W4705 HW2 (wj2227)

For fast run of all scripts, please use *run.sh*

Question 4

Part1: how to run your code step by step

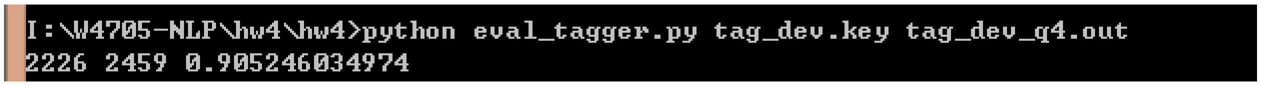
To run the code, please follow the following commands:

```
python question4.py tag.model tag_dev.dat > tag_dev_q4.out
python eval_tagger.py tag_dev.key tag_dev_q4.out
```

After running, the test sentences in *tag_dev.dat* will be labeled and output to file "*tag_dev_q4.out*".

Part2: performance for your algorithm

Using the given model, the performance of in tagging test sentences is show below:



```
I:\W4705-NLP\hw4\hw4>python eval_tagger.py tag_dev.key tag_dev_q4.out
2226 2459 0.905246034974
```

The evaluation shows that the accuracy is 90.5% with this model using bigram features.

Part3: observations and comments about your experimental results.

This model uses only bigram features and get a fairly high accuracy, which is about 10% higher than the best result in HW1. It indicates that as the loglinear model encodes more information, it can give better performance than simple HMM model. This model is used as baseline for later comparison.

Question 5

Part1: how to run your code step by step

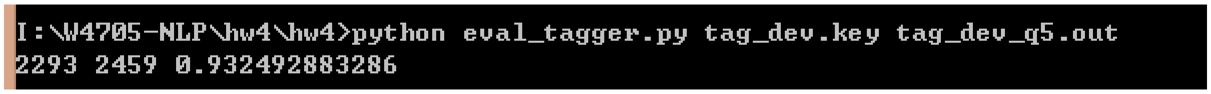
To run the code, please follow the following commands:

```
python question5.py tag_train.dat tag_dev.dat > tag_dev_q5.out
python eval_tagger.py tag_dev.key tag_dev_q5.out
```

After running, the test sentences in *tag_dev.dat* will be labeled and output to file "*tag_dev_q5.out*". Besides, the model trained by Perceptron algorithm is saved in "*suffix_tagger.model*".

Part2: performance for your algorithm

Using the new model, the performance of in tagging test sentences is show below:



```
I:\W4705-NLP\hw4\hw4>python eval_tagger.py tag_dev.key tag_dev_q5.out
2293 2459 0.932492883286
```

The evaluation shows that the accuracy is 93.2% with this model using bigram and suffix features.

Part3: observations and comments about your experimental results.

The new model trained by Perceptron algorithm with bigram and suffix features gives 93.2% accuracy, which has an improvement of ~3% from the baseline. It indicates that the addition suffix information is useful for the part-of-speech tagging problem. Since the suffix usually is relevant to the form of different types of words (for example, *ing, ed, er, ly*), it is not surprising that the inclusion of suffix should makes the result better. Since the baseline accuracy is already good, the improvement may be hard and 3% is a fairly large improvement.

Question 6

Part1: how to run your code step by step

To run the code, please follow the following commands:

python question6_1.py tag_train.dat tag_dev.dat > tag_dev_q6_1.out
python eval_tagger.py tag_dev.key tag_dev_q6_1.out

python question6_2.py tag_train.dat tag_dev.dat > tag_dev_q6_2.out
python eval_tagger.py tag_dev.key tag_dev_q6_2.out

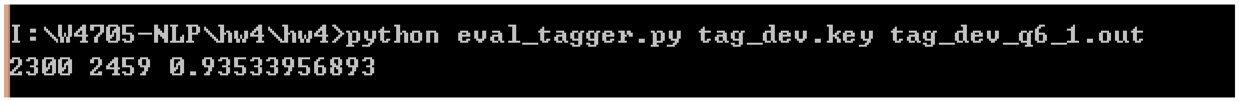
python question6_3.py tag_train.dat tag_dev.dat > tag_dev_q6_3.out
python eval_tagger.py tag_dev.key tag_dev_q6_3.out

After running, the test sentences in *tag_dev.dat* will be labeled and output to file "*tag_dev_q6_1.out*", "*tag_dev_q6_2.out*" and "*tag_dev_q6_3.out*" respectively, and the corresponding model trained by Perceptron algorithm is saved in "*suffix_prefix_tagger.model*", "*suffix_prefix_preword_tagger.model*" and "*suffix_prefix_cap_tagger.model*".

Part2: performance for your algorithm

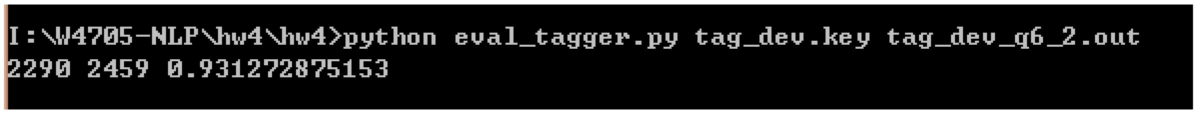
Three models are used in this parts:

1. suffix + prefix + bigram: 93.5% accuracy



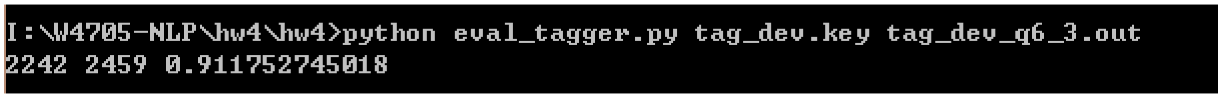
```
I:\W4705-NLP\hw4\hw4>python eval_tagger.py tag_dev.key tag_dev_q6_1.out
2300 2459 0.93533956893
```

2. suffix + prefix + word-bigram + bigram: 93.1% accuracy



```
I:\W4705-NLP\hw4\hw4>python eval_tagger.py tag_dev.key tag_dev_q6_2.out
2290 2459 0.931272875153
```

3. suffix + prefix + capital + bigram: 91.2% accuracy



```
I:\W4705-NLP\hw4\hw4>python eval_tagger.py tag_dev.key tag_dev_q6_3.out
2242 2459 0.911752745018
```

Part3: observations and comments about your experimental results.

Models with different features were tried. When adding prefix feathers (first 1, 2 and 3 letters of words) to the model in question5, the accuracy further improves 0.5% to 93.5%, which is the best result seen during experiment. When adding the word-bigram features to previous model, the accuracy dropped 0.1% below the model with only suffix features. Similarly, when adding features of capital initial words and all capital words, the accuracy drop to 91.2, which is better than the baseline but worse than other models used. Other experiments were also carried out but results not represented here, most of which have accuracy falling in 92-93% range. So it is not the case that the more information encapsulated in the model, the better the results, which leads to the difficulty of choosing features.