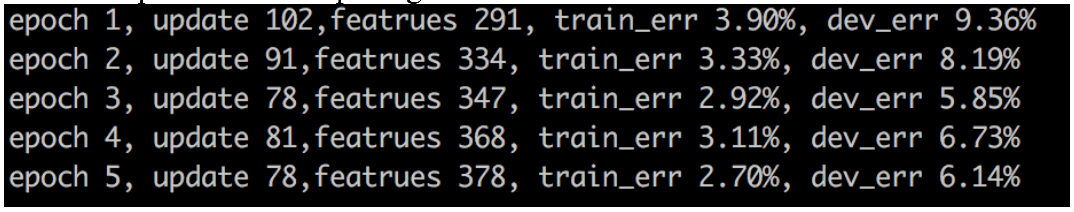
Machine Learning, Individual HW 3: Structured Prediction

Xiao Tan tanx2@oregonstate.edu

1) Structured Perceptron

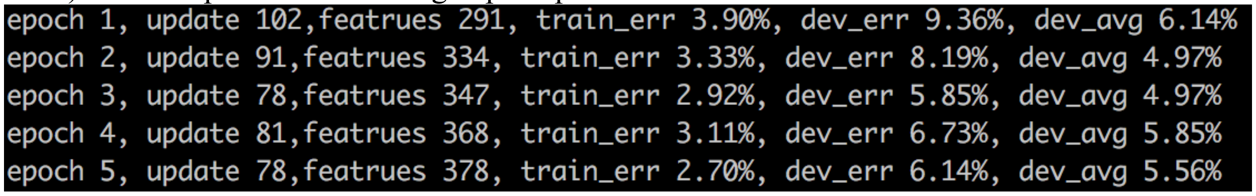
a) First just use two feature templates: ⟨t, t′⟩ and ⟨t, w⟩. Training unaveraged perceptron for 5 epochs. Which epoch gives the best error rates on dev?



```
epoch 1, update 102,featrues 291, train_err 3.90%, dev_err 9.36%
epoch 2, update 91,featrues 334, train_err 3.33%, dev_err 8.19%
epoch 3, update 78,featrues 347, train_err 2.92%, dev_err 5.85%
epoch 4, update 81,featrues 368, train_err 3.11%, dev_err 6.73%
epoch 5, update 78,featrues 378, train_err 2.70%, dev_err 6.14%
```

Training unaveraged perceptron for 5 epochs, I got the results as the picture shows upon. The best error rates will be 5.85%, while epoch 3.

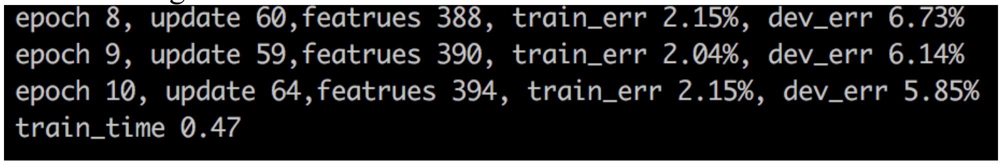
b) Now implement the averaged perceptron. What is the new best error rate on dev?



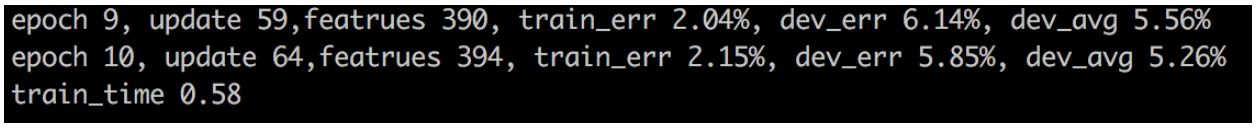
```
epoch 1, update 102,featrues 291, train_err 3.90%, dev_err 9.36%, dev_avg 6.14%
epoch 2, update 91,featrues 334, train_err 3.33%, dev_err 8.19%, dev_avg 4.97%
epoch 3, update 78,featrues 347, train_err 2.92%, dev_err 5.85%, dev_avg 4.97%
epoch 4, update 81,featrues 368, train_err 3.11%, dev_err 6.73%, dev_avg 5.85%
epoch 5, update 78,featrues 378, train_err 2.70%, dev_err 6.14%, dev_avg 5.56%
```

Here are the results of averaged perceptron for 5 epochs, I got the best error rate as 4.97% in epoch 2 and epoch 3.

c) How much slower is averaged perceptron? Compare the time between unaveraged and averaged version.



```
epoch 8, update 60,featrues 388, train_err 2.15%, dev_err 6.73%
epoch 9, update 59,featrues 390, train_err 2.04%, dev_err 6.14%
epoch 10, update 64,featrues 394, train_err 2.15%, dev_err 5.85%
train_time 0.47
```

