COMP9318 Project: Predict Stress in English Words

z5098663 Yichen Zhu

1. Features:

1.1. Length of vowels in word

1.2. The vowels in word

1.3. The phonemes before each vowel(Assume 4 vowels)

1.4. The phonemes after each vowel(Assume 4 vowels)

1.5. The type of the word

1.6. prefix in word

1.7. suffix in word

1.1 Length of vowels in word

By running: features_and_label.groupby([0,16]).size()

```
0  16
2  1    21121
   2     6498
3  1     6920
   2     6871
   3     2604
4  1     2341
   2     1546
   3     1684
   4      415
dtype: int64
```

We can get the distribution of prime stress. When there’re 2 vowel phonemes in a word, very like to have 1st vowel as prime stress.

1.2. The vowels in word

I assume there’re 4 vowels [-1,-1,-1,-1], put all vowel phonemes from the word.

1.3&1.4 The phonemes before each vowel(Assume 4 vowels)

The phonemes after each vowel(Assume 4 vowels)

[bef1,af1,bef2,af2,bef3,af3,bef4,af4]

When there's some certain phoneme combinations in a word, there's some connection which one has more weight to be the prime. Therefore I use its index in the vowel list to represent a vowel phoneme. And record the phoneme before and after to represent a combination.

1.5. The type of the word

features_and_label.groupby([13,16]).size()

```
2   5   1       81
        2       84
   11   1    13222
        2     4408
```

There I list 2 kind of types "IN" and "NN".

5 represents "IN", when there's 2 vowels phonemes in a word, the possibility in 1st and 2nd is nearly equal.

11 represents "NN", 65% prime stress is 1st, 25% is 2nd.

I get the type of each word according to the index.

The type of word can be get from nltk.pos_tag([word]). We can see there's some connection.

1.6. prefix in word

According to some survey, for most prefix, the primary stress is just after the prefix. I get 1 or 0 if the word has prefix.

Etc. a'bove a'go com'bine en'courage

1.7. suffix in word

Also for most suffix, the primary stress is before suffix. I get 1 or 0 if the word has suffix.

-ian mu'sician poli'tician
-ic a'tomic demo'cratic

2. Decision Tree Model:

I tried many models like GaussianNB and KNN, but their training and test error is much larger than decision tree. Could be the value is too discrete.

I choose decision tree there because decision tree is easy to understand and interpret, effective in dealing with discrete values. Most features I choose have discrete values.

To implement decision tree we need to import this module:

```
from sklearn.tree import DecisionTreeClassifier
```

Then use the above features, set

```
clf = DecisionTreeClassifier(criterion = "gini",max_depth = depth)
```

To find proper depth for decision and avoid overfitting, I experiment the possible depth and then do the pruning. I implement Reduce Error Pruning there, I tried from 30, do cross validation there, if test error increase after pruning, stop pruning and assume it to be optimal depth. But the result is not outstanding, could be the mistake in selecting features.

I also tried one hot encoder there, but improvement is not obvious.

To do the cross validation, I implement two functions:

```
from sklearn.cross_validation import train_test_split
evaluate(data, repeat_times = 10, depth = 5):
experiment(train, test, features, depth=5):
```

```
for i in range(repeat_times):
        train_data, test_data = train_test_split(data, test_size = 0.2, random_state = 9 + i)
        err_training, err_testing = experiment(train_data, test_data, features, depth)
        total_err_training += err_training
        total_err_testing += err_testing
```

I call experiment in evaluate function, give it the ramdon_state so the sample in train and test sets are randomly.

3. Improvement:

Obviously, from my score in F1, many improvements can be made. It’s really hard to predict the word with 4 vowel phonemes from my present features.

When building a decision tree, if sample data is not sufficient and not well processed, split a node can be tricky. If the value of a feature is not exist, it will be referred as 0, which becomes error. Decision tree is insensitive to missing data. Therefore, more domain knowledge is needed for better features selecting.