Assignment 1

Natural Language Processing

Index	:	PCDAPR/16/04
Name	:	Herath U.H.M.E.M.
Date of Submission	:	19/02/2017

Question 1

Background

In its core the objective of sentiment analysis is to deciding the intended sentiment behind a given text. Most popular use of sentiment analysis is classifying comments/posts/tweets/etc. in to positive or negative.

One simple yet relatively good enough approach to making such a classifier is the 'n-gram' method. Even though different n-gram classifiers (eg: unigram, bigram, trigram, etc.) performs differently based on the type of text on which they are being trained, it is mostly considered that bigram is a good bet for a considerably accurate n-gram model.

Bigram Model

In bigram approach a model is based upon two probability tables. A unigram probability table and a bigram probability table. The model needs a training corpora to calculate the probability values for the above mentioned tables and once that is completed the model is ready to output the probability of a given test sentence begin belonging to the aforementioned training corpora. The model uses the single word counts (unigram) and two word pairs count (bigram) for the probability calculation.

Steps

The use of bigram model for sentiment analysis can be broken down to following steps.
Two types of training corpora are needed from each sentiment type. In this case a set of documents labeled 'positive' and a set of documents labeled 'negative'.
We use these two corpora separately to calculate each unigram and bigram counts.
For example we will assume that the 'positive' and 'negative' documents contains vocabularies of following words.

	Vocabulary
Positive	I, Liked, Loved, The, Movie
Negative	I, Hated, Didn’t, Like, That, Movie

Table 1.1 - Vocabulary

First we calculate the unigram counts (word frequency for each word) so we get a table such as follows. (values are hypothetical)

	I	Liked	Loved	The	Movie	Hated	Didn’t	Like	That
Positive	15	9	6	8	10	0	0	4	8
Negative	12	0	0	7	11	11	10	10	8

Table 1.2 - Unigram Counts

Then we need to calculate the bigram probabilities. For example if the word ‘Liked’ appears after the word ‘I’ in the positive corpora 7 time we say that the bigram count of the word pair (I, Liked) is 7. And we count the same word combination in negative corpora. By counting all the bigram combinations we get a table such as follows. (Note: not all the Bigrams are shown below due to space restrictions)

	(I,Liked)	(Liked, The)	(I,Hated)	(Hated, The)	(The, Movie)	...
Positive	12	8	0	0	10	...
Negative	1	0	13	7	11	...

Table 1.3 - Bigram Counts

Once we have these two tables our Bigram model is complete.

The next step would be to input a text and use this model to get the most probable sentiment of the input text.

The formal probability calculation can be noted as follows.

$$\mathbf{S = w_1\ w_2\ w_3 \ldots w_i \ldots\ w_n}$$

Where S is the input sentence and w_i is the i th word of that sentence.

$$\mathbf{P_u(w_i) = Count(w_i) / Total\ word}$$

Where $P_u(w_i)$ is the unigram probability of i th word.

$$\mathbf{P_b(w_{i-1,}\ w_i) = Count(w_{i-1,}\ w_i) / P_u(w_i)}$$

Where $P_b(w_{i-1,}\ w_i)$ is the bigram probability of i th word and $Count(w_{i-1,}\ w_i)$ is the number of times the word w_i occurs after the word w_{i-1}.

Finally we have the bigram probability of the input sentence.

$$\mathbf{P_b(S) = P_b(w_{0,}\ w_1) * P_b(w_{1,}\ w_2) * \ldots P_b(w_{i-1,}\ w_i) \ldots * P_b(w_{n-1,}\ w_n)}$$

Where w_0 is the start of sentence tag.

Calculations and Decision Making

We use the above equation to calculate the bigram probability of the given sentence for each of the corpora counts we have. Hypothetically let's assume that we get following Bigram probabilities when calculated on each corpora. (Note: subscript + is used for indicate the probability is from positive corpora and - is used to indicate the probability is from negative corpora)

On positive corpora $P_{b+}(S) = 4.5 \times 10^{-5}$
On negative corpora $P_{b-}(S) = 5.0 \times 10^{-10}$

We see that the $P_{b+}(S) >> P_{b-}(S)$. This indicates that there is a higher probability of the sentence S being appearing in the Positive corpora.

Therefore we (the model) concludes that the sentiment of the sentence S is positive.

Thus the assignment of the sentiment can be achieved using Bigram model through the calculation and then comparison of the Bigram probabilities of the given sentence on each corpora.

Question 2

a)

Comment 1 = “Happy and enjoyable movie.”
Comment 2 = “But for me I cannot support this movie due to the bad behavior from Paul.”

