CSCI 404 Natural Language Processing

Winter 2018 - Homework V

Haley Beavers & Robert Harley

Part-of-Speech Tagging with a Hidden Markov Model

Introduction

We implemented the hidden markov model (HMM) where the parameters are built from a given training set. We then tested the performance of this model on a test set it had not seen before.

Method

The HMM was implemented using the viterbi algorithm. In order to improve the speed of the algorithm we used dictionaries in order to avoid checking tags that were not used for a given word. We also calculated the log probabilities in order to prevent underflow.

In order to account for novel words, we implemented the method *add_word_oovs* that will replace all novel words (words not found in our training set) as OOVs and add them accordingly to our emission, tag dictionary (for viterbi), and transmission probabilities.

We implemented backoff smoothing and lambda smoothing as shown in the equations below. These smoothing methods attempt to ameliorate the task of tagging novel words, words that didn't appear in the training set but appear in the test set.

$$P_{tt-backoff}(t_i|t_{i-1}) = p_{t-unsmoothed}(t_i) = \frac{C(t_i)}{n}$$

$$P_{tw-backoff}(w_i|t_i) = p_{w-addone}(w_i) = \frac{C(w_i)+1}{n+V}$$

$$sing_{tt}(\cdot|t_{i-1}) = number\ of\ tag\ types\ t\ such\ that\ c(t_{i-1},\ t) = 1$$

$$sing_{tt}(\cdot|t) = number\ of\ tag\ types\ w\ such\ that\ c(t_i,\ w) = 1$$

$$p_{tt}(t_i|\ t_{i-1}) = \frac{c(t_{i-1},\ t_i) + \lambda \cdot p_{tt-backoff}(t_i|t_{i-1})}{c(t_{i-1}) + \lambda} where\ \lambda = 1 + sing_{tt}(\cdot|t_{i-1})$$

$$p_{tt}(t_i|\ t_{i-1}) = \frac{c(t_i,\ w_i) + \lambda \cdot p_{tw-backoff}(w_i|t_{i-1})}{c(t_i) + \lambda} where\ \lambda = 1 + sing_{tw}(\cdot|t_i)$$

Results

We were able to get good results with our test set, especially when there were no novel words or the sentences were small. Example output of our dataset can be found in results.txt.

Conclusion

The HMM has proven to be an appropriate approach for classifying unknown states based on observed data.

Summary of functions

- **count(filename):** gathers counts for HMM such as initial, transitions, and emissions from training set.
- **add_oov():** adds OOV tags for novel words encountered in test set.
- **viterbi(wrd_seq,counts):** implementation of Viterbi algorithm.
- **test(path):** gets words and tags in given file and puts in list.
- **sing_tw(emissions):** number of singleton words for any given tag.
- **sing_tt(transitions):** number of singleton transitions for any given tag.
- **backoff_tw(w_i,emissions):** implements backoff smoothing given word.
- **backoff_tt(tag_2,emissions):** implements backoff smoothing given tag.
- **count_tag(emissions,ti):** counts number of times tag appears.
- **p_tw(emissions,tag,word):** probability of given emission of word given tag with backoff smoothing, and taking into account singletons.
- **p_tt(emissions,transitions,tag_1,tag_2):** probability of given transition from tag_1 to tag_2 given backoff smoothing, and taking into account singletons.
- **decode(wrd_seq,tags,tags_dict):** finds most likely tag sequence using viterbi algorithm.
- **make_seq(wrd_seq):** separates sentences in given sequence, for formatting.
- **raw(filename):** used to extract vocabulary from rawtrain.txt in list form.