README

Decscription

The code is developed based on the Naive Bayes method with the Multinomial distribution. Key parameters are max_df , the upper limit of term document frequency, and α, the smoothing factor used when calculateing the Multinomial probability. With max_df = 1.0 and α =0.5, the code achieves an accuracy of 79.8% in the final testing phase.

Build count matrix

```
WordCountModel = CountVectorizer(corpus,max_df=1.0) ### Create an CountVectorizer object###
WordCountModel.BuildVocabularyDic()  ### Build vocabulary dictionary ###
WordCountModel.BuildWordCountCorpus() ### Construc count matrix ###
X = WordCountModel.CountVectorizer_array ### X is the count matrix
```

Train and predict using the classifier

```
alpha_value=.5
clf = BayesClassifier(alpha_value)
clf.train(X_train,Y_train)
```

Make predictions

```
Y_pred_test = clf.get_pred(X_test)
```

An example of building count matrix based on given corpus

```
corpus = [
    'This is the first document. d',
    'This document is the second document.',
    'And this is the third one.',
    'Is this the first document?',
]
### Create an CountVectorizer object###
WordCountModel = CountVectorizer(corpus,max_df=1.0)

### Build vocabulary dictionary ###
WordCountModel.BuildVocabularyDic()

### Construc count matrix ###
WordCountModel.BuildWordCountCorpus()


### Print the vocablary disctionary and count matrix X###
print(WordCountModel.get_feature_names())

X = WordCountModel.CountVectorizer_array
print(X)
```

The output is:

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```