UNIVERSITÄT WIEN
CSLEARN - EDUCATIONAL TECHNOLOGIES

Natural Language Processing

Exercise Sheet 6

Learning to Classify Text

Exercise 1

Write a name gender classifier using the Names Corpus, the `apply_features` function, shuffling, and a test set of 500 instances. Use the following features:

a) first letter;

b) last letter;

c) last two letters;

d) length;

e) for each letter one feature, which is true if the name contains the letter.

Use the `NaiveBayesClassifier`, calculate the accuracy, and display the 10 most informative features.

Exercise 2

The Senseval 2 Corpus contains data intended to train word-sense disambiguation classifiers. Using this dataset, build a `NaiveBayesClassifier` that predicts the correct sense tag for a given instance for the word "hard":

```
>>> from nltk.corpus import senseval
>>> instances = senseval.instances('hard.pos')
>>> labeled_instances = [(inst, inst.senses) for inst in instances]
>>> size = int(len(labeled_instances) * 0.1)
>>> random.shuffle(labeled_instances)
>>> train_set = apply_features(features, labeled_instances[size:])
>>> test_set = apply_features(features, labeled_instances[:size])
```

Use the preceding and following word as features. They can be calculated by retrieving the position of the word "hard" as `p=inst.position` and then accessing `inst.context[p-1]` and `inst.context[p+1]`.

Run 10 iterations by reshuffling the instances and printing the individual accuracies. Finally, print the average accuracy.

Exercise 3

The synonyms "strong" and "powerful" pattern differently. Use the tagged Brown corpus with the universal tagset to first list the nouns which follow "strong" vs. "powerful". Write for this a function `next_noun(word, tagged_text)` which returns the list of nouns that follow `word` in the `tagged_text`. Build then a `NaiveBayesClassifier` that predicts when each word should be used by using the function `apply_features` and the following noun as single feature.

Run 10 iterations by reshuffling the instances and printing the individual accuracies. Finally, print the average accuracy.

Exercise 4

Based on the Movie Reviews document classifier discussed in this chapter, build a new `NaiveBayesClassifier`. Tag first the Movie Reviews Corpus using the combined tagger from the previous chapter stored in `t2.pkl`. Filter the tagged words to contain only words for the tags ['JJ', 'JJR', 'JJS', 'RB', 'NN', 'NNS', 'VB', 'VBN', 'VBG', 'VBZ', 'VBD', 'QL'] as well as only alphabetic tokens with at least three characters. Convert the words to lowercase. Use the most common 5000 words as `word_features` in the function `document_features`.

Run 10 iterations by reshuffling the instances and printing the accuracy and 5 most informative features for each iteration. Finally, print the average accuracy.

Exercise 5

The PP Attachment Corpus is a corpus describing prepositional phrase attachment decisions. Each instance in the training corpus is encoded as a `PPAttachment` object:

```
>>> from nltk.corpus import ppattach
>>> ppattach.attachments('training')
[PPAttachment(sent='0', verb='join', noun1='board',
              prep='as', noun2='director', attachment='V'),
 PPAttachment(sent='1', verb='is', noun1='chairman',
              prep='of', noun2='N.V.', attachment='N'),
 ...]
>>> inst = ppattach.attachments('training')[1]
>>> (inst.noun1, inst.prep, inst.noun2)
('chairman', 'of', 'N.V.')
```

In the same way, `ppattach.attachments('test')` accesses the test instances. Select only the instances where `inst.attachment` is `'N'`:

```
>>> nattach = [inst for inst in ppattach.attachments('training')
...             if inst.attachment == 'N']
```

Using this sub-corpus, build a `NaiveBayesClassifier` that attempts to predict which preposition is used to connect a given pair of nouns. For example, given the pair of nouns "team" and "researchers", the classifier should predict the preposition "of".

Write for this purpose a function `prepare_featuresets(subcorpus)`, where `subcorpus` is either the string "training" or "test" to return the training set or the test set.

Print the achieved accuracy as well as the result of `classifier.classify({'noun1': 'team', 'noun2': 'researchers'})`.