Wk3_SLP2_Building a sentence segmenter

March 24, 2021

[1]:
```
student_name = "Nikki Fitzherbert"
student_id = "13848336"
```

Step 1 Obtain data already segmented into sentences and convert it into a form suitable for extracting features that can be used to train a classifier.

[2]:
```
import nltk

sents = nltk.corpus.treebank_raw.sents()

tokens = []
boundaries = set()
offset = 0

for sent in sents:
    tokens.extend(sent)
    offset += len(sent)
    boundaries.add(offset - 1)
```

Step 2 Specify the features of the data used in order to determine whether punctuation indicates a sentence boundary. Note that ***tokens*** is a merged list of tokens from the individual sentences and ***boundaries*** is a set containing the indexes of all sentence-boundary tokens.

[3]:
```
def punct_features(tokens, i):
    return {'next-word-capitalized': tokens[i+1][0].isupper(),
            'prev-word': tokens[i-1].lower(),
            'punct': tokens[i],
            'prev-word-is-one-char': len(tokens[i-1]) == 1
           }
```

Step 3 Create a list of labelled feature sets using the feature extractor defined in the previous step by selecting all the punctuation tokens and tagging whether or not they are boundary tokens.

[4]:
```
featuresets = [(punct_features(tokens, i), (i in boundaries))
               for i in range(1, len(tokens)-1)
               if tokens[i] in '.?!;']
```

Step 4 Train and evaluate the punctuation classifier using the featuresets.

[5]:
```
size = int(len(featuresets) * 0.1)
train_set, test_set = featuresets[size:], featuresets[:size]
classifier = nltk.NaiveBayesClassifier.train(train_set)

nltk.classify.accuracy(classifier, test_set)
```

[5]:
```
0.9377049180327869
```

Step 5 The classifier can be used for sentence segmentatiuon by checking each punctuation mark to see whether or not it's labelled as a boundary and then dividing the list of words at the boundary marks.

[6]:
```
def segment_sentence(words):
    start = 0
    sents = []
    for i, word in enumerate(words[:-1]):
        if word in '.?!' and classifier.classify(punct_features(words, i)) ==␣
↪True:
            sents.append(words[start: i+1])
            start = i + 1
        if start < len(words):
            sents.append(words[start:])
    return sents
```

[16]:
```
words = "Hello, my name is Lucky. I am a good boy!"
tokens = nltk.word_tokenize(words)
```

```
['Hello', ',', 'my', 'name', 'is', 'Lucky', '.', 'I', 'am', 'a', 'good', 'boy',
'!']
```

[18]:
```
segment_sentence(tokens)[0]
```

[18]:
```
['Hello',
 ',',
 'my',
 'name',
 'is',
 'Lucky',
 '.',
 'I',
 'am',
 'a',
 'good',
 'boy',
 '!']
```