06 - Sentiment Analysis (class exercise)

April 24, 2020

1 CountVectorizer and BoW

Analysis of Social Media Contents
Alessandro Ortis - University of Catania

1. Download movie reviews from the following URL (opinions about “The Da Vinci Code”, "Harry Potter" and "Brokeback Mountain" movies labeled as positive or negative) http://www.dmi.unict.it/~ortis/PhDCourseSentiment/davincireviews.txt
2. Vectorize the text using tf-idf (Term Frequency – Inverse Document Frequency) skipping stopwords
3. Split dataset in X_train, X_test, Y_train, Y_test (Y variable is 0 or 1)
4. Train a Naive Bayes classifier with X_train and Y_train
5. Test model with Y_test, X_test

```
In [1]: from pprint import pprint
        file = open('davincireviews.txt','r')
        dataset = file.readlines()
        ds_size = len(dataset)
        print(ds_size)
        pprint(dataset[:5])
```

```
7086
['1\tThe Da Vinci Code book is just awesome.\n',
 "1\tthis was the first clive cussler i've ever read, but even books like "
 'Relic, and Da Vinci code were more plausible than this.\n',
 '1\ti liked the Da Vinci Code a lot.\n',
 '1\ti liked the Da Vinci Code a lot.\n',
 "1\tI liked the Da Vinci Code but it ultimatly didn't seem to hold it's "
 'own.\n']
```

```
In [2]: R = [] # reviews
        S = [] # sentiments
        to_find = 1 # little trick for visualization of sentence examples
        for r_i, review in enumerate(dataset):
            review.replace('\n','')
            [s_label, sentence] = review.split('\t')
```

```
            R.append(sentence)
            label = int(s_label)
            S.append(label)
            if r_i < 10:
                print("("+s_label+")\t"+sentence)
```

```
(1)     The Da Vinci Code book is just awesome.

(1)     this was the first clive cussler i've ever read, but even books like Relic, and Da V

(1)     i liked the Da Vinci Code a lot.

(1)     i liked the Da Vinci Code a lot.

(1)     I liked the Da Vinci Code but it ultimatly didn't seem to hold it's own.

(1)     that's not even an exaggeration ) and at midnight we went to Wal-Mart to buy the Da

(1)     I loved the Da Vinci Code, but now I want something better and different!..

(1)     i thought da vinci code was great, same with kite runner.

(1)     The Da Vinci Code is actually a good movie...

(1)     I thought the Da Vinci Code was a pretty good book.
```

In [3]:
```
#Example of positive review
print(R[0])
first_negative_idx = S.index(0)
# Example of negative review
print(R[first_negative_idx])
```

```
The Da Vinci Code book is just awesome.

da vinci code was a terrible movie.
```

In [4]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english') + ['.',',',':',';','\'','*'])

sentence = R[0]
print(sentence)
# Tokenize the sentence
```

```
        tok_sentence = word_tokenize(sentence)
        # Removing stop words (plus some punctuation)
        f_sentence = [w for w in tok_sentence if not w in stop_words]
        print(f_sentence)
        # Untokenize back the sentence
        sentence = ' '.join(f_sentence)
        print(sentence)
```

```
The Da Vinci Code book is just awesome.

['The', 'Da', 'Vinci', 'Code', 'book', 'awesome']
The Da Vinci Code book awesome
```

Now create a routine to do that.

```
In [5]: stop_words = set(stopwords.words('english') + ['.',',',':',';','\'','*'])
        def sentence_preprocessing(sentence):
            tok_sentence = word_tokenize(sentence)
            f_sentence = [w for w in tok_sentence if not w in stop_words]
            sentence = " ".join(f_sentence)
            return sentence
```

```
In [6]: corpus = []
        for s in R:
            corpus.append(sentence_preprocessing(s))

        pprint(corpus[:2])
```

```
['The Da Vinci Code book awesome',
 "first clive cussler 've ever read even books like Relic Da Vinci code "
 'plausible']
```

```
In [7]: from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizer = TfidfVectorizer()

        X = vectorizer.fit_transform(corpus)
```

```
In [8]: # (sentences, words)
        print(X.shape)
        # the representations are stored in form of dictionaries for computational/space effic
        print(X[0])
        example_vector = X[0].toarray().flatten()
        print(len(example_vector))
```

```
(7086, 2087)
  (0, 1830)        0.39595230139931936
  (0, 431)        0.300116326497803
```

```
  (0, 1971)        0.300116326497803
  (0, 344)        0.3000500585611166
  (0, 227)        0.6569832347943919
  (0, 146)        0.3760653503210292
2087
```

```
In [9]: from sklearn.naive_bayes import MultinomialNB
        clf = MultinomialNB().fit(X, S)

In [10]: docs_new = ['Da vinci code is awesome',
                     'da vinci code is horrible']
         X_new_tfidf = vectorizer.transform(docs_new)

         predicted = clf.predict(X_new_tfidf)
         print(predicted) # 1: positive, 0: negative
```

```
[1 0]
```

Now perform a more accurate inference.

```
In [11]: from sklearn.model_selection import train_test_split

         y = S
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
         clf = MultinomialNB().fit(X_train, y_train)

In [12]: from sklearn import metrics
         test_predicted = clf.predict(X_test)

         print(metrics.classification_report(y_test, test_predicted,
                                             target_names=['neg','pos']))
```

```
              precision    recall  f1-score   support

         neg       0.98      0.96      0.97       626
         pos       0.97      0.99      0.98       792

    accuracy                           0.98      1418
   macro avg       0.98      0.97      0.98      1418
weighted avg       0.98      0.98      0.98      1418
```