ECE 570: Assignment 1

Instructions

1. Please follow the thread in Piazza for detailed usage of Google Colab.
2. All submissions should be uploaded to Gradescope as a PDF version of your current Jupyter notebook. In this assignment you only need to submit section 3, 4, 5, and 6.
3. Have fun!

1. Background

In this assignment, we are trying to do simple sentiment analysis. Sentiment analysis is the process of detecting positive or negative sentiment in text. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers.

The dataset we will be using is called the ***Stanford Sentiment Treebank*** (https://nlp.stanford.edu/sentiment/code.html). This dataset is collected from movie reviews on *Rotten Tomatoes* and includes over 20k sentences. All reviews were later re-organized as distinct phrases with a label as a number between 0.0 and 1.0. Labels can later be divided in to five intervals [0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], (0.8, 1.0] which means very negative, negative, neutral, positive, and very positive, respectively.

The dataset we are using in this assignment is a subset of Stanford Sentiment Treebank and consists of only **400 phrases**. For simplicity, the dataset only contains extreme sentiments; half of them are extremely positive reviews (have corresponding range (0.9, 1.0]), and the other half are extremely negative reviews (have corresponding range [0.0, 0.1]).

In machine learning (ML), the process of optimizing the model given some dataset is known as *training* or *learning*---thus the data used during training is merely called the *training data* (e.g., historicial reviews). This is analagous to a student seeing worked-out problems in class to help understand a concept. However, after the model has been trained, it will be used to predict on new data (e.g., new reviews)---this is what is often called *test data*. This process is analagous to a student taking a test and requires them to understand the concepts rather than just memorizing the problems worked-out in class. We will cover these concepts in more detail later on in the lectures.

To *simulate* this process and evaluate your model *as if it was deployed*, we will split the dataset into two parts: a training dataset (50%) and test dataset (50%), where both have an equal number of positive and negative reviews.

Your job is to manually construct a simple function that determines whether a single phrase (input) has a positive or negative sentiment (output/return value). You are only allowed to see and optimize using the **training dataset** (simulating model training in ML), but we provide a function that performs a final evaluation of your model on the **testing dataset** (simulating the model being deployed in the real world).

Note: We obfuscate (i.e., make secret) the testing function because in the real world, you won't know what the new data will look like.

2. Mounting your google drive on Colab

Since colab is running on a remote server on Google, you need to mount your google drive on Colab to serve as a 'local directory' to your coding environment. Luckily, it is as simple as two steps! Try to run this block and follow the instructions that got popped out.

Note: This part is not necessary if you are using your own Python environment or other remote python environment.

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

3. Load data (10/100 points)

Now, we need to load the data from the "train.txt" and "test.txt" file. Please change the location for **dir_root** in the following code block to where you saved all your files.

The train dataset is stored in the "train.txt" file which stores 100 positive phrases and 100 negative phrases. Each line in the file is consist of a phrase and the corresponding sentiment positive(1) or negative(-1) followed by a separation mark '|'.

Tip: It is helpful and sometimes necessary to have a separate folder for each assignment!

In [1]:
```
import os
# for better path controls
#########################        YOUR CODE        #########################
dir_root = 'D:\Jupyter\ECE570\Assignment-1'    # change this root directory
#########################      END YOUR CODE      #########################
train_dir = os.path.join(dir_root, 'train.txt')
# locate the train.txt file
```

In [2]:
```
# use built-in function "open" to read files
#  and use the "with" syntax to automatically close the file after the block
with open(train_dir, 'r') as f:
    train_lines = f.readlines()

# construct two lists to store phrases and labels seperately
train_data, train_label = [], []
for line in train_lines:
    line_sec = line.split("|", -1)
    train_data.append(line_sec[0])
    train_label.append(int(line_sec[1]))

# preview some data here
preview = 10                # feel free to toggle this number to see more/les
s data
for i, (phrase, label) in enumerate(zip(train_data[:preview], train_label[:pre
view])):
    print(f'Phrase {i:03} \"{phrase}\" has the sentiment {label}')
```

```
Phrase 000 "Astonishingly skillful and moving" has the sentiment 1
Phrase 001 "are incredibly beautiful to look at" has the sentiment 1
Phrase 002 "as the most magical and most fun family fare of this or any recen
t holiday season" has the sentiment 1
Phrase 003 "It shows that some studios firmly believe that people have lost t
he ability to think and will forgive any shoddy product as long as there 's a
little girl-on-girl action ." has the sentiment -1
Phrase 004 "Will assuredly rank as one of the cleverest , most deceptively am
using comedies of the year ." has the sentiment 1
Phrase 005 "disintegrates into a dreary , humorless soap opera" has the senti
ment -1
Phrase 006 "The editing is chaotic , the photography grainy and badly focused
, the writing unintentionally hilarious , the direction unfocused ," has the
sentiment -1
Phrase 007 "The film is often filled with a sense of pure wonderment and exci
tement not often seen in today 's cinema du sarcasm" has the sentiment 1
Phrase 008 "is as appalling as any ` comedy ' to ever spill from a projector
's lens" has the sentiment -1
Phrase 009 "... could easily be called the best Korean film of 2002 ." has th
e sentiment 1
```

4. Classifier (80/100 points)

Please fill in code in the provided skeleton for the function `sentiment_analysis` which has the following structure:

- Input: a single string `phrase`
- Output/Return value: an integer `-1` or `1`. `-1` stands for negative sentiment and `1` stands for positive sentiment

This task is similar to the hand-crafted knowledge phase of AI in which you are manually creating rules or performing manual computation by using your intuition and looking at some examples. You should not use any ML packages for this assignment (simple Python code should be enough).

