FinalProject_NLP520

October 10, 2024

1 ChatBot Final Project: Team 7

Gurleen Virk, Jay Patel, and Will Kencel

October 21, 2024

Professor Kahila Mokhtari Jadid

AAI: NLP520

[132]:
```
# load libraries
import pandas as pd
import numpy as np
import re
import os

import tensorflow as tf
from tensorflow.keras.layers import Input, Embedding, LSTM, Bidirectional,␣
 ↪Dense, Concatenate, TimeDistributed
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
from sklearn.model_selection import train_test_split
```

1.1 Load Data & Text Pre-Processing

[65]:
```
# load movie line from local directory (remember to change to YOUR file␣
 ↪location)
with open('/Users/gurl/Documents/AAI520_NLP/CornellData/movie_lines.txt', 'r',␣
 ↪encoding='utf-8', errors='replace') as file:
    lines = pd.read_table(file, sep='\t', header=None, on_bad_lines='skip')

# load conversation file from github (no need to change anything here for data␣
 ↪to load)
convolines = pd.read_table('https://raw.githubusercontent.com/wkencel/
 ↪Generative-Chatbot-Project/refs/heads/main/movie_conversations.txt',␣
 ↪sep='\t', header=None, encoding='utf-8', on_bad_lines='skip')
```

[66]:
```
# view lines
lines[:10]
```

[66]:
```
                                                   0
0  L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++…
1  L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON ++…
2  L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$…
3  L984 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++…
4  L925 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$…
5  L924 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++…
6  L872 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$…
7  L871 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++…
8  L870 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$…
9  L869 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$…
```

[67]:
```
convolines[:10]
```

[67]:
```
                                                   0
0  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L194', 'L19…
1  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L198', 'L199']
2  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L200', 'L20…
3  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L204', 'L20…
4  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L207', 'L208']
5  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L271', 'L27…
6  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L276', 'L277']
7  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L280', 'L281']
8  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L363', 'L364']
9  u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L365', 'L366']
```

[68]:
```
# Create dictionary to map each line's id with its text
id2line = {}

# Iterate over each row in the dataframe and access the text data
for line in lines[0]:  # Access the first column which contains the movie lines
    _line = line.split(' +++$+++ ')
    if len(_line) == 5:
        id2line[_line[0]] = _line[4]
```

[69]:
```
# Print the first 10 entries
for i, (key, value) in enumerate(id2line.items()):
    if i < 10:  # Change this number to see more or fewer entries
        print(f"{key}: {value}")
    else:
        break
```

```
L1045: They do not!
L1044: They do to!
L985: I hope so.
```

```
L984: She okay?
L925: Let's go.
L924: Wow
L872: Okay -- you're gonna need to learn how to lie.
L871: No
L870: I'm kidding.  You know how sometimes you just become this "persona"?  And
you don't know how to quit?
L869: Like my fear of wearing pastels?
```

[70]:
```
# Create a list of all of the conversations' lines' ids
convs = []
for index, row in convolines.iterrows():
    line = row[0]  # Access the first column of the row
    _line = line.split(' +++$+++ ')[-1][1:-1].replace("'", "").replace(" ", "")
    convs.append(_line.split(','))
```

[71]:
```
# Print the first 10 entries
convs[:10]
```

[71]:
```
[['L194', 'L195', 'L196', 'L197'],
 ['L198', 'L199'],
 ['L200', 'L201', 'L202', 'L203'],
 ['L204', 'L205', 'L206'],
 ['L207', 'L208'],
 ['L271', 'L272', 'L273', 'L274', 'L275'],
 ['L276', 'L277'],
 ['L280', 'L281'],
 ['L363', 'L364'],
 ['L365', 'L366']]
```

[72]:
```
# Sort the sentences: inputs (questions) and targets (answers)
questions = []
answers = []

for conv in convs:
    for i in range(len(conv) - 1):
        if conv[i] in id2line and conv[i + 1] in id2line:
            questions.append(id2line[conv[i]])
            answers.append(id2line[conv[i + 1]])
```

[73]:
```
print("Number of questions:", len(questions))
print("Number of answers:", len(answers))
```

```
Number of questions: 221416
Number of answers: 221416
```

[74]:
```
# Check if data is loaded correctly
limit = 0
```

```
for i in range(limit, limit+5):
    print(questions[i])
    print(answers[i])
    print()
```

```
Can we make this quick?  Roxanne Korrine and Andrew Barrett are having an
incredibly horrendous public break- up on the quad.  Again.
Well, I thought we'd start with pronunciation, if that's okay with you.

Well, I thought we'd start with pronunciation, if that's okay with you.
Not the hacking and gagging and spitting part.  Please.

Not the hacking and gagging and spitting part.  Please.
Okay… then how 'bout we try out some French cuisine.  Saturday?  Night?

You're asking me out.  That's so cute. What's your name again?
Forget it.

No, no, it's my fault -- we didn't have a proper introduction ---
Cameron.
```

[75]:
```
# Create a DataFrame from questions and answers
data = {'Questions': questions, 'Answers': answers}
data = pd.DataFrame(data)
```

[76]:
```
data.head()
```

[76]:
```
                                           Questions  \
0  Can we make this quick?  Roxanne Korrine and A…
1  Well, I thought we'd start with pronunciation,…
2  Not the hacking and gagging and spitting part…
3  You're asking me out.  That's so cute. What's …
4  No, no, it's my fault -- we didn't have a prop…

                                             Answers
0  Well, I thought we'd start with pronunciation,…
1  Not the hacking and gagging and spitting part…
2  Okay… then how 'bout we try out some French …
3                                         Forget it.
4                                           Cameron.
```

[77]:
```
data.shape
```

[77]:
```
(221416, 2)
```

[78]:
```
# Remove duplicates
data.drop_duplicates(inplace  = True)
```

```
[79]: data.shape
```

```
[79]: (220021, 2)
```

```
[80]: # Function for cleaning the text: lowercase, remove punctuations, and replace␣
      ↪certain words

      def clean_text(text):
          '''Clean text by removing unnecessary characters and altering the format of␣
      ↪words.'''

          text = text.lower()

          text = re.sub(r"i'm", "i am", text)
          text = re.sub(r"he's", "he is", text)
          text = re.sub(r"she's", "she is", text)
          text = re.sub(r"it's", "it is", text)
          text = re.sub(r"that's", "that is", text)
          text = re.sub(r"what's", "what is", text)
          text = re.sub(r"where's", "where is", text)
          text = re.sub(r"there's", "there is", text)
          text = re.sub(r"how's", "how is", text)
          text = re.sub(r"\'ll", " will", text)
          text = re.sub(r"\'ve", " have", text)
          text = re.sub(r"\'re", " are", text)
          text = re.sub(r"\'d", " would", text)
          text = re.sub(r"\'re", " are", text)
          text = re.sub(r"won't", "will not", text)
          text = re.sub(r"can't", "cannot", text)
          text = re.sub(r"n't", " not", text)
          text = re.sub(r"n'", "ng", text)
          text = re.sub(r"'bout", "about", text)
          text = re.sub(r"'til", "until", text)
          text = re.sub(r"[-()\"#/@;:<>{}`+=~|.!?,]", "", text)

          return text
```

```
[96]: # Apply the function to the DataFrame
      data['Questions'] = data['Questions'].apply(clean_text)
      data['Answers'] = data['Answers'].apply(clean_text)

      # Display the cleaned DataFrame
      data.head()
```

```
[96]:                                             Questions  \
      0  can we make this quick  roxanne korrine and an…
      1  well i thought we would start with pronunciati…
```

```
2  not the hacking and gagging and spitting part …
3  you are asking me out  that is so cute what is…
4  no no it is my fault  we did not have a proper…

                                           Answers
0  well i thought we would start with pronunciati…
1  not the hacking and gagging and spitting part …
2  okay then how about we try out some french cui…
3                                        forget it
4                                          cameron
```

[99]:
```python
# More text pre-processing
import string

exclude = set(string.punctuation)
remove_digits = str.maketrans('', '', string.digits)
```

[100]:
```python
# More text pre-processing
def preprocess_questions_sentences(sent):
    '''Function to preprocess English Sentence'''
    sent = sent.lower()
    sent = sent.replace("'", '')
    sent = ''.join(ch for ch in sent if ch not in exclude)
    sent = sent.translate(remove_digits)

    sent = sent.strip()
    sent = re.sub(" +", " ", sent)
    return sent


# include SOS (start of sent.) & EOS (end of sent.) tokens
def preprocess_answer_sentence(sent):
    if isinstance(sent, str):
        sent = sent.lower()
        sent = sent.replace("'", '')
        sent = ''.join(ch for ch in sent if ch not in exclude)
        sent = sent.translate(remove_digits)
        sent = sent.strip()
        sent = re.sub(" +", " ", sent)
        sent = "startseq " + sent + " endseq"
        return sent
    else:

        return sent
```

[101]:
```python
# Apply preprocess function on data
data['Questions'] = data['Questions'].apply(preprocess_questions_sentences)
```

```
data['Answers'] = data['Answers'].apply(preprocess_answer_sentence)

# Display the cleaned DataFrame
data.head()
```

[101]:
```
                                           Questions  \
0  can we make this quick roxanne korrine and and…
1  well i thought we would start with pronunciati…
2  not the hacking and gagging and spitting part …
3  you are asking me out that is so cute what is …
4  no no it is my fault we did not have a proper …

                                             Answers
0  startseq well i thought we would start with pr…
1  startseq not the hacking and gagging and spitt…
2  startseq okay then how about we try out some f…
3                        startseq forget it endseq
4                          startseq cameron endseq
```

[83]:
```
# Remove questions and answers shorter than 1 word and longer than 20 words
min_line_length = 1
max_line_length = 20
```

[102]:
```
# Create a function to count the number of words in a text
def count_words(text):
    return len(text.split())

# Filter the DataFrame
filtered_data = data[
    (data['Questions'].apply(count_words).between(min_line_length,␣
 ↪max_line_length)) &
    (data['Answers'].apply(count_words).between(min_line_length,␣
 ↪max_line_length))
]

# Update the original DataFrame
data = filtered_data

data.head()
```

[102]:
```
                                           Questions  \
1  well i thought we would start with pronunciati…
2  not the hacking and gagging and spitting part …
3  you are asking me out that is so cute what is …
4  no no it is my fault we did not have a proper …
9         gosh if only we could find kat a boyfriend
```

```
                                             Answers
1  startseq not the hacking and gagging and spitt…
2  startseq okay then how about we try out some f…
3                        startseq forget it endseq
4                          startseq cameron endseq
9            startseq let me see what i can do endseq
```

[103]:
```python
# Sort Qs and As by length of questions to reduce amount of padding during␣
↪training
# Hope to speed up training and reduce the loss

# Convert questions and answers to their respective lengths
data['Question_Length'] = data['Questions'].apply(lambda x: len(x.split()))
data['Answer_Length'] = data['Answers'].apply(lambda x: len(x.split()))

# Sort Qs and As by length of questions
sorted_questions = []
sorted_answers = []

for length in range(1, max_line_length + 1):
    for index, row in data.iterrows():
        if row['Question_Length'] == length:
            sorted_questions.append(row['Questions'])
            sorted_answers.append(row['Answers'])

# Output the results
print(len(sorted_questions))
print(len(sorted_answers))
print()
for i in range(min(3, len(sorted_questions))):  # Use min to avoid index errors
    print(f"Question {i + 1}: {sorted_questions[i]}")
    print(f"Answer {i + 1}: {sorted_answers[i]}")
    print()
```

```
160580
160580

Question 1: there
Answer 1: startseq where endseq

Question 2: hi
Answer 2: startseq looks like things worked out tonight huh endseq

Question 3: but
Answer 3: startseq you always been this selfish endseq
```

[106]:
```
# Sort the DataFrame by question length
data = data.sort_values(by='Question_Length')

# Reset index if needed
data.reset_index(drop=True, inplace=True)

# Output the sorted DataFrame
data[['Questions', 'Answers', 'Question_Length']].head()
```

[106]:
```
  Questions                                           Answers  \
0      ryan           startseq i am sure you understand endseq
1   exactly  startseq then a sympathetic mouth then a sympa…
2        no  startseq you came because it is taking over yo…
3     lydia  startseq and lydia telling natalie the truth m…
4     jerry                        startseq listen to me endseq

   Question_Length
0                1
1                1
2                1
3                1
4                1
```

[107]:
```
data.shape
```

[107]:
```
(160580, 4)
```

1.1.1 Vectorizing text

[133]:
```
# Convert DataFrame columns to lists
q_sentences = data['Questions'].tolist()
a_sentences = data['Answers'].tolist()

# Define the split ratios
train_ratio = 0.80   # 80% for training
val_ratio = 0.10     # 10% for validation
test_ratio = 0.10    # 10% for testing

# Ensure the sum of ratios equals 1
assert train_ratio + val_ratio + test_ratio == 1.0, "Split ratios must sum to 1.
 ↪"

# Split into training and temporary sets (which will later be split into␣
 ↪validation and test)
train_q_sents, temp_q_sents, train_a_sents, temp_a_sents = train_test_split(
    q_sentences, a_sentences, test_size=(1 - train_ratio), random_state=42,␣
 ↪shuffle=True)
```

```
# Now split the temporary set into validation and test sets
val_size = val_ratio / (val_ratio + test_ratio)  # Calculate validation size␣
↪relative to temp set
val_q_sents, test_q_sents, val_a_sents, test_a_sents = train_test_split(
    temp_q_sents, temp_a_sents, test_size=val_size, random_state=42,␣
↪shuffle=True)
```

[134]:
```
# VOCABULARY
# Filter out non-string elements from training sets
train_q_sents = [str(sent) for sent in train_q_sents]
train_a_sents = [str(sent) for sent in train_a_sents]

# Tokenize question sentences
ques_tokenizer = Tokenizer(oov_token='<OOV>')
ques_tokenizer.fit_on_texts(train_q_sents)
ques_vocab_size = len(ques_tokenizer.word_index) + 1

# Tokenize answer sentences
ans_tokenizer = Tokenizer()
ans_tokenizer.fit_on_texts(train_a_sents)
ans_vocab_size = len(ans_tokenizer.word_index) + 1

print(f"Question Vocabulary Size: {ques_vocab_size}\nAnswer Vocabulary Size:␣
↪{ans_vocab_size}")
```

```
Question Vocabulary Size: 32642
Answer Vocabulary Size: 31517
```

The code prepares question and answer sentences for further processing by converting them to strings, creating tokenizers to build vocabularies, and calculating the vocabulary sizes for both questions and answers.

[135]:
```
max_length = 30

# Convert text to sequences
ques_sequences = ques_tokenizer.texts_to_sequences(train_q_sents)
ans_sequences = ans_tokenizer.texts_to_sequences(train_a_sents)

# Pad sequences
source_seqs = pad_sequences(ques_sequences, maxlen=max_length, padding='post')
target_seqs = pad_sequences(ans_sequences, maxlen=max_length, padding='post')
```

[136]:
```
# Create training dataset
train_dataset = tf.data.Dataset.from_tensor_slices((source_seqs, target_seqs))
train_dataset = train_dataset.shuffle(buffer_size=len(source_seqs)).batch(16,␣
↪drop_remainder=True)
```

```
# Create validation dataset
val_sequences = ques_tokenizer.texts_to_sequences(val_q_sents)
val_sequences = pad_sequences(val_sequences, maxlen=max_length, padding='post')
val_target_sequences = ans_tokenizer.texts_to_sequences(val_a_sents)
val_target_sequences = pad_sequences(val_target_sequences, maxlen=max_length,␣
 ↪padding='post')
val_dataset = tf.data.Dataset.from_tensor_slices((val_sequences,␣
 ↪val_target_sequences))
val_dataset = val_dataset.batch(16, drop_remainder=True)

# Create test dataset
test_sequences = ques_tokenizer.texts_to_sequences(test_q_sents)
test_sequences = pad_sequences(test_sequences, maxlen=max_length,␣
 ↪padding='post')
test_target_sequences = ans_tokenizer.texts_to_sequences(test_a_sents)
test_target_sequences = pad_sequences(test_target_sequences, maxlen=max_length,␣
 ↪padding='post')
test_dataset = tf.data.Dataset.from_tensor_slices((test_sequences,␣
 ↪test_target_sequences))
test_dataset = test_dataset.batch(16, drop_remainder=True)
```

[137]:
```
# Print sizes of the datasets
print(f"Training set size: {len(train_q_sents)}")
print(f"Validation set size: {len(val_q_sents)}")
print(f"Test set size: {len(test_q_sents)}")
```

```
Training set size: 128464
Validation set size: 16058
Test set size: 16058
```

[ ]:

[ ]: