Project 3: POS tagging with the Perceptron algorithm

COSI 134 Fall 2019

Due: November 15, 2019

Task definition.

POS tagging is the task of assigning a POS tag to each word token in the sentence.

As with all supervised learning approaches to POS tagging, this task has two parts. The first part is the decoding problem: given a sequence of word tokens and learned model parameters, you are asked to write code that finds the corresponding best tagging sequence. For this you need to implement the Viterbi algorithm to find the best sequence efficiently. The second part is the training process, which takes an annotated training set as input and outputs a set of model parameters. For this, you are asked to implement the Perception algorithm. The Perceptron algorithm is simple yet effective. It has been known to produce results that are at or close to the state of the art. If you implement the learning and decoding algorithms correctly, you will end up with a usable POS tagger that is not far off from the current state of the art.

Data

We will be using the standard train / development / test split in the Penn TreeBank for our experiments: Sections 02-21 are used for training, Section 22 is used for devevelopment, and Section 23 is used as the final test set. You can use the development set to select the features and tune the hyperparameters. When you are done training and tuning, you need to run your code on the test set and produce an automatically tagged version of it.

The data format is very straightforward: each line of the data file contains one sentence. The annotated gold standard data is in the form of $word_1\_pos_1\ word_2\_pos_2\ \cdots\ word_n\_pos_n$ For the training and dev data, you are provided with the sentences with their gold POS tags. For the test set, you are only given the word tokens. The TAs will run your code on the test set to get the accuracy of your model.

Experiments

You are asked to run a number of experiments to help you understand the behavior of the Perceptron algorithm. You can run these experiments on your development set.

Perceptron vs Averaged perceptron

Test your model on the development set using both the Perceptron model and averaged Perceptron model, and observe if there is any difference in performance.

Ablation study on features

Test your model on different feature sets. As we have discussed in class, state of the art models use a five-word window centered on the current word at each position in the sequence:

- `Current word:` $word_0$
- `Previous words:` $word\_1, word\_2$
- `Next words:` $word_1, word_2$
- `Tag of previous word:` pos_1
- `Prefix and suffix of current word`

You are asked to do an ablation study and see how the performance of your tagger changes with different feature sets. When doing your ablation study, first run an experiment with the full set of features. Next take out each feature group at a time and observe the change in accuracy.

Implementation details

- The feature space for the tagging model is the vocabulary size V times the size of the tagset K times the feature window size W. with such a large feature space, if you use a vector representation for the features, it will be very sparse. As such it may not be the most efficient approach. One alternative is to implement the feature weights as a dictionary that is only updated for features that fire for a given training instance.
- The core Perceptron learning algorithm and the Viterbi decoding algorithm should be implemented in `perceptron pos tagger.py`
- The experiments can be encoded in `train test tagger.py` . Make sure you tag the test set with your tagger and write it to a file that has the same format as the training data.
- Feel free to add new classes, functions, methods or modules to the starter code, or modify it as you see fit.

Report

Write a report that a) briefly describe the structure of your code, b) present your experimental settings and results, and c) any insights that you have learned from your experiments. Your report should be no longer than 3 pages.