Pre-processing Steps:

a) Removing HTML and (Lowercasing)
b) Removing URL tags (and usernames)
c) Removing whitespace(and expand contractions)
d) Remove punctuations
e) Tokenize and Lemmatize
f) Removing stop words
g) Spelling Correction

PART A:

1. **Method**
    a) Firstly, we create a vocabulary of unique words in the training set.
    b) Then we generate bigrams from the train dataset and record the count with respect to each unique word in the co-occurrence matrix.
    c) We then normalize the matrix to find probability scores
    d) Smoothing: We add 1 to the count where we are getting zero count

2. **Top-4 bigrams and their score(count) after smoothing**

```
[(('can', 'not'), 170),
 (('but', 'not'), 49),
 (('not', 'get'), 48),
 (('wake', 'up'), 44)]
```

3. **Accuracy of Test set using dataset A for training**

```
The accuracy of Test set using dataset A for training is
:0.8509316770186336
```

PART B:

$$score_{w_2} = Compound\ Score\ (from\ Vader)\ of\ word\ w_2$$

$$score_{w_1 w_2} = Compound\ Score\ (from\ Vader)\ of\ word\ w_1 w_2$$

$$\alpha = scaling\ factor$$

If Polarity= 'Positive':

$$\beta_1 = \alpha.u(score_{w_1 w_2}) * score_{w1w2}$$

$$\beta_2 = \alpha.u(score_{w_2}) * score_{w2}$$

If Polarity= 'Negative':

$$\beta_1 = \alpha.u(-score_{w_1 w_2}) * (-score_{w_1 w_2})$$

$$\beta_2 = \alpha.u(-score_{w_2}) * (-score_{w2})$$

Approach 1

1. **Method**

$$Transition\ Probability \propto Count(w_1 w_2) + \beta_2 * prob(w_2)$$

With this approach, we generate sentences such that more importance is given to generating sentimentally rich sentences. Here, we neglect the bigram transition probabilities. This results in semantically poor sentences (which can be seen by high perplexity w.r.t the smoothed model).

2. **Average Perplexity of the generated 500 sentences**

Average perplexity: 202.41238382633253

3. **10 generated samples: 5 positives + 5 negatives**

Positive Samples:

a) jason great morning walk church hang screen
b) millionaire thank enough sleep anything all great
c) branch haha color yay fit like angry
d) daw dog die drug sky hope great
e) canadian wow win gt heart happy mother

Negative Samples

a) frankly sick get good timefollow fail since
b) din jealous nothing wear shit sure italian
c) whoop wrong finish fashion alarm hate use
d) dense male sad not pic new york
e) sici hate phone sea wtf crash blow

4. Accuracy of the test set using dataset B for training

```
The accuracy of Test set using dataset B for training is
:0.8788819875776398
```

Approach 2

1. **Method**

In this approach, we take into account the polarity of bigram/unigram.

$$Transition\ Probability \propto \beta_1 * Count(w_1 w_2 + \beta_2 * prob(w_2))$$

In this approach, we weight the bigram counts with their sentiment score, thus giving more importance to the sentimental transitions rather than randomly choosing sentimentally rich words. This in turn generates sentences with lower perplexity and richer semantic meaning.

We get the highest improvement in model accuracy with respect to this approach.

2. **Average Perplexity of the generated 500 sentences**

Average perplexity: 62.21735592764733

3. **10 generated samples: 5 positives + 5 negatives**

Positive samples:

a) publish nice weekend I love life great
b) audiobook good music play guitar hero party
c) panda thanks may but nice clothes still
d) found best friend add everyone great company
e) floor play much fun party make laugh

Negative Samples:

a) laboratory pain up fake tan think tired
b) chicago no else issue hate happen poor
c) cost evil poorly place fear find new
d) sick little sad last night cry sound
e) mailed no ticket pay weekly webcomic crying

4. Accuracy of the test set using dataset B for training

```
The accuracy of Test set using dataset B for training is
:0.8913043478260869
```