Predict Stress in English Words

For this project which is predicting English words stress by machine learning, in order to train classifier and predict result successfully, feature and label are two basic information types should be determined. Selection of features and evaluation will be discussed during the Feature part. Then the label will discussed in part 2. The final part will be conclusion.

1. Feature

In pd.DataFrame(), features are some columns of a data frame which will be fitted in classifier. These features contribute prediction of classifier. Therefore, the selection of features is significant.

For example, the pattern of training dataset as follows:

COED:K OW1 EH2 D

By using split(“ : ”), the data of each line will be split into [“CODE”, “K OW1 EH2 D”].

After file processing, it should be noticed that vowel phonemes and consonant phonemes should be obvious features. Therefore, sets of vowel phonemes and consonant phonemes are pre-defined in the code as v_p and c_p. Arguably, numbers of vowel phonemes ,consonant phonemes and total phonemes also could be weak features.

1.1 Phonemes sequences

The first challenge I met in this project is how to use a matrix to mapping vowel phonemes and consonant phonemes in words as well as the sequences of phonemes. The first idea is creating a function to traverse the phonemes in both v_p and c_p. This function will create (15+24) * (15+24) columns. For “CODE” example, values in column((K,OW),(OW,EH),(EH,D)) will be set 1. The major problem of this idea is unacceptable time complexity. The second idea is storing every phoneme of words into a column by sequences. For “CODE” example, column(1,4) = (K),(OW),(EH),(D). After this process for whole dataset, using DictVectorizer to vectorize these columns. The issue of this approach is obviously, due to the unmatchable combinations of training dataset and test dataset, the training features fitted into classifier do not match the test features fitted into the same classifier. In order to resolve this issue, a fake combination data frame, which hand code all combination of phonemes, is appended to original data frame. After vectorizing new data frame, remove the fake features from the result.

This features are respected to get similarity of phonemes sequences and stress of words.

1.2 Prefixes and suffixes

The second main type of features could be prefixes and suffixes. According to phoneticsiiuam.wikispaces.com[1], both prefixes and suffixes could be divided into two subset. The first two parts are strong_prefixes and strong_suffixes, the similarity of them is when these kind of prefixes or suffixes exist in word, the stress of the word will be shifted. The other two part are neutral_prefixes and neutral_suffixes, these kind of prefixes or suffixes seems like do not change stress position. The approach to implementing this function is create four columns for each words as strong_prefixes, strong_suffixes, neutral_prefixes and neutral_suffixes. If any components of these four set exist in the word, the columns will be set value 1. Arguably, this method may not represent accurate relation between words and suffixes or prefixes.

Noticeable, according to www.wordstress.info[2], another different stress rule splits suffixes into three part. When suffixes in part1 exist, the stress will shifted to prefixes or suffixes. Suffixes belong to part2 will force the stress shifted to the syllable before last one. The part3’s suffixes do not affect stress.

After comparing these two way of how suffixes and prefixes influence stress, the first method achieve better result than second one.

1.3 part of speech

The third main type of features could be property of words such as noun, verb,and so on. Thanks to NLTK(natural language toolkit), the easier way is using nltk.pos_tag function to get taggers. For instance, nltk.pos_tag([“word"]) will get a list [('word', ‘NN’)]. The last element in the list is a tagger. I hand code a set tagger contains common used tagger. Using function to create a columns for each tagger, then mapping tagger exist in the word into these columns.

This features are respected to get similarity of property and stress of words.

2. Label

According to the specification of project, test function should return a list which contains primary stress position among vowel phonemes. The label of the code should be that.

In order to determine which position is primary stress, I hand code a dictionary Pattern. Pattern as follows:

```
pattern = {'10'  : 1,'01'  : 2,
           '100' : 1,'010' : 2,'001' : 3,
           '1000': 1,'0100': 2,'0010': 3,'0001': 4}
```

The key is primary stress pattern associated with different vowel phonemes number. The value is the primary stress position.

For example, the pattern of training dataset as follows:

[“CODE”, “K OW1 EH2 D”].

Every pronunciation will be transfer to pattern like “10”, then using get_postion function to get vowel phonemes.

3. Conclusion

According to the f1_scores of my classifier prediction, there were some interesting things happened. For the same features and max_depth of decision tree classifier, 10 times execution of the codes based on training dataset gives nearly 5 overfitted results, other 5 results have training f1_scores and test f1_scores which have gap values less than 0.1. The reason of this issue might be 2,3 and 4 vowel phonemes differences.

In conclusion, classifier type do influences the accuracy of prediction due to different parameters. However, the most significant factor is features. It seems like if the major feature is optimal, other features will have less contribution for the final result. For this project, the major features I found is sequences of vowel phonemes and non-vowel phonemes.

Reference:

1. http://phoneticsiiuam.wikispaces.com/file/view/affixes_neutral&strong.pdf
2. https://www.wordstress.info/wp-content/uploads/2014/08/Stress_Rules_suffixes.pdf