<u>*Steps*</u>

1. *Repeat following for each corpora*
   1.1 Read a corpora (eg: rt-polarity.pos) line by line
   1.2 Tokenize each line
   1.3 Convert tokens to lowercase
   1.4 Remove stop words
   1.5 Lemmatize each word (also disregard fullstop)
   1.6 Count total words
   1.7 Count word frequency
2. Use the counts from 1.6 and 1.7 steps to calculate the unigram probability for each word (subscript ‘+’ indicates the counts and probabilities are from positive corpora and subscript ‘-’ indicates they are from negative corpora)

   $\mathbf{P_{u+}(w_i) = Count_{+}(w_i) / total\_words_{+}}$

   $\mathbf{P_{u-}(w_i) = Count_{-}(w_i) / total\_words_{-}}$
3. Multiply each unigram probability to get the total unigram probability for the comment on each corpora.

   $\mathbf{P_{u+}(s) = P_{u+}(happy) + P_{u+}(enjoyable) + P_{u+}(movie)}$

   $\mathbf{P_{u-}(s) = P_{u-}(happy) + P_{u-}(enjoyable) + P_{u-}(movie)}$

Used languages/libraries/functions

- Python
    - Nltk library
        - WordPunctTokenizer
        - WordNetLemmatizer
        - stopwords

Unigram tables and unigram probabilities

i)

Comment 1 (Total words = 76227)

	happy	enjoable	movie
positive	20	53	673
negative	6	9	927

P_{u+}(comment 1) **= 20/76227 * 53/76227 * 673/76227**
= 1.47313652007e-10

P_{u-}(comment 1) **= 6/76227 * 9/76227 * 927/76227**
= 1.08696787469e-11

ii)

Comment 2 (Total words = 75453)

	cannot	support	movie	due	bad	behavior	paul
positive	12	1	673	5	34	8	9
negative	13	5	927	9	214	9	4

P_{u+}(comment 2) **= 12/75253 * 1/75253 * 673/75253 * 5/75253 * 34/75253 * 8/75253 * 9/75253**
= 6.04594904834e-28

P_{u-}(comment 2) **= 13/75253 * 5/75253 * 927/75253 * 9/75253 * 214/75253 * 9/75253 * 4/75253**
= 2.7989106931e-26

Note : The full python code and code output is attached at the end of the assignment under Appendix A

b)

From the calculations on question (2 a) we have the following

$\mathbf{P_{u+}}$**(comment 1)**	**= 1.47313652007e-10**
$\mathbf{P_{u-}}$**(comment 1)**	**= 1.08696787469e-11**

We see that $\mathbf{P_{u+}}$**(comment 1) > $\mathbf{P_{u-}}$(comment 1)**

Therefore the Comment 1 "*Happy and enjoyable movie.*" can be classified as having a "Positive" sentiment.

$\mathbf{P_{u+}}$**(comment 2)**	**= 6.04594904834e-28**
$\mathbf{P_{u-}}$**(comment 2)**	**= 2.7989106931e-26**

We see that $\mathbf{P_{u+}}$**(comment 2) < $\mathbf{P_{u-}}$(comment 2)**

Therefore the Comment 2 "*But for me I cannot support this movie due to the bad behavior from Paul.*" can be classified as having a "Negative" sentiment.

Question 3

The steps to improve the classification accuracy of the classifier can be addressed through two main approaches.

1. Experiment with other n-gram models and compare the performance through intrinsic evaluations (eg: perplexity) or extrinsic evaluations and then choosing a better n-gram model
2. Optimizing the unigram model itself.

The answer will be focused on the 2nd approach. Following are steps which can be taken in order to optimize a general unigram sentiment analyzer. (Note: Few of the below optimizations are applied in the code in answer to Question 2)

1. Domain Specific Corpora

   Using a general corpora or a corpora that is not directly related to the domain in which the sentiment analyzer would run would minimize the quality of the analyzer. For example, if the sentiment analyzer is for analyzing ‘tweets’ then using a corpora of comments from a more professional and standardized site such as ‘Quora’ will be unwise. This is due to the fact that most of the words/slangs/emoticons/etc we witness in Tweeter will not appear in a comment section of Quora. Thus training on a more domain specific corpora is advised.

2. Lemmatization

   In sentiment analysis the words ‘like, liked, likes’ all points towards the same sentiment (Positive). Therefore it is more efficient to use one word to represent all three variations, since it will reduce the computations we require by a considerable factor when we apply this to all the words. Thus using lemmatization we can represent all the variations of a word through its lemma. This will increase the word count (word frequency) thus helping to have more distinction between the calculations between positive and negative.

3. Removing Stop Word

   Every language has stopwords, words which are common and does not play a huge role in deciding the sentiment of a sentence. These will just add complexity to the model through having to have more calculations, thus it is advised to remove stopwords from the calculations during the training period.

4. Attention to case sensitivity

   This can be argued as a performance improvement and a barrier for performance improvement. We will pay attention to the plus side of having case sensitivity. For example the two occurrences; ‘loved the movie’ and ‘LOVED the movie’ has the same sentiment but the second one has a higher weight on the positive sentiment. Though it’s a subtle change but capturing such differences will increase the accuracy and the confidence of a model which is expected to be sensitive.

