Paraphrase Recognition: Shallow Model

Framework:

I view this problem as a classfication problem. After extracting several different features in different levels, I combine 130 features to form a feature vector. The features vector are sent to Random Forest classfier. Features and Pairs are listed in the following [1,2]:

9 features:

- Edit Distance(Levenshtein distance)
- Jaro-Winkler distance
- Soundex
- Manhattan distance
- Euclidean distance
- Cosine similarity
- N-gram distance
- Matching coefficient
- Dice coefficient
- Jaccard coefficient

10 string pairs:

- Original String: two strings consisting of the original tokens of S1 and S2, respectively, with the original order of the tokens maintained
- as in the previous case, but now the tokens are replaced by their stems
- as in the previous case, but now the tokens are replaced by their part-of-speech (POS) tags
- as in the previous case, but now the tokens are replaced by their soundex codes
- two strings consisting of only the nouns of S1 and S2, as identified by a POS-tagger, with the original order of the nouns maintained
- as in the previous case, but now with nouns replaced by their stems
- as in the previous case, but now with nouns replaced by their soundex codes
- two strings consisting of only the verbs of S1 and S2, as identified by a POS-tagger, with the original or- der of the verbs maintained
- as in the previous case, but now with verbs replaced by their stems
- as in the previous case, but now with verbs replaced by their soundex codes.

Another Substring Pairs

9 features per pair and together with 10 pairs, we get 90 features. Besides these features, I calculate every substring of longer string to get the 9 features. The substring with highest sum-score is selected together with average sum-score. Hence there will be 10 features. This procedure is applied in first 4

pairs. Then I get totally 130 features.

I use these features and Random Forest (15 trees) to build a classifier. The test data is predicted by this classifier and the result is in results file.

- [1] Malakasiotis, Prodromos, and Ion Androutsopoulos. "Learning textual entailment using SVMs and string similarity measures." Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. Association for Computational Linguistics, 2007.
- [2] Malakasiotis, Prodromos. "Paraphrase recognition using machine learning to combine similarity measures." Proceedings of the ACL-IJCNLP 2009 Student Research Workshop. Association for Computational Linguistics, 2009.

Experiment Details

Procedure

To repeat this experiment, just: (all are written by 徐威迪, 1401111377)

```
make clean && make
```

- The POS taggings are at first generated by using *NLTK* tool.
- The sentences and their ID are seperated in two files by **gen_pos.py**, i.e. 'data/sentence-id/sentence' and 'data/sentence-id/id'.
- Each pair in train/dev/test set is generated by **Pair_Generator.py**
- Given one pair, the features are generated by **Feature_Generator.py**
- **Feature_Combiner.py** is used to generated every aforementioned pairs and corresponding features for train/dev/test set.
- After generating all features, I use RandomForest classifier in *weka* to build a model:

  ```
  java -cp lib/weka.jar weka.classifiers.trees.RandomForest -t data/feature/train_feature.arff -c first -d $@ -T data/feature/dev_feature.arff -I 15
  ```

- The results are generated by:

  ```
  java -cp lib/weka.jar weka.classifiers.trees.RandomForest -t data/feature/train_feature.arff -c first -T data/feature/test_feature.arff -I 15 -classifications weka.classifiers.evaluation.output.prediction.PlainText
  ```

- I use **gen_answer.py** to get the answer style required in the assignment.

Dev-Set Results

The detailed accuracy by class is illustrated in the following:

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
0	0.597	0.138	0.661	0.597	0.627
1	0.862	0.403	0.826	0.862	0.844
weighted	0.780	0.321	0.775	0.780	0.777

The Reasons to Choose This Method

I think the following reasons are most important aspects for me to choose this method.

- Absolutely, there are several cutting-edge methods to solve this problem very well, i.e. 1. build a recursive neural network to generate a vector embedding for paraphrase pairs and then use a convolution neural network to classify two gethered vectors [socher], or 2. just build a translation model between them [...]. But the problem here is lack of data. To train a such model, the quantity of train data here is far too small.
- It's not very meanningful to repeat others' method without any new idea. LOL. I have an idea but have to face the same problem mentioned in the first point.
- I wonder if semi-supervised method is available here.