UIP HW 3
Aniruddha Singh Jafa

Collaborators: Paul Kurian, Vidur Singh, Prakhar Jain

Q1)

:: Pre-processing and Lemmatization

- All data pre-processing was done without the use of libraries.
- The following manipulations helped remove special characters:

```
file_contents = file_contents.strip().replace("<br /><br />",

file_contents =  sub("[^\w\d\s]+"," ",file_contents)
```

-
- Based on how -ve or -ve the reviews were, more extreme reviews were replicated in the dataset more times so as to give them more weight (approach credited to Prakhar Jain)
  - Review with score 1,2,3,4 were weighted 4,3,2,1 respectively
  - Reviews with scores 7,8,9,10 were weighted 1,2,3,4 respectively

:: Classification method and justification

- The classifier used is a Naive Bayes classifier, from the scikit library.
- This classifier was chosen for the simplicity of implementation, as well as relatively good performance
- The 'Bernoulli' type Naive Bayes classifier was used from Scikit, since the underlying data was binary (0 for negative review, 1 for positive review).
- While code was run with the Multinomial Naive Bayes classifier as well, the accuracy (0.8425572792704958) was slightly poorer than that of Bernoulli Naive Bayes

```
vectorizer = CountVectorizer(binary = 'true')

training_data_text =
vectorizer.fit_transform(training_data_text)
```

- Each review in the corpus (text data) was converted into a vector of length |V|, where V is the vocabulary. The i'th entry of the review signified the count of the it's word of the vocabulary[1]
- After that, the Bernoulli Bayes classifier was fit to the labeled training data.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

```
my_classifier = BernoulliNB().fit(training_data_text,
training_data_class)
```

- Naive lemmatisation was attempted (using WordNetLemmatizer from nltk.stem), but the computation overhead was too high, and the results were indistinguishable form the non-lemmatised case.

:: Accuracy

condition (on which false positive and false negative are defined): the film being good

accuracy_for_positive is 0.76808

false_negative (says condition doesn't exist, when it actually does):0.23192

accuracy_for_negative is 0.884
false_positive (says condition exists, when it does not) = 0.116

total accuracy of the model is 0.82604

- A slightly better accuracy was found when scores of 0 and 10 were weighted at 10 instead of 4:

accuracy_for_positive data is 0.81968

accuracy_for_negative data is 0.8464

total accuracy is 0.83304

\

Q2)

:: Starts

P(start is H) = 0.5
P(start is C) = 0.5

:: State transitions
Note: not counting last H, since we don't know the weather of the day succeeding it.

P(q_t = H | q_t-1 = H) = #HH / #H = 19 / 21
P(q_t = C | q_t-1 = H) =   #HC / #H =   2 /21

P(q_t = C | q_t-1 = C) = #CC / #C = 12 / 15
P(q_t = H | q_t-1 = C) =   #CH / #H =   3 / 15

:: Emission probabilities

P (V_k = 0 | q_t = H) = 1/7
P(V_k = 1 | q_t = H) = 2/7
P(V_k = 2 | q_t = H) = 4/7

P (V_k = 0 | q_t = C) = 5/9
P(V_k = 1 | q_t = C) = 3/9
P(V_k = 2 | q_t = C) = 1/9

References

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn-naive-bayes-bernoullinb

https://scikit-learn.org/stable/modules/generated
sklearn.feature_extraction.text.CountVectorizer.html