Notes:

1. To receive full credit, your training accuracy must be greater than 60%. (After some effort, Prof. Inouye was able to get 66.5%. Can you do better?)
2. Your code should be less than **50 lines without importing any additional packages** (i.e, this assignment does not require you to perform any complicated model analysis). Only 10-20 lines is likely required if written concisely.
3. You can view all the training phrases by opening file *'train.txt'* in the provided zip file. This may help understand what could be used in your function.
4. Throughout the design of your algorithm, **you should only have access to the train dataset** stored in "train.txt". The test dataset stored in 'test.npy' is used in the next evaluation section but you should not look at this (see discussion above). Again, you can think that train dataset is what we would actually have to learn from (like course materials and lectures) while test is new data that simulates real-world posts (where we wouldn't usually know the true labels).

You might find the following hints helpful (not required to use them):

1. The Python keyword `in` can be used to determine if a string is within another string. e.g., `'ece' in 'hello ece 570'` would evaluate to `True`.
2. The `lower()` or `upper()` methods of string can be helpful, e.g., if `a = 'HeLlo'` then `print(a.lower())` would print `hello`.
3. A partial frequency table for all words in the training dataset is given as the follow:

Word	# positive	# negative	# total
best	12	0	12
i	0	11	11
are	9	1	10
most	9	1	10
bad	0	10	10
at	2	6	8
his	7	1	8
has	5	3	8
about	2	6	8

Word	# positive	# negative	# total
have	1	6	7
from	2	4	6
worst	0	6	6
does	2	4	6
brilliant	6	0	6
films	6	0	6
any	1	4	5
enough	1	4	5
what	4	1	5
work	5	0	5
great	4	1	5
time	1	4	5
or	1	3	4
some	1	3	4
will	3	1	4
sense	3	1	4
cinema	3	1	4
comedy	1	3	4
just	1	3	4
first	4	0	4
masterpiece	3	1	4
my	0	4	4
want	1	3	4
if	0	4	4
something	3	1	4
story	3	1	4
love	4	0	4
filmmaking	2	2	4
their	4	0	4
when	0	4	4
than	1	3	4
look	1	2	3
recent	3	0	3
product	0	3	3
into	0	3	3
hilarious	2	1	3

Word	# positive	# negative	# total
often	3	0	3
easily	3	0	3
performances	3	0	3
deserves	3	0	3

In [34]:
```
def sentiment_analysis(phrase):
    """
    sentiment_analysis function determines whether a phrase is positive (1) or
negative (-1).

    :param1(string) phrase: a single phrase in the format of string
    :return(int)          : 1 if the phrase is postive or -1 if the phrase is
 negative
    """
    ##########################        YOUR CODE        ########################
#

    score = 0
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
    words = ['the','he','be','to','of','and','a','in','that','have','I','it',
'for','not','on','with','he','as','you','do',
             'at','this','but','his','by','from','they','we','say','her','she'
,'or','an','will','my','one','all','would',
             'there','their','what','so','up','out','if','about','who','get',
'which','go','me','when','make','can','like',
             'time','no','just','him','are']
    positive = ['best','brilliant','most','love','films','great','masterpiece'
,'deserves']
    negative = ['worst','enough','bad']

    for x in phrase:
        if x in punctuations:
            phrase = phrase.replace(x, "")
    res = phrase.lower().split()


    for l in range(3):
        for x in res:
            if x in words or len(x) < 2:
                res.remove(x)

    for s in res:
        if s in positive:
            score = score + 1

        elif s in negative:
            score = score - 1

    if score > 0:
        return 1      # change this line and add lines above
    return -1
    ##########################      END YOUR CODE      ########################
#


def evaluate(func, data, label):
    score = 0
    for x, y in zip(data, label):
        score += (func(x) == y)
    return score/len(data)
```

```
train_acc = evaluate(sentiment_analysis, train_data, train_label)
print(f"Your method has a training accuracy of {train_acc*100}%")
```

```
Your method has a training accuracy of 69.0%
```

5. Evaluate (10/100 points)

You may already notice that there is an extra evaluation function in the above coding block which helps calculate the accuracy for your algorithm on the training dataset. The metric that we used to evaluate is straightforward:

$$\text{accuracy} = \frac{\#\text{ of correct prediction}}{\#\text{ of total cases}}$$

Now, let's test the performance of your algorithm on the test dataset (as if deployed on new reviews)!

Try to get the **test accuracy** to be higher than 55% to receive **full credit**!

Note: You should not have the accuracy to be lower than 50%! (Why?)

In [35]:
```
import sys
sys.path.append(dir_root)
from top_classified_file import super_secret_function

test_dir = os.path.join(dir_root, 'test.npy')
test_acc = super_secret_function(test_dir, sentiment_analysis)

print(f"Your method has a test accuracy of {test_acc*100}%")
```

```
Your method has a test accuracy of 61.5%
```

6. (Optional) Did you notice something interesting?

1. During your design, was the training accuracy always higher than test accuracy? Can you explain why that might be true?
2. Was the sentiment analysis task harder than you expected? Why or why not?
3. Anything else you learned or feedback you may have?

Answers:

1. The frequency table that we used in our algorithm uses words extracted from the training model. Therefore, logically that would generate better results.
2. At first, I thought it would be hard since I have started to create an algorithm that trains itself to achieve better results each time it runs. Nevertheless, when I tried to simplify things, the exercise seemed way easier than I thought.
3. That exercise is much helpful than an introductory exercise. Thank you!

7. (Optional) Further reading and exploration

If you are interested in text data, you could look into the scikit-learn tutorial on analyzing text: https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html (https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html) We will discuss scikit-learn a little later in the semester.

In [ ]: