Maximum Entropy Markov Models

October 3, 2017; due October 21, 2017 (11:59pm)

In this homework, you will be using a first-order maximum entropy Markov model for part-of-speech tagging.

- `memm_tagger.py`: A script to train an MEMM for POS tagging.
- `wsj.pos.train`: Data to train the MEMM with. The data set consists of multiple sentences. Each sentence is a single instance of a sequence whose POS tags we are trying to predict.
- `wsj.pos.dev`: Data to use during training to test the accuracy of a trained MEMM. We will use a separate test data set to evaluate your final model.

During the training phase, a multi-class logistic regression model (from scikit-learn package) is trained to predict the probabilities of each possible label in a sequence for a given set of features. During testing, given the probabilities that we have learned, the most probable sequence of tags needs to be decoded. For each sequence, we iterate through the words to get the probabilities of all labels for each word for each possible preceding label i.e. for every word, we compute L possibilities for L + 1 possible preceding tags. This information is passed to a decoding algorithm to get the most likely sequence for that sentence.

1 Viterbi decoding

The MEMM implemented in `memm_tagger.py` currently uses greedy decoding, conditioning on the single best tag predicted at time step t when estimating the likelihood of a tag at time $t + 1$. Implement the `viterbi_decoding` function in `memm_tagger.py`; like `greedy_decoding`, its sole argument is a $T \times (L + 1) \times L$ tensor M. For a sentence of length T and a label set with L labels, this tensor encodes the probability of label x given label y at time t as $M_{t,y,x}$. (Note that for a first-order MEMM, the conditioning context includes the L labels + the START beginning-of-sentence tag.)

2 Features

The features provided in `memm_tagger.py` are very simple (one feature for the identity of the word at time step t and one feature for the label at time step $t-1$, making it roughly equivalent to a first-order HMM). The power of MEMMs (like many discriminative models) is their ability to encode a rich feature representation of the input, which can be scoped over the entire input sequence $\{x_1, \ldots, x_T\}$ but only over the labels $\{y_{t-1}, y_t\}$, given the first-order Markov assumption. Your primary task for this homework is to create a richer feature representation using the insight into part-of-speech tagging you developed while completing homework 3. Feel free to load external resources to support the features you develop (e.g., a gazetteer of known names, a list of synonyms, pre-trained word embeddings) but you must upload those resources with your homework. Please do not use any libraries to do out-of-box NLP functions (You are free to implement anything yourself!). You will be evaluated here along several dimensions: a.) how much your features improve performance for POS tagging; b.) how creative they are; and c.) how unique they are (i.e., other students didn't think of them). Implement those features in the `get_features` function of `memm_tagger.py` and submit a short description of them as a pdf.

3 Ablation tests

Ablation tests help assess the importance of a feature class on the overall performance of the model by asking: if we train a model with everything *except* a given feature, how much does performance degrade? For each of the features you implement in section 2 above, carry out an ablation test: train a full model on all features and evaluate its performance; then train separate models with each feature ablated in turn and evaluate its performance.

Feature	Ablation
Full model	0.914
Word bigram	-0.043
Word trigram	-0.012
...	...

Table 1: Sample ablation results.

4 Deliverables

- `model.zip` containing `memm_tagger.py` with `viterbi_decoding` and `get_features` both implemented. You are free to add other functions as long as we can run the two options specified in the main function. This folder should also contain other resources you are loading to create any new features.
- A pdf containing a short description of each of the feature classes you implemented, along with the results from an ablation test.