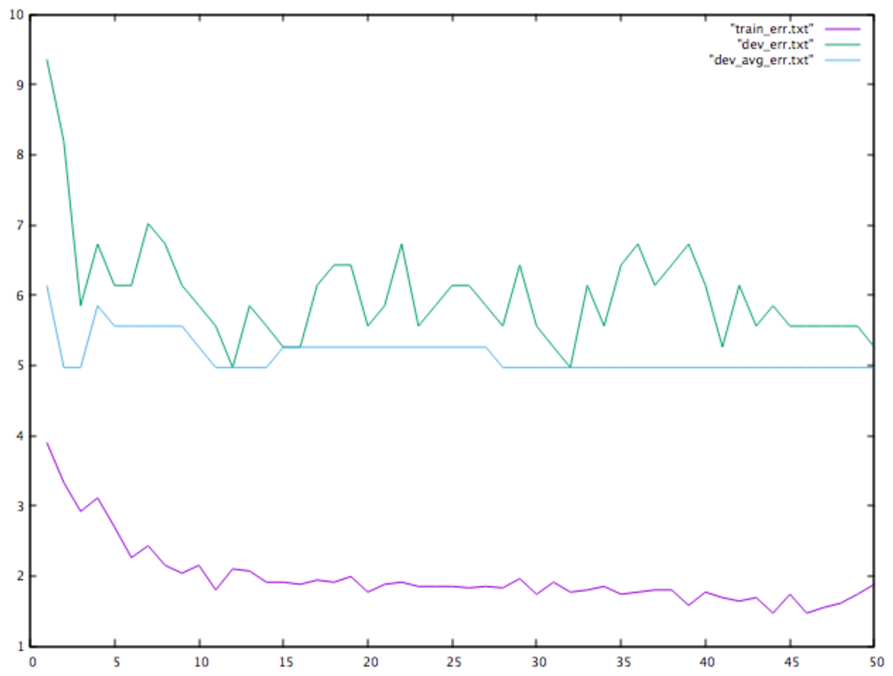


```
epoch 9, update 59,featrues 390, train_err 2.04%, dev_err 6.14%, dev_avg 5.56%
epoch 10, update 64,featrues 394, train_err 2.15%, dev_err 5.85%, dev_avg 5.26%
train_time 0.58
```

Picture 1 and 2 show the time used by unaveraged and averaged perception for 10 epoch. Averaged perceptron will cost more time than unaveraged, but not too much.

d) Plot a single plot that includes four curves: {unaveraged, averaged} x {train err, dev err}. What did you find from this plot?

Because unaveraged and averaged train err will be the same, so we can only see 3 curves on the plot. The result is showed in following graph.



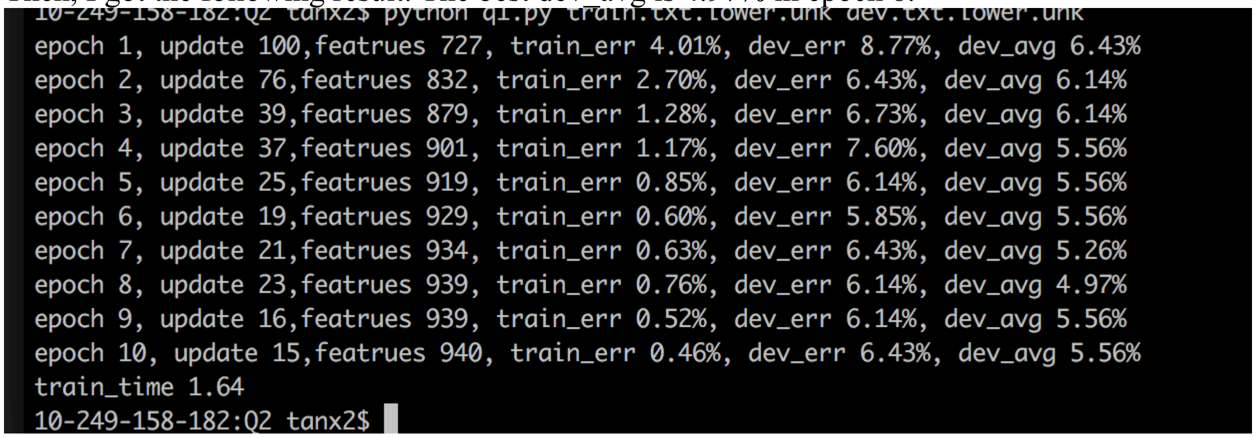
e) Explain why we replaced all one-count and zero-count words by <unk>.

Because training data won't have every word we may test in the future, replaced all one-count and zero-count words by <unk> will make this model can handle unknown words, no matter what words it is, just treat those low frequency words or unknown words as <unk>.

2) Feature Engineering (based on averaged perceptron)

I tried t0_(t-1_t-2) and t0_w0 as template to solve this problem. Use t-1_t-2 instead of t-1 only to achieve trigrams.

Then, I got the following result. The best dev_avg is 4.97% in epoch 8.



```
10-249-158-182:Q2 tanx2$ python q1.py train.txt.lower.unk dev.txt.lower.unk
epoch 1, update 100,featrues 727, train_err 4.01%, dev_err 8.77%, dev_avg 6.43%
epoch 2, update 76,featrues 832, train_err 2.70%, dev_err 6.43%, dev_avg 6.14%
epoch 3, update 39,featrues 879, train_err 1.28%, dev_err 6.73%, dev_avg 6.14%
epoch 4, update 37,featrues 901, train_err 1.17%, dev_err 7.60%, dev_avg 5.56%
epoch 5, update 25,featrues 919, train_err 0.85%, dev_err 6.14%, dev_avg 5.56%
epoch 6, update 19,featrues 929, train_err 0.60%, dev_err 5.85%, dev_avg 5.56%
epoch 7, update 21,featrues 934, train_err 0.63%, dev_err 6.43%, dev_avg 5.26%
epoch 8, update 23,featrues 939, train_err 0.76%, dev_err 6.14%, dev_avg 4.97%
epoch 9, update 16,featrues 939, train_err 0.52%, dev_err 6.14%, dev_avg 5.56%
epoch 10, update 15,featrues 940, train_err 0.46%, dev_err 6.43%, dev_avg 5.56%
train_time 1.64
10-249-158-182:Q2 tanx2$
```

Every template should include t0, I assume because t0 is the tag position the sentence makes mistake, so that we can know which features we need to reward and which features we need to punish.

Moreover, with the best accuracy, I’ve generated dev.lower.unk.best and test.lower.unk.best in folder.