POS Tagging Using HMM

CS 626

Ashwin Ramachandran (200050016)
Hardik Rajpal (200050048)
Anustup Bhattacharyya (22D0714)

Team ID: 18

September 4, 2022

Table of Contents

Problem Statement

Problem Statement

- Point A: Given a sequence of words, produce the POS tag sequence
- Point B: Technique to be used: HMM-Viterbi
- Point C: Universal Tag Set (12 in number)
- Point D: 5-fold cross-validation
- Point E: Universal Tagset

1. .
2. ADJ
3. ADP
4. ADV
5. CONJ
6. DET
7. NOUN
8. NUM
9. PRON
10. PRT
11. VERB
12. X

Overall Performance

Overall Performance

The overall precision and recall are calculated as frequency-weighted averages of per-POS precision and recall values. The F-β scores are calculated by using the standard formula on these weighted averages.

Metric	Value
Precision	93.77
Recall	91.50
F-0.5	93.31
F-1	92.62
F-2	91.94

Per POS Performance

Recall, Precision and F1 Scores

The values below are unweighted averages of the five runs of the model on the corpus (using different fifths of the corpus for testing).

POS Tag	Precision	Recall	F1 Score
.	97.65	99.99	98.81
ADJ	86.51	88.29	87.39
ADP	92.58	97.24	94.85
ADV	90.03	86.92	88.45
CONJ	98.83	99.28	99.05
DET	93.79	98.71	96.19
NOUN	94.44	92.52	93.47
NUM	94.13	92.45	93.28
PRON	91.82	95.84	93.79
PRT	91.70	84.93	88.19
VERB	97.14	91.35	94.16
X	89.26	15.22	25.88

Table: Per-POS Performance Results

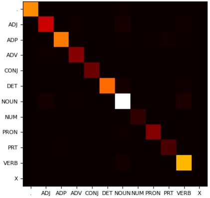
Per-POS Analysis

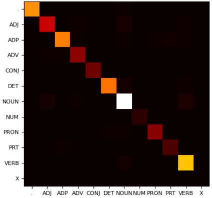
We see that most tags have good precision and recall values (> 80) except the unknown marker 'X', which seems to have low recall and consequently a low F1 score. This is expected as the model has no way of learning from its past experiences with unknown markers, hence the low recall.

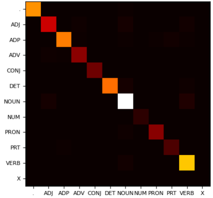
Confusion Matrix and Analysis

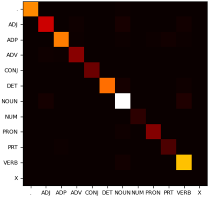
Confusion Matrix

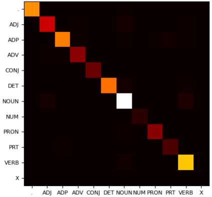
Given below are the confusion matrix heat maps for with the test sets as different fifths of the corpus.











Confusion Matrix Analysis

- The brighter spots (high values) on the diagonal indicate a considerably higher frequency of correct predictions (true positives) by the model than incorrect ones. We note that the model is particularly bright at predicting nouns correctly.
- Verbs and determiners follow behind closely. This is inline with the corpus having a high frequency of these three tags, and in predictable patterns.
- We also see that unknown tag marker 'X' has very few true positives, as compared to the other tags. This is also expected as the model is much more likely to observe an unknown word and classify it as one of the other, more frequent tags, based on the usage of a separate marker for unknown words.

Data Processing Info (Pre-processing)

Data (Pre) Processing Info

During training, we first pre-process the sentences to make all words lowercase, substitute amt for numerical values and qnw for alpha-numeric-symbolic values. We then extract all unique words and tags and store them in two arrays, `words` and `tags`. We append a marker '*' for unknown words to the `words`. We also construct index-dictionaries for the two arrays, namely `wordinds` and `taginds`. A frequency map is also generated for the unique words.

We then replace all words and tags in the training sentences with their corresponding indices in the dictionaries. We initialize three `numpy` arrays, one for storing transmission frequencies ($|\text{tags}| \times |\text{tags}|$), one for emission frequencies ($|\text{words}| \times |\text{tags}|$) and one for initial probabilities of the tags ($|\text{tags}| \times 1$), all with 1's.

Then, we simply iterate over the sentences, filling the three arrays as we encounter the pair of indices of `(wi,ti)`. If we encounter a word whose frequency is less than 2 (a low threshold), we use its tag to populate the emission frequencies of the unknown word marker, '*'.

Inferencing/Decoding

Viterbi Implementation

We are provided with a hidden markov model with state space S, initial probabilities π_i of being in state i, transition probabilities $trellis_{i,j}$ of transitioning from state s_i to state s_j, observation $y_1, y_2, ..., y_T$ and probabilities $emmis_{i,j}$ of word w_i occurring given the state s_j. We find the most likely sequence $x_1, x_2, ..., x_T$ using the following recurrence relation

$$V_{1,k} = log(P(y_1|k)) + log(\pi_k)$$

$$V_{t,k} = max_{x \in S}(log(P(y_t|k)) + log(trellis_{x,k}) + log(V_{t-1,x}))$$

Here $V_{t,k}$ is the probability of the most probable state sequence $P(x_1, .., x_t, y_1, .., y_t)$ responsible for the first t observations that have k as its final state. We also simultaneously store back pointers to each state x used in the second equation in order to obtain the complete state sequence path. In our implementation, we replace probability product computation with summation of it's logarithms in order to avoid floating point underflow.

Thank you!