NLP Assignment – 4
POS Tagging

Chetan Kasireddy (201201124)

Hemant Kancharla (201225002)

The aim of the present assignment is to build a POS tagger using the given training data, which can tag a given sentence. The deliverable can be used to build a tagger for any language, given the right training data. The HMM framework is used, and the tagger is based on the Viterbi algorithm. The tagger can also use morph-analysis for handling unseen words.

Working:

We follow the following steps:

- Initialize the data and obtain the training data:

  The given collection of 1000 sentences is taken line by line and split at every white space. We store this data in matrices to use in the Viterbi Algorithm.

- Obtain transition and emission probabilities:

  First, we build the emission and transmission matrices. In the emission matrix, the words are the rows and the tags are the columns, so, in any column all the possible words for the tag are present, and in any row all the possible tags for the word are present.

  The matrix is populated with the counts. Then, the probability is obtained by:

  P(word/tag)=P(word,tag)/P(tag)

  I.e. the ratio of the number of times the word is produced by the tag and the number of occurrences of the tag.

  The transition matrix is a 2D array with the rows being the previous tags and columns being the current tags. The count(x, y) of a pair of tags(where x is the previous tag and y is the current tag) is stored. Probability is obtained using:

  P(tag1,tag2)=P(tag2,tag1)/P(tag1)

- Using the probabilities obtained above, apply Viterbi Algorithm on the test data:

We use the forward Viterbi Algorithm. The probabilities obtained above are used to assign the most probable tag sequence to the test data. The probability is calculated by:

$$\mathbf{V(Word_k,Tag_k) = max(V(Word_{k-1},Tagk\text{-}1) * Transition(Tag_k / Tag_{k-1}) * Emission(Word_k,Tag_k)}$$

A matrix of w*t is constructed where w is the number of words in the sentence and t is the number of tags for each sentence, which contains the present probability of the tag sequence up until that position in the sentence. After obtaining the probabilities for every position using the aforementioned formula, we start from the last word of a sentence, obtain the tag for it with the maximum probability, and backtrack doing the same foe the rest of the words. In this way, the tag sequence is obtained.

- Use morph analysis for unseen words(Model 2):

Many unseen words in the test data are actually different forms of words seen in the training data. To resolve this, we use another model with all the nouns in the training data replaced with their root forms. For replacing, we use SFST to obtain the root form of a word given paradigm tables of all the words(Nouns) in training data in the FST file. We also obtain the case and number of the word.

- Output the tagged test data(tagged taking the most probable tag):

We output the obtained maximum probability tagged sentence for each sentence.

Assumptions:

- For every unknown word, the precedent word's most probable tag(t1) is taken and the tag(t2) into which this tag(t1) undergoes transition with high probability is taken as the tag for the unknown word and the unknown word is given that tag(t2) It is also added to the emission and transition matrix.
- For every unseen word, the emission probabilities are set as $Emission(Word_k,Tag_k) = 1/Count(Tag_k)$
- If the trasition probability is 0, 1/Vocabulary Count is set as the probability.

Observations:

- A broader training data would give a more accurate POS tagger
- In model 2, we get higher accuracies, because there are much lesser unseen words as we have replaced all nouns with root forms
- In telugu, there exist many more cases than direct and oblique, so many difficulties were faced in writing the fst.
- Setting low probability values for unseen words gives higher overall accuracy