• Applying language model concepts to an NLP dataset
• sequential text data in python
• (sentiment analysis & text categorization) with language models

PROBLEM 1 – Reading the data

• Read in file "train.tsv" from the Stanford Sentiment Treebank (SST) as shared in the GLUE task
Cleaning the data and train test validation split

PROBLEM 2 – Tokenizing data

• ***function*** that takes a sentence as input, represented as a string, and converts it to a tokenized sequence padded by ***start and end symbols.*** For example, "hello `class`" would be converted to:
  ○ `['<s>', 'hello', 'class', '</s>']`
• Apply your function to all sentences in your training set. Show the tokenization of the first sentence of your training set in your notebook output.
• What is the ***vocabulary size*** of your training set? Include your start and end symbol in your
vocabulary. Show your result in your notebook.

PROBLEM 3 – Bigram counts

•***function*** that takes an array of tokenized sequences as input (i.e., a list of lists) and counts ***bigram*** frequencies in that dataset. Your function should return a two-level dictionary
(dictionary of dictionaries) or similar data structure, where the value at index $[w_i][w_j]$ gives the
frequency count of bigram (w_i, w_j). For example, this expression would give the counts of the bigram "academy award":
`bigram_counts["academy"]["award"]`
• Apply your function to the output of problem 2. You should build one counter that represents all sentences in the training dataset.
• Use this result to show how many times a sentence starts with "the". That is, how often do you
see the bigram ("<s>","the") in your training set? Show results in your notebook.

PROBLEM 4 – Smoothing

• Write a ***function*** that implements formula in that E-NLP textbook (page 129, 6.2 Smoothing and discounting). That is, write a function that applies smoothing and returns a (negative) log-probability of a word given the previous word in the sequence. It is suggested that you use these parameters:
  ○ The current word, w_m
  ○ The previous word, w_{m-1}
  ○ bigram counts (output of Problem 3)
  ○ alpha, a smoothing parameter
  ○ vocabulary size

• Using this function to show the log probability that the word "academy" will be followed by the
word "award". Varying with alpha=0.001 and alpha=0.5 (you should see very different results!).

PROBLEM 5 – Sentence log-probability • a **function** that returns the log-probability of a sentence which is expected to be a
negative number. To do this, assume that the probability of a word in a sequence **only depends**
on the previous word. It is suggested that you use these parameters:

- A sentence represented as a single python string
- bigram counts (output of Problem 3)
- alpha, a smoothing parameter
- vocabulary size

• Use your function to compute the log probability of these two sentences (Note that the 2nd is
not natural English, so it should have a lower (more negative) result that the first):

- "*this was a really great movie but it was a little too long.*"
- "*long too little a was it but movie great really a was this.*"

PROBLEM 6 – Tuning Alpha

Next, use your validation set to select a good value for "alpha".

• Apply the function wrote in Problem 5 to your validation dataset using 3 different values of "alpha", such as (0.001, 0.01, 0.1). For each value, show the log-likelihood estimate of the validation set. That is, in your notebook show the sum of the log probabilities of all sentences.

• Which alpha gives you the best result? To indicate your selection to the grader, save the selected value to a variable named "**selected_alpha**".

PROBLEM 7 – Applying Language Models

In this problem, you will classify your test set of 100 sentences by sentiment, by applying your work
from previous problems and modeling the language of both positive and negative sentiment. To do this, you can follow these steps:

• Separate your training dataset into positive and negative sentences, and compute vocabulary
size and bigram counts for both datasets.

• For each of the 100 sentences in your **test** set:

- Compute both a "positive sentiment score" and a "negative sentiment score" using (1) the function you wrote in Problem 5, (2) Bayes rule, and (3) class priors as computed in Problem 1.
- Compare these scores to assign a **predicted sentiment label** to the sentence.

• What is the class distribution of your **predicted label**? That is, how often did your method
predict positive sentiment, correctly or incorrectly? How often did it predict negative sentiment? Show results in your notebook.

• Compare your ***predicted label*** to the ***true sentiment label***. What is the accuracy of this
experiment? That is, how often did the true and predicted label match on the test set? Show results in your notebook.
For this problem, you do not need to re-tune alpha for your positive and negative