PROJECT TITLE: CHATBOT IMPLEMENTATION IN CUSTOMER SERVICE INDUSTRY

TEAMMATES

Likhita Chandana Adabala
Naveen Kumar Desireddygari

In [1]:
```
import re
import torch
import pandas as pd
from collections import Counter
from torch.utils.data import Dataset, DataLoader
from torch.nn.utils.rnn import pad_sequence
from sklearn.model_selection import train_test_split
```

In [2]:
```
df = pd.read_excel('Conversation_Chatbot.xlsx', usecols=['question', 'answer'])
```

In [3]:
```
df.head()
```

Out[3]:

	question	answer
0	hi, how are you doing?	i'm fine. how about yourself?
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
2	i'm pretty good. thanks for asking.	no problem. so how have you been?
3	no problem. so how have you been?	i've been great. what about you?
4	i've been great. what about you?	i've been good. i'm in school right now.

In [4]:
```
# data preprocessing

def preprocessing(sentence):
    return re.sub('[,.]', '', sentence.lower())
df['question'] = df['question'].apply(preprocessing)
df['answer'] = df['answer'].apply(preprocessing)
```

In [5]:
```
df.head()
```

Out[5]:

	question	answer
0	hi how are you doing?	i'm fine how about yourself?
1	i'm fine how about yourself?	i'm pretty good thanks for asking
2	i'm pretty good thanks for asking	no problem so how have you been?
3	no problem so how have you been?	i've been great what about you?
4	i've been great what about you?	i've been good i'm in school right now

In [6]:
```
class tokenGenerator:
    def __init__(self, conversation):
        self.conversation = conversation
        self.tokens_list = []
        self.word_freq = {}
        self.unique_words = set()
        self.W2I = {}
        self.I2W = {}

    def counter(self):
        self.tokens_list = [word for sentence in self.conversation for word in sent
        self.word_freq = Counter(self.tokens_list)
        self.unique_words = set(self.tokens_list)
        self.W2I = {word: i for i, word in enumerate(self.unique_words)}
        self.I2W = {i: word for word, i in self.W2I.items()}
```

In [7]:
```
quest_tokenizer = tokenGenerator(df['question'])
quest_tokenizer.counter()
ans_tokenizer = tokenGenerator(df['answer'])
ans_tokenizer.counter()
```

In [8]:
```
# Printing out results
print(f"Number of words in questions: {len(quest_tokenizer.tokens_list)}")
print(f"Number of unique words in questions: {len(quest_tokenizer.unique_words)}")
print(f"Most common words in questions: {quest_tokenizer.word_freq.most_common(10)}
print(f"Index of 'the' in questions: {quest_tokenizer.W2I.get('the')}")
print(f"Word at index 100 in questions: {quest_tokenizer.I2W.get(100)}")

print(f"Number of words in answers: {len(ans_tokenizer.tokens_list)}")
print(f"Number of unique words in answers: {len(ans_tokenizer.unique_words)}")
print(f"Most common words in answers: {ans_tokenizer.word_freq.most_common(10)}")
print(f"Index of 'the' in answers: {ans_tokenizer.W2I.get('the')}")
print(f"Word at index 100 in answers: {ans_tokenizer.I2W.get(100)}")
```

```
Number of words in questions: 1193
Number of unique words in questions: 356
Most common words in questions: [('the', 54), ('yale', 40), ('in', 35), ('you', 3
4), ('is', 30), ('a', 28), ('for', 19), ('what', 19), ('to', 19), ('it', 19)]
Index of 'the' in questions: 29
Word at index 100 in questions: attend?
Number of words in answers: 1021
Number of unique words in answers: 343
Most common words in answers: [('the', 39), ('you', 33), ('in', 26), ('yale', 25),
('is', 24), ('a', 23), ('it', 22), ('and', 22), ('yes', 17), ('program', 17)]
Index of 'the' in answers: 26
Word at index 100 in answers: at
```

In [9]:
```
# creating custom dataset
```

In [19]:
```
class OwnDataset(Dataset):
    def __init__(self, conversation, quest_tokenizer, ans_tokenizer):
        self.conversation = conversation
        self.end_token = 1

    def __len__(self):
        return len(self.conversation)

    def __getitem__(self, index):
        quest_and_answer = self.conversation.iloc[index]
        quest_indexes = [quest_tokenizer.W2I[token] for token in quest_and_answer["
        quest_indexes.append(self.end_token)
        ans_indexes = [ans_tokenizer.W2I[token] for token in quest_and_answer["answ
        ans_indexes.append(self.end_token)
        return torch.tensor(quest_indexes), torch.tensor(ans_indexes)
```

In [20]:
```
trainSet, testSet = train_test_split(df, test_size=0.20)

validSet, testSet = train_test_split(testSet, test_size=0.50)
```

In [21]:
```
CustomTrainSet = OwnDataset(trainSet, quest_tokenizer, ans_tokenizer)
CustomTestSet = OwnDataset(testSet, quest_tokenizer, ans_tokenizer)
CustomValidSet = OwnDataset(validSet, quest_tokenizer, ans_tokenizer)
```

In [22]:
```
print(f"Number of samples in training set: {len(CustomTrainSet)}")
print(f"Number of samples in test set: {len(CustomTestSet)}")
print(f"Number of samples in validation set: {len(CustomValidSet)}")
```

```
Number of samples in training set: 104
Number of samples in test set: 13
Number of samples in validation set: 13
```

In [23]:
```
# printing one sample
CustomTestSet[4]
```

Out[23]:
```
(tensor([167, 146,  78, 307, 220,   7,   1]),
 tensor([118, 298, 156, 116,  94,   1]))
```

In [24]:
```
# Creating mini batches
```

In [25]:
```
def padding(mini_batch):
    padding_value=0
    mini_batch = sorted(mini_batch, key=lambda pair: len(pair[0]), reverse=True)
    question_tensor_list, answer_tensor_list = zip(*mini_batch)
    max_len = len(question_tensor_list[0])
    padded_question_tensor = pad_sequence(question_tensor_list, batch_first=True, p
    padded_answer_tensor = pad_sequence(answer_tensor_list, batch_first=True, paddi
    return padded_question_tensor, padded_answer_tensor
```

In [26]:
```
trainDataloader = DataLoader(CustomTrainSet, batch_size=8, shuffle=True, collate_fn
print(f"Number of batches in trainDataloader: {len(trainDataloader)}")
for i, batch in enumerate(trainDataloader):
    if i == 10:
        break
    print(f"\nSample from batch {i}:")
    print(f"Questions:\n{batch[0][0]}")
    print(f"Answers:\n{batch[1][0]}")

valDataloader = DataLoader(CustomTestSet, batch_size=8, shuffle=True, collate_fn=pa
print(f"Number of batches in valDataloader: {len(valDataloader)}")
for i, batch in enumerate(valDataloader):
    if i == 10:
        break
    print(f"\nSample from batch {i}:")
    print(f"Questions:\n{batch[0][0]}")
    print(f"Answers:\n{batch[1][0]}")

testDataloader = DataLoader(CustomValidSet, batch_size=8, shuffle=True, collate_fn=
print(f"Number of batches in testDataloader: {len(testDataloader)}")
for i, batch in enumerate(testDataloader):
    if i == 10:
        break
    print(f"\nSample from batch {i}:")
    print(f"Questions:\n{batch[0][0]}")
    print(f"Answers:\n{batch[1][0]}")
```

```
Number of batches in trainDataloader: 13

Sample from batch 0:
Questions:
tensor([ 29, 343, 285, 183, 181,  19,  77, 206, 289, 132, 181, 102, 224, 238,
        260, 282, 256,  27, 163,   1])
Answers:
tensor([104,   1, 281, 200,  45, 276,  26, 227,   1,   0,   0,   0,   0,   0,
          0,   0,   0])

Sample from batch 1:
Questions:
tensor([188, 167, 225,  78, 311, 224,  28, 345, 106,  83,  79,   1])
Answers:
tensor([104, 243, 229, 174, 225,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0])

Sample from batch 2:
Questions:
tensor([188,  23, 332, 280, 181, 253,  90,  12,  29, 329,  40,  78, 274, 263,
         28, 236,   1])
Answers:
tensor([233, 256, 191, 197, 202, 313,  53,   1,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0])

Sample from batch 3:
Questions:
tensor([ 29, 177,  78, 256,  27, 345, 106,  83,  79, 285, 224, 120,  62, 345,
          1])
Answers:
tensor([104,  35, 204,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0])

Sample from batch 4:
Questions:
tensor([326,  83,  79, 318, 224,  67, 172, 282,  28, 330,  22, 339, 132,  15,
         29, 174, 341, 109, 276, 210, 351, 276,   1])
Answers:
tensor([ 84,  10,  26, 320, 319,  73,   3, 222,   1,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0])

Sample from batch 5:
Questions:
tensor([327,  83,  79, 285, 154, 181, 113, 272, 210, 287, 210,  32, 111, 123,
        285, 258,  29, 319, 247,  86, 284,   1])
Answers:
tensor([ 57, 229, 174, 225, 143,  26, 340,   1,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0])

Sample from batch 6:
Questions:
tensor([203, 176, 293, 251, 140, 342, 282, 325, 224, 321, 262, 210, 338, 129,
        222,   1])
Answers:
tensor([290, 128, 116,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0])

Sample from batch 7:
Questions:
```

```
tensor([203, 318, 248, 224, 177, 210, 224, 226, 161, 210, 298, 131, 211,  78,
        160, 305,  17,  58, 263, 286, 210, 171,  26,   1])
Answers:
tensor([167, 215, 218,  93,  34, 289,   1,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0])

Sample from batch 8:
Questions:
tensor([ 29,  83, 247,  86, 285,  29, 139, 247, 320, 153,  78,  29, 101, 173,
        198,  78,  69, 210,  98, 336,  83, 298,   1])
Answers:
tensor([ 60, 193, 185, 243, 221,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0])

Sample from batch 9:
Questions:
tensor([326,  83,  79, 318, 224,  67, 172, 282,  28, 330,  22, 339, 132,  15,
         29, 174, 341, 109, 276, 210, 351, 276,   1])
Answers:
tensor([ 84,  10,  26, 320, 319,  73,   3, 222,   1,   0,   0,   0,   0,   0,
          0])
Number of batches in valDataloader: 2

Sample from batch 0:
Questions:
tensor([131, 124,  94, 166, 208,   0,  61, 155, 165,  13, 118, 181, 191,  29,
        274, 190, 236,   1])
Answers:
tensor([249, 181,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0])

Sample from batch 1:
Questions:
tensor([188,  23, 332, 280, 181, 218, 184, 224, 190, 278,  78, 310, 263,   1])
Answers:
tensor([ 60, 104, 243, 221,   1,   0,   0,   0,   0,   0])
Number of batches in testDataloader: 2

Sample from batch 0:
Questions:
tensor([ 29, 343, 285, 183, 181, 295, 298, 232, 158,   6, 210, 264, 271,   4,
         12, 292,  35, 168,  78,  29, 117,  46,   1])
Answers:
tensor([  3,  35,  91,  84,  67,  26,  77, 125, 298, 334, 194, 129,   1])

Sample from batch 1:
Questions:
tensor([ 29, 275, 345, 106,  83, 133, 307, 285, 224, 346, 345,   4, 187, 298,
         93, 224, 192, 117, 141,   1])
Answers:
tensor([ 84, 305, 193, 214,  37,   1,   0])
```

In [ ]:

In [ ]:

In [ ]: