POS Tagging Using Hidden Markov Models and Maximum Entropy Markov Models

Soumyodeep Dey

[ELL 884 ASSIGNMENT 1]

0.1 Introduction to Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical model used to represent systems where hidden states influence observable outcomes.

- There are a set of hidden states, denoted as $S = \{s_1, s_2, \ldots, s_N\}$.
- Each state emits observable symbols, denoted as $O = \{o_1, o_2, \ldots, o_M\}$.
- Transitions between states occur probabilistically, governed by a transition probability matrix A, where $A_{ij} = P(s_j|s_i)$ is the probability of transitioning from state i to state j.
- Each state emits symbols probabilistically, defined by an emission probability matrix B, where $B_{ij} = P(o_j|s_i)$ is the probability of emitting symbol o_j from state i.

The key assumption in HMMs is the Markov property, which states that the probability of the current state depends only on the previous state and is independent of any previous history:

$$P(s_t|s_{t-1}, s_{t-2}, \ldots, s_1, o_1, o_2, \ldots, o_{t-1}) = P(s_t|s_{t-1}) \tag{1}$$

0.2 POS Tagging with HMMs

1. States: Each state represents a possible POS tag

2. Observations: Words in the sentence represent the observations.

3. Transition probabilities: $A_{ij} = P(s_j|s_i)$ captures the probability of transitioning from state i (one POS tag) to state j (another POS tag).

4. Emission probabilities: $B_{ik} = P(o_k|s_i)$ represents the probability of emitting observation k (a word) from state i (a POS tag).

Viterbi Algorithm:

$$
\begin{aligned}
M_{ti} &= \max_{j=1}^{N}(V_{(t-1)j}A_{ji})B_{io_t} \\
\delta_{ti} &= \arg\max_{j=1}^{N}(V_{(t-1)j}A_{ji}) \\
\hat{q}_t &= \arg\max_{i} M_{ti} \\
\hat{q}_1, \ldots, \hat{q}_n &= \text{Backtrack using } \delta_{ti}
\end{aligned}
$$

M_{ti}: highest probability of observing first t words and ending in state i.
δ_{ti}: previous state maximizing M_{ti}.
$\hat{q}_t$: most likely POS tag for word t.

0.3 POS Tagging with MEMMs

Common probabilities: $A_{ij} = P(s_j|s_i, word_j)$ captures the probability of a given state j given that the previous state was i and the current word is j.

Viterbi Algorithm:

$$
\begin{aligned}
M_{ti} &= \max_{j=1}^{N} V_{(t-1)j}A_{ji} \\
\delta_{ti} &= \arg\max_{j=1}^{N} V_{(t-1)j}A_{ji} \\
\hat{q}_t &= \arg\max_{i} M_{ti} \\
\hat{q}_1, \ldots, \hat{q}_n &= \text{Backtrack using } \delta_{ti}
\end{aligned}
$$

M_{ti}: highest probability of observing first t words and ending in state i.
δ_{ti}: previous state maximizing M_{ti}.
$\hat{q}_t$: most likely POS tag for word t.

0.4 Evaluation and Comparison

1. My first few attempts to submission, I was doing the wrong modelling of the problem and I was calculating the likelihood of the tag sequence given my observed variables (words). But this was the wrong approach. This implementation did not do very good on the test set despite zero error on the

training corpus.

2. When I correctly implemented the viterbi algorithm along with backpointer, I had two options to choose from to deal with the OOV words.
Approach 1: To replace any OOV word with the single most frequent word in corpus. The vