
```
#@title 2.4: Load the title_data
def loadMovieTitlesMetadata(filePath, fields):
    """
    Args:
        filePath (str): full path to the movie titles metadata file to load
        fields (set<str>): fields to extract
    Return:
        dict<dict<str>>: the extracted fields for each movie title
    """
    movie_titles = {}

    with open(filePath, 'r', encoding='iso-8859-1') as f:
        for line in f:
            values = line.split(" +++$+++ ")

            # Extract fields
            movieTitleObj = {}
            for i, field in enumerate(fields):
                movieTitleObj[field] = values[i]

            movie_titles[movieTitleObj['movieID']] = movieTitleObj

    return movie_titles

# Usage example
movie_title_fields_to_extract = ['movieID', 'movieTitle', 'releaseYear',␣
 ↪'imdbRating', 'numVotes', 'genres']
movie_title_file_path = dataset_dir + "/movie_titles_metadata.txt"
movie_title_data = loadMovieTitlesMetadata(movie_title_file_path,␣
 ↪movie_title_fields_to_extract)
```

[88]:
```
#@title 2.5: Load the url_data
def loadRawScriptUrls(filePath, fields):
    """
    Args:
        filePath (str): full path to the raw script URLs file to load
        fields (list<str>): fields to extract
    Return:
        dict<str, dict<str>>: a dictionary with movieID as keys and␣
 ↪dictionaries with field values as values
    """
    urls = {}

    with open(filePath, 'r', encoding='iso-8859-1') as f:
        for line in f:
            values = line.split(" +++$+++ ")
            # Extract fields
            loadRawScriptUrls = {}
            for i, field in enumerate(fields):
                loadRawScriptUrls[field] = values[i]

            urls[loadRawScriptUrls['movieID']] = loadRawScriptUrls

    return urls

# Usage example
raw_script_urls_fields_to_extract = ['movieID', 'scriptURL', 'url']
raw_script_urls_file_path = dataset_dir + "/raw_script_urls.txt"
script_urls_data = loadRawScriptUrls(raw_script_urls_file_path,␣
 ↪raw_script_urls_fields_to_extract)
```

[91]:
```
#@title 3: Convert dictionaries/arrays to DataFrames
df_lines = pd.DataFrame.from_dict(lines_data, orient='index')

df_characters = pd.DataFrame.from_dict(character_data, orient='index')
gender_map = {
    'f': 'female',
    'm': 'male',
    '?': 'unknown'
}
df_characters['gender'] = df_characters['gender'].map(gender_map)

df_conversations = pd.DataFrame.from_dict(conversation_data, orient='index')

df_movie_titles = pd.DataFrame.from_dict(movie_title_data, orient='index')
df_movie_titles['genres'] = df_movie_titles['genres'].apply(ast.literal_eval)

df_script_urls =  pd.DataFrame.from_dict(script_urls_data, orient='index')
```

1 4: Generate the Train dataset

1.0.1 Example text data:

Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: Give three tips for staying healthy. ### Response: 1.Eat a balanced diet and make sure to include plenty of fruits and vegetables. 2. Exercise regularly to keep your body active and strong. 3. Get enough sleep and maintain a consistent sleep schedule.

[96]:
```
# Initialize an empty train list
questions_list = []

def print_last_question():
    for question in questions_list[-1:]:
        print(question)

def add_question(question, answer):
    template = """
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a␣
  ↪response that appropriately answers the question using the Cornell␣
  ↪Movie-Dialog Corpus.
    <</SYS>>

    {}
    [/INST]

    According to the Cornell Movie-Dialog Corpus, {}
    </s>
    """
    result = template.format(question, answer)
    questions_list.append(result)
```

1.0.2 4.1: When was movie released?

[97]:
```
for title, year in zip(df_movie_titles["movieTitle"],␣
  ↪df_movie_titles["releaseYear"]):
    question = f"When was {title} released?"
    answer = f"{title} was released in {year}"
    add_question(question, answer)

print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    When was zulu dawn released?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, zulu dawn was released in 1979
    </s>
```

1.0.3 4.2: What is the rating on the movie {}?

[98]:
```
for title, imdb in zip(df_movie_titles["movieTitle"],␣
 ↪df_movie_titles["imdbRating"]):
    question = f"what is the rating on the movie {title}?"
    answer = f"The rating on {title} is {imdb}"
    add_question(question, answer)


print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    what is the rating on the movie zulu dawn?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, The rating on zulu dawn is
6.40
    </s>
```

1.0.4 4.3: What is the genres on the movie {}?

[99]:
```
for title, genres in zip(df_movie_titles["movieTitle"],␣
 ↪df_movie_titles["genres"]):
    # Convert list of genres to a comma-separated string
    if len(genres) > 1:
        genres_str = ', '.join(genres[:-2]) + ', ' + genres[-2] + ', and ' +␣
 ↪genres[-1]
    elif (len(genres) == 1):
        genres_str = genres[0]
    else:
        genres_str = "unknown"

    question = f"what is the genre of the movie {title}?"
    answer = f"The genres for the movie {title} are {genres_str}"
    add_question(question, answer)

print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
```

```
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    what is the genre of the movie zulu dawn?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, The genres for the movie zulu
dawn are action, adventure, drama, history, and war
    </s>
```

1.0.5 4.4: What gender is the character {} in the movie {}?

[100]:
```python
# what gender is the character {} in the movie {}?

for character, title, gender in zip(df_characters["characterName"],␣
 ↪df_characters["movieTitle"], df_characters["gender"]):
    question = f"what gender is the character {character} in the movie {title}?"
    answer = f"{character}'s gender is {gender}"
    add_question(question, answer)

print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    what gender is the character VEREKER in the movie zulu dawn?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, VEREKER's gender is unknown
    </s>
```

1.0.6 4.5: Do you have the full script for the movie {}?

[101]:
```python
# do you have the full script for the movie {}?

for title, url in zip(df_script_urls["scriptURL"], df_script_urls["url"]):
    question = f"do you have the full script for the movie {title}?"
    answer = f"sure you can find it here {url}"
    add_question(question, answer)
```

```
print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    do you have the full script for the movie zulu dawn?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, sure you can find it here
http://www.aellea.com/script/zuludawn.txt

    </s>
```

1.0.7 4.6: What is the highest rated movies in {}?

[102]:
```
# What are the highest rated movies of {}?

# Get list of possible years
poss_years = []
for year in zip(df_movie_titles["releaseYear"]):
    # Remove non-numeric characters
    year = re.sub('[^0-9]','', str(year))

    # Add the year to the list if it's not in it yet
    if year in poss_years:
        continue
    else:
        poss_years.append(year)

poss_years.sort()
print(f"List of possible years: {poss_years}")

def getRating(item):
    return item['rating']

for targetyear in poss_years:

    titles_list = []

    for year, title, imdb, votes in zip(df_movie_titles["releaseYear"],␣
  ↪df_characters["movieTitle"], df_movie_titles["imdbRating"],␣
  ↪df_movie_titles['numVotes']):
        # Get list of movies from that year
        if year == targetyear:
            titles_list.append({'title': title, 'rating': imdb, 'votes': votes})

    # Grab the movie at the top of list (sorted by rating)
    titles_list.sort(reverse=True, key=getRating)
    top_title = str(titles_list[0]['title'])
    top_rating = str(titles_list[0]['rating'])
    top_votes = str(titles_list[0]['votes'])


    question = f"what is the highest rated movie in {targetyear}?"
    answer = f"the highest rated movie in {targetyear} is {top_title} with an␣
↪IMDb rating of {top_rating} from {top_votes} votes"
    add_question(question, answer)

print_last_question();
```

```
List of possible years: ['1927', '1931', '1932', '1933', '1934', '1936', '1937',
'1939', '1940', '1941', '1942', '1943', '1944', '1945', '1946', '1949', '1950',
'1953', '1954', '1955', '1956', '1957', '1958', '1959', '1960', '1961', '1963',
'1964', '1965', '1966', '1967', '1968', '1969', '1970', '1971', '1972', '1973',
'1974', '1975', '1976', '1977', '1978', '1979', '1980', '1981', '1982', '1983',
'1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
'1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010']

    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    what is the highest rated movie in 2010?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, the highest rated movie in
2010 is airplane ii: the sequel with an IMDb rating of 8.30 from 9 votes
    </s>
```

1.0.8 4.7: How many movies were released in {}?

[103]:
```
# How many movies were released in {}?

template = """
```

```
Below is an instruction that describes a movie related question. Write a␣
↪response that appropriately answers the question.

### Instruction:
how many movies were released in {}?

### Response:
there were {} movies released in {}
"""

# Get list of possible years
# Did this in previous code. See "What is the highest rated movie in {}?"
# print(f"List of possible years: {poss_years}")

for targetyear in poss_years:

    num_movies = 0

    for year in df_movie_titles["releaseYear"]:
        # Get list of movies from that year
        if year == targetyear:
            num_movies += 1

    num_movies = len(df_movie_titles[df_movie_titles["releaseYear"] ==␣
↪targetyear])

    question = f"how many movies were released in {targetyear}?"
    answer = f"there were {num_movies} movies released in {targetyear}"
    add_question(question, answer)


print_last_question();
```

```
    <s>[INST] <<SYS>>
    Below is an instruction that describes a movie related question. Write a
response that appropriately answers the question using the Cornell Movie-Dialog
Corpus.
    <</SYS>>

    how many movies were released in 2010?
    [/INST]

    According to the Cornell Movie-Dialog Corpus, there were 1 movies released
in 2010
    </s>
```

[104]:
```
#@title 5: Store the csv for training
df_from_list = pd.DataFrame({'text': questions_list})
df_from_list.to_csv(train_dir + "train.csv", index=True)
```