CS134 Project 3: Perceptron POS Tagging

CS134 2018 Fall

Due Date: November 16, 2018

1 Task

For this project, we are doing the POS tagging task. You will design your own features, and implement the perceptron training algorithm and the viterbi decoding algorithm.

2 Data

We will be using the Penn Treebank data. We'll use Penn Treebank section 02∼21 as our training data set, section 22 as the dev data set, and section 23 as the test data set.

Data files hold one sentence per line, and each sentence is represented as follows:

$$word_1\_pos_1 \quad word_2\_pos_2 \quad ..... \quad word_n\_pos_n$$

For training and dev data, you are given the sentences with gold POS tags. For test data, you are only given the plain sentences.

3 Experiments

3.1 Experiment 1: Perceptron V.S. Averaged Perceptron

Using a fixed feature set, experiment with regular perceptron and averaged perceptron algorithms. Report their performances on dev.

3.2 Experiment 2: Plot Learning Curves

Using a fixed feature set, experiment with increasing training data sizes (e.g. {1000, 10000, 25000, all}). Make a plot of accuracy on training data v.s. training size, and a plot of accuracy on dev data v.s. training size, and compare how your tagger's accuracies on training data and dev data change differently with an increasing training data size.

4 Evaluation

To evaluate your tagger's output, run:

python scorer.py [gold file] [auto file]

(e.g. python scorer.py dev/ptb_22.tagged dev/ptb_23.auto_tagged)

5 Requirements

1. Implementation. In appropriate classes/functions/methods in the starter code:
   - Implement a POS tagger using the perceptron training algorithm and the viterbi decoding algorithm. (in perceptron_pos_tagger.py)
   - Design and implement your own features. (in data_structures.py)
   - Implement code to output your auto-tagged data into a file, following the same data format as our training data. (in train_test_tagger.py)
   - * Feel free to add new classes/functions/methods or .py files to the starter code, or modify existing code if necessary.
2. Experiment with different paramter/feature/algorithm setups, as described in Section 3.
3. Write a report:
   - Briefly describe your code structure.
   - Explain your feature designs.
   - Explain your experimental setups (parameters, features, etc.) and results.