5. Handling punctuation and Numeric value

   Removing punctuation and numeric values can improve the performance of the analyzer. For example a negative comment on a movie ticket price will vary based on the theater, show time, movie type, etc. and does not carry much weight towards the sentiment since its relative. Also the use of punctuation differ and also many times the punctuation is used incorrectly in web comments and such. Thus paying attention to punctuation will might lead the analyzer towards erroneous results and also the capturing of words with punctuation will push for a more complex and sparse probability space.

6. Continuous learning

    Though it will make the system more complex, it would be wise to add a mechanism which will enable the analyzer to update its unigram counts on both positive and negative corpora while its functional. This can be achieved through a small integration of reinforcement learning or a separate background task which will use more upto date corpora to update the unigram table.

Above are some of the steps which can be taken in order to enhance the performance of a general unigram based sentiment analyzer.

Appendix A

Full python code for question 2.

```
__author__ = 'eshan'

import io

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import WordPunctTokenizer

POSITIVE_FILE = "rt-polarity.pos"
NEGATIVE_FILE = "rt-polarity.neg"
TOKENIZER = WordPunctTokenizer()
LEMMATIZER = WordNetLemmatizer()
ENGLISH_STOP_WORDS = set(stopwords.words('english'))


def get_tokens(l):
    tokens = TOKENIZER.tokenize(l)
    return [token.lower() for token in tokens]


def get_lemma(word1):
    return LEMMATIZER.lemmatize(word1)


def remove_stop_words(words):
    return [w for w in words if w not in ENGLISH_STOP_WORDS]


def get_unigram_counts(file_name):
    total_words_in_corpora = 0
    word_counts = {}
    with io.open(file_name, encoding='utf-8', errors='ignore') as f:
        for line in f:
            words_of_line = get_tokens(line)
            filtered_words = remove_stop_words(words_of_line)
            for word in filtered_words:
                processed_word = get_lemma(word)
                total_words_in_corpora += 1
                if processed_word not in word_counts.keys():
                    word_counts[processed_word]=1
                else:
                    current_count = word_counts[processed_word]
                    word_counts[processed_word] = current_count+1
    return [total_words_in_corpora, word_counts]


def get_bigram_probability(comment, total_words, word_counts):
    comment_tokens = get_tokens(comment)
    comment_words = remove_stop_words(comment_tokens)
    probability = 1
    for w in comment_words:
        lem = get_lemma(w)
        if lem is not '.':
            if lem not in word_counts.keys():
                print("Error : The analyzer hasn't seen the word " + lem + " during training")
            else:
                probability *= word_counts[lem]*1.0/total_words
    return probability

print("Sentiment Analyzer Creation Started\n")

[total_words_in_positive_corpora, positive_word_counts] = get_unigram_counts(POSITIVE_FILE)
[total_words_in_negative_corpora, negative_word_counts] = get_unigram_counts(NEGATIVE_FILE)

print("Sentiment Analyzer Creation Completed\n")

COMMENT_1 = "Happy and enjoyable movie."
COMMENT_2 = "But for me I cannot support this movie due to the bad behavior from Paul."

print("Analyzing Comment [" + COMMENT_1 + "]")
comment_1_positive_probability = get_bigram_probability(COMMENT_1, total_words_in_positive_corpora, positive_word_counts)
comment_1_negative_probability = get_bigram_probability(COMMENT_1, total_words_in_negative_corpora, negative_word_counts)

print("Comment 1 bigram probability on positive corpora : " + str(comment_1_positive_probability))
print("Comment 1 bigram probability on negative corpora : " + str(comment_1_negative_probability))

print("\nAnalyzing Comment [" + COMMENT_2 + "]")
comment_2_positive_probability = get_bigram_probability(COMMENT_2, total_words_in_positive_corpora, positive_word_counts)
comment_2_negative_probability = get_bigram_probability(COMMENT_2, total_words_in_negative_corpora, negative_word_counts)

print("Comment 2 bigram probability on positive corpora : " + str(comment_2_positive_probability))
print("Comment 2 bigram probability on negative corpora : " + str(comment_2_negative_probability))
```

Code output

```
Sentiment Analyzer Creation Started

Sentiment Analyzer Creation Completed

Analyzing Comment [Happy and enjoyable movie.]
Comment 1 bigram probability on positive corpora : 1.47313652007e-10
Comment 1 bigram probability on negative corpora : 1.08696787469e-11

Analyzing Comment [But for me I cannot support this movie due to the bad behavior from Paul.]
Comment 2 bigram probability on positive corpora : 6.04594904834e-28
Comment 2 bigram probability on negative corpora : 2.7989106931e-26
```