Assignment-Discussion HMM Implementation

Goda Nagakalyani, 214050010
Karthika N. J, 21405007
Atul Kumar Singh, 22m0823

4 September,2022

Problem Statement

- Given a sequence of words, produce the POS tag sequence
- Technique to be used: HMM-Viterbi
- Use Universal Tag Set (12 in number)
- 5-fold cross validation
- '.' ,'ADJ' ,'ADP' ,'ADV' ,'CONJ' ,'DET' ,'NOUN' ,'NUM' , 'PRON' , 'PRT' , 'VERB' , 'X'

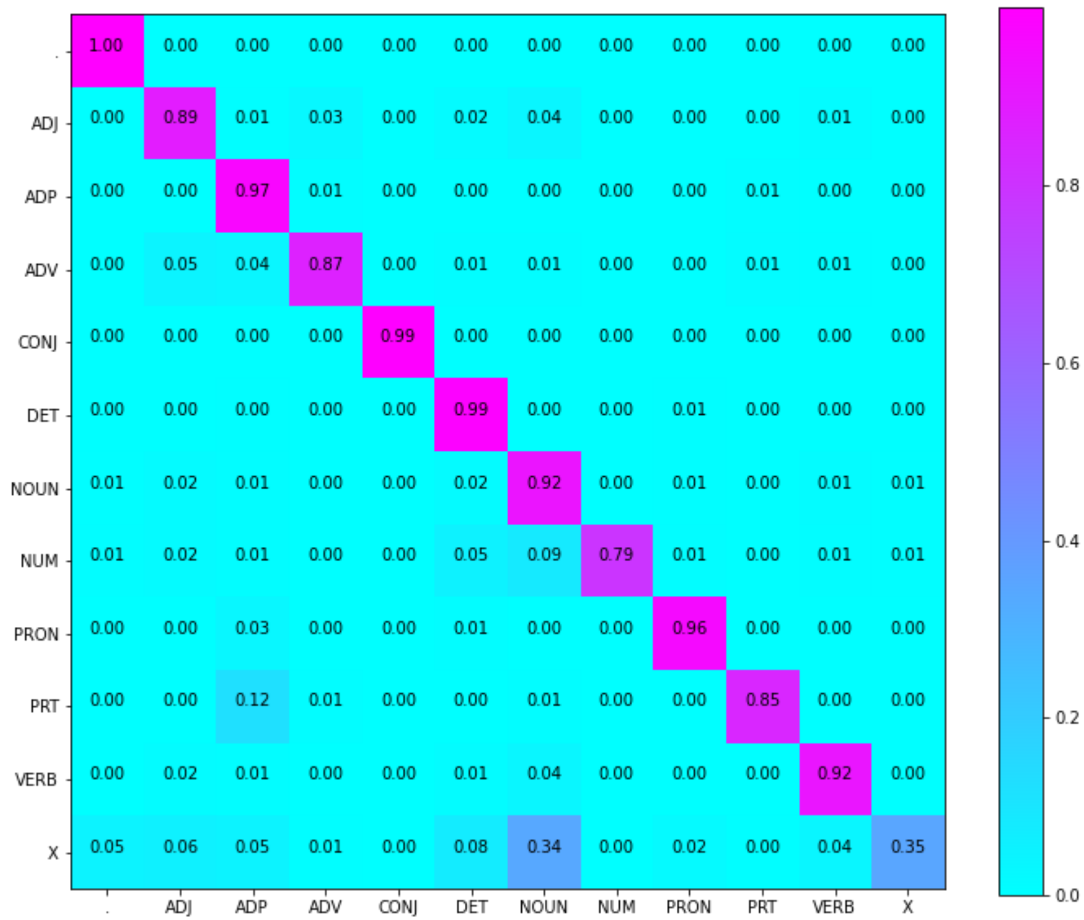
Overall performance

- Precision : 0.940198283308899
- Recall : 0.9385240363559173
- F-score (3 values)
  - F1-score : 0.9386931499010809
  - F0.5-score : 0.9394349573883816
  - F2-score : 0.938428994887414

Per POS performance

- Precision Recall F1-Score
- Tag-1('.'): 0.98, 1.00, 0.99
- Tag-2('ADJ'): 0.87, 0.89, 0.88
- Tag-3('ADP'): 0.92, 0.97, 0.94
- Tag-4('ADV'): 0.90, 0.87, 0.88
- Tag-5('CONJ'): 0.99, 0.99, 0.99
- Tag-6('DET'): 0.92, 0.99, 0.95
- Tag-7('NOUN'): 0.95, 0.92,

Confusion Matrix (12 X 12)



Interpretation of confusion (error analysis)

- The ‘PRT’ tag is most confused with the ‘ADP’ tag. It is clearly visible from confusion matrix that whenever ‘PRT’ was wrongly predicted, it was predicted as ‘ADP’.
- ‘PRT’ tag words: on, at, with, that, over
- ‘ADP’ tag words: on, at, with, by, into, of
- So there are many words which can be used as both ‘PRT’ and ‘ADP’ tags
- The reason can be the similar nature of adpositions and particles which can be understood from definitions.

Data Processing Info (Pre-processing)

- First we preprocessed the data which included converting the text into lower case and adding ‘^’ as start token and ‘$’ as end token in each sentence.
- We created 3 dictionaries:
- POS_tag_counts – stores no of times each tag has occured in training set

word_pos_tags – 2D dictionary with 1 key as word w and 2 key as tag t and stores no of times word w was uses as tag t

Tag_transition_count – 2D dictionary with 1 key as tag1 & 2 key as tag2 and stores no of times tag2 follows tag1

Let w represent word and t, t1, t2 represent tag, tag1, tag2

- P(w | t) = word_pos_tags[w][t] / POS_tag_counts[t]

Inferencing/Decoding Info

- Implemented Viterbi As Follows:

    i) prev dictionary stores P(word|tag) for last seen word (i.e previous level)

    ii) curr dictionary stores P(word|tag) for curr word and for each iteration update it's value accordingly. After compeletion of one level dat of curr is stored in prev.

    iii) parent dictionary helps to backtrack for finding best possible tag for each word in given sentence.

    - backtacking formula :

        final_tags[i] = parent[i+1][final_tags[i+1]] .

    - trying to estimating (i)th level tag due to which (i+1)th level tag has high probabilty.

final_tags is list of tags for given word sequence.

Marking Scheme

- 1. Demo working- 10/10 (if not, 0)
- 2. Implemented Viterbi and Clarity on Viterbi- 5/5
- 3. Transition and Lexical tables clearly described- 5/5
- 4. Confusion matrix drawn and error analysed- 5/5
- 5. Overall F-score > 90- 10/10, >80 & <=90- 8/10, else 6/10
- 6. Unknown word handling- done (5/5; else 0)

Any thoughts on generative vs. discriminative POS tagging

Discriminative POS tagging is Conditional model and therefore it is based on Conditional probability. It needs labelled data, and so it can be used only in supervised learning.

Generative POS tagging is a Joint model and therefore it is based on Joint probability. It can be used in unsupervised learning.