Kadyrov_Daniel_Assignment3

October 14, 2020

Daniel Kadyrov

Stevens ID: 10455680

CS557 - Natural Language Processing

Group 32 - Daniel Kadyrov

1 Part 1

Respond to J&M 2nd Exercises 3.10 and 3.11.

1.1 Exercise 3.10

Add an option to your program to generate random sentences.

[1]:
```
from nltk.corpus import brown

corpus = brown.words(fileids=['ca16'])
```

[2]:
```
from nltk import word_tokenize, Text, RegexpTokenizer

# tokens = word_tokenize(" ".join(corpus))
tokenizer = RegexpTokenizer(r"\w+")
tokens = tokenizer.tokenize(" ".join(corpus))
```

[3]:
```
def ngram(text, n):
    for i in range(len(text)-n):
        yield (text[i:i+n])

ngrams = ngram(tokens, 3)
ngrams
```

[3]:
```
<generator object ngram at 0x122d93970>
```

[5]:
```
import numpy as np

def markov_chain(ngrams, length):
```

```
    word_dict = {}
    for gram in ngrams:
        if gram[0] in word_dict.keys():
            word_dict[gram[0]].append(gram[1])
        else:
            word_dict[gram[0]] = [gram[1]]

    chain = [np.random.choice(tokens)]

    for i in range(length):
        chain.append(np.random.choice(word_dict[chain[-1]]))

    chain = " ".join(chain)

    return chain

text = markov_chain(ngrams, 30)
text
```

[5]:
```
'Columnist Walter Monroe Jr and it but more months before getting back to
explain that she hadn t been addressed to Danny Thomas and there A Gift of
masterful movie making'
```

1.2 3.11

Add an option to your program to compute the perplexity of a test set.

[6]:
```
from nltk.lm import MLE
from nltk.lm.preprocessing import padded_everygram_pipeline

train, vocab = padded_everygram_pipeline(2, text)

lm = MLE(2)
lm.fit(train, vocab)
lm.perplexity(text)
```

[6]:
```
55.63655037257783
```

2 Part 2

Find Python packages that apply Bayesian logic to classification and apply one to sentiment data. Such data can be downloaded from Amazon which collects reviews from customers, and possibly specialized rating sites for electronics, entertainment, restaurants, and other products and services. A sentiment data set using Amazon data is available with papers that report on research using it from here (Links to an external site.).

[15]:
```
import re

with open("data/processed_stars/books/all_balanced.review", "r") as file:
    reviews = []
    labels = []
    for line in file:
        review = []
        liner = line.split(" ")
        for word in liner:
            if "#label#" in word:
                labels.append(int(float(word.split(":")[-1].split()[0])))
            else:
                review.append(word.split(":")[0]+" " * int(word.split(":")[1]))
        reviews.append(" ".join(review))
```

[16]:
```
import numpy as np

reviews = np.asarray(reviews)
labels = np.asarray(labels)
```

[17]:
```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(reviews)
```

[18]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, labels, random_state=7)
```

[19]:
```
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
MNB.score(X_test, Y_test)
```

[19]:
```
0.5123546511627907
```

3 Part 3

SentiWordNet can be used with Python and WordNet to sentitimentally classify a corpus because it assigns sentiment values, in positive, negative, and objectivity scores, to WordNet synsets. The following study, "Analyzing Movie Reviews - Sentiment Analysis I", available on Kaggle, follows using SentiWordNet for classification of movie reviews.

The author goes through the reviews, tagging and assigning the SentiWordNet labels to each token of the corpus.

4 Part 4

4.1 6.3

The Senseval 2 Corpus contains data intended to train word-sense disambigua- tion classifiers. It contains data for four words: hard, interest, line, and serve. Choose one of these four words, and load the corresponding data:

```
from nltk.corpus import senseval
instances = senseval.instances('hard.pos')
size = int(len(instances) * 0.1)
train_set, test_set = instances[size:], instances[:size]
```

Using this dataset, build a classifier that predicts the correct sense tag for a given instance. See the corpus HOWTO at http://www.nltk.org/howto for information on using the instance objects returned by the Senseval 2 Corpus.

[20]:
```
from nltk.corpus import senseval

instances = senseval.instances('hard.pos')
size = int(len(instances) * 0.1)
train_set, test_set = instances[size:], instances[:size]

def features(instance):
    feat = dict()
    p = instance.position
       ## previous word and tag
    if p: ## > 0
        feat['wp'] = instance.context[p-1][0]
        feat['tp'] = instance.context[p-1][1]
       ## use BOS if it is the first word
    else: #
        feat['wp'] = (p, 'BOS')
        feat['tp'] = (p, 'BOS')
       ## following word and tag
        feat['wf'] = instance.context[p+1][0]
        feat['tf'] = instance.context[p+1][1]
    return feat

featureset =[(features(i), i.senses[0]) for i in instances if len(i.senses)==1]
```

[21]:
```
from nltk import NaiveBayesClassifier, classify

train, dev, test = featureset[500:], featureset[:250], featureset[250:500]
classifier = NaiveBayesClassifier.train(train)
classify.accuracy(classifier, dev)
```

[21]: 0.944

[22]:
```
classify.accuracy(classifier, test)
```

[22]: 0.9