Predicting Star Trek TOS Spoken Lines with scikit-Learn

2018-07-22 BY GENE

In a previous post, I collected the transcripts of all the original Star Trek seasons and analyzed linguistic features of Kirk and Spock. In this post, I train a scikit-Learn model on the Kirk, Spock and McCoy lines spoken in hopes that one of those three speakers can be guessed when given unknown lines.

Why do this? I am studying machine learning and am trying to come up with exercises! Also, this came about after going through the classic "20 Newsgroups" prediction tutorial. My Kirk, Spock, McCoy exercise is similar to that with a bit of pandas data munging first.

tl;dr: star-trek.py

The first part takes files of lines spoken by the three ST-TOS characters, tokenizes into sentences and then generates a pandas dataframe with columns for the text line and the speaker.

```
import pandas as pd
```

```
import nltk.data

def file_to_df(path, name):
    fh = open(path + '/' + name + '.txt')
    data = fh.read()
    fh.close()
    tokenizer = nltk.data.load('/Users/gene/nltk_data/tokenizers/punkt/engl
    return pd.DataFrame(
        zip(
            tokenizer.tokenize(data),
            ([name] * len(tokenizer.tokenize(data)))
        ),
        columns=['text','person']
    )

path = '/Users/gene/tmp/Kirk-Spock-McCoy'

mccoy = file_to_df(path, 'mccoy')
spock = file_to_df(path, 'spock')
kirk  = file_to_df(path, 'kirk')

result = pd.concat([mccoy, spock, kirk])
```

Next up is to perform the scikit-Learn steps. And the first of those is to split the data into training and testing parts as such:

```
X = result.text
y = result.person

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

Next we use scikit-Learn's CountVectorizer which learns the vocabulary.

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer()

X_train_dtm = vect.fit_transform(X_train)
X_test_dtm  = vect.transform(X_test)
```

By the way, at this point, I also tried the CountVectorizer stop_words hyperparameter, but found that the prediction accuracy was slightly less than the result below.

I also tried the scikit-Learn TfidfTransformer, but again the prediction accuracy was reduced.

So the next step is to use MultinomialNB which learns the association between the text and the speaker. Here is the code to actually fit the model and generate a prediction based on our train/test split:

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB().fit(X_train_dtm, y_train)

y_pred = clf.predict(X_test_dtm)
```

When the prediction accuracy is computed we get 66.6%. Hmmm

```
from sklearn import metrics

metrics.accuracy_score(y_test, y_pred)
# 0.6656685327431324
```

Ok. That is so-so… But let's try to predict some made-up lines and see how it does.

```
docs = [ "Captian's log, Stardate...", 'That is highly illogical.', "He's d

X_new_counts = vect.transform(docs)
predicted = clf.predict(X_new_counts)

for doc, who in zip(docs, predicted):
    print '%r => %s' % (doc, who)
```

These lines are pretty obvious to anyone who has watched the series and the model gets them right. Yay! Other, more ambiguous lines (in X_test/y_test for instance) are not so easy to predict. But 66.6% is decent I think. :-)

Anyway…

```
>>> docs = [ 'our father who art in heaven' ]
>>> X_new_counts = vect.transform(docs)
>>> predicted = clf.predict(X_new_counts)
>>> for doc, who in zip(docs, predicted):
...     print '%r => %s' % (doc, who)
...
'our father who art in heaven' => kirk
```

FILED UNDER: DATA, SOFTWARE
TAGGED WITH: PYTHON, SCIKIT-LEARN, STAR TREK

Epistemologist-at-large

^ Top