Final Project: Sentiment Analysis

Project questions: Here are some questions intended to guide you:

1. Problem1.

Model#	Feature	Accuracy
M1	Unigrams (absence/presence)	0.85
M2	Unigrams with frequency count	0.805
M3	Unigrams (only adjectives/adverbs)	0.85
M4	Unigrams (sublinear tf-idf)	0.85
M5	Bigrams (absence/presence)	0.835

We do need preprocessing such as limiting the max and min term frequency. First, this limit can improve the accuracy performance (0.845 to 0.85 for tf-idf); second, such preprocessing can reduce feature dimension and reduce time & space complexity of the model.

As we can see, M1, M3 and M4 are doing the best, and unigram frequency is the worst, which means:

1) Word frequency don't matter so much compared with occurrences; the frequency sometimes is noisy information;
2) Adjective and adverbs are meaningful in terms of they are reflecting the sentiment of the writer; ;
3) Tf-idf is also an improvement on pure frequency;
4) Bigram is not an improvement on unigram for this sentiment analysis problem.

M1

```
Important words in positive reviews
1 803.0 film
1 668.0 movie
1 653.0 like
1 578.0 just
1 576.0 time
----------------------------------------
Important words in negative reviews
0 772.0 film
0 733.0 movie
0 693.0 like
0 612.0 just
0 554.0 time
```

M2

```
Important words in positive reviews
1 4703.0 film
1 2227.0 movie
1 1636.0 like
1 1200.0 just
1 1130.0 story
----------------------------------------
Important words in negative reviews
0 3866.0 film
0 2900.0 movie
0 1690.0 like
0 1392.0 just
0 1043.0 time
```

M3

```
Important words in positive reviews
1 1199.0 just
1 1099.0 good
1 729.0 best
1 693.0 really
1 691.0 little
----------------------------------------
Important words in negative reviews
0 1391.0 just
0 1012.0 good
0 928.0 bad
0 709.0 really
0 660.0 little
```

M4

```
Important words in positive reviews
1 22.321209797866082 movie
1 18.722587051105048 like
1 16.861012933794072 story
1 16.81347161698053 life
1 16.742999677712586 just
----------------------------------------
Important words in negative reviews
0 27.18739892096153 movie
0 21.17460434117034 like
0 19.76141374526718 just
0 18.672792925114685 bad
0 17.070786727241277 good
```

These top5 words are showing:

1) Among most unigram models, the indicative words are generally the same, and it's also similar for positive and negative reviews; it is the occurrences or frequency of these words that is determining the sentiment.
2) For adj, adv words it is more intuitive; for example, good and bad is deterministic in the sentiment judgement;
3) If we show top10 words, there are actually more different words than top5, which means some other less frequent words also plays an important part in determining the sentiment of review.

2. Problem2.

Model#	**Feature**	**Accuracy**
M1	Unigrams (absence/presence)	0.865
M2	Unigrams with frequency count	0.815
M3	Unigrams (only adjectives/adverbs)	0.855
M4	Unigrams (sublinear tf-idf)	0.85
M5	Bigrams (absence/presence)	0.835

With Porter's stemmer, multiple models (M1-M3) showed improvements and generally it's worth doing. By stemming the words we may capture the core meaning of multiple words together, while also risking a little bit by losing some information. It turns out the result is an improvement, especially for unigram.

3. Problem 3.

This model is not performing better than the previous n-gram ones.

```
]: from sklearn.feature_extraction import DictVectorizer
   lexicon_value = [{k:v} for k,v in lexicon_clean['flag'].to_dict().items()]
   v = DictVectorizer(sparse=False)
   X = v.fit_transform(lexicon_value)

   threshold = test_features.dot(X).sum(axis=1).mean()
   predictions = (test_features.dot(X).sum(axis=1) > threshold) * 1
   accuracy = accuracy_score(test_labels, predictions)
   print(accuracy)
```

```
0.7
```

In this model I used unsupervised model, with the pre-determined lexicon as input. By getting the pos/neg ratio we then decide the sentiment of it.

This pos/neg ratio is not regarded to the order of the words, or the meaning in the text, it's just assigning each word a value. This is resulting a problem that, it completely neglects the contents, thus cannot distinguish between the review and the review talking about the movie. A sad story movie could be impressive, but the review might spend some words describing the movie, making the review negative falsely.

4. Problem 4.

Adjective and adverb are useful in this classification problem, while n-gram is not actually improving the performance. Filtering out too frequent or too infrequent words could also improve the performance, as was demonstrated in problem 1 tf-idf.

Instead of choosing fixed features, I would like to do more experiments with different combination of POS, e.g. maybe include verbs as well (like, dislike etc could be significant indicator) and also experiment with different n-gram.

However, there still exist some problems with the program design. For example, there might be a bottleneck for performances since movie reviews sometimes describes the movie content instead of the movie itself, in which case it may get the machine confused. Later I will like to try normalization to reduce the dimension of the model, and also play with some other predictive modelling methodologies, such as designing a neural network. Also, this lexicon element could be assembled into the model as well by stacking or bagging.

I also tried building classification model with Fasttext, it's pretty easy and fast to use. After tuning the parameters the best result I could have is 0.88, and 1 or 2 ngram is not making any difference either.

```
eric@eric-ThinkPad-X1-Carbon-5th:~/package/fasttext/fastText-0.1.0$ ./fasttext t
est /home/eric/Dropbox/490/final/model_fasttext.bin /home/eric/Dropbox/490/final
/test_fasttext.txt 1
N       200
P@1     0.88
R@1     0.88
Number of examples: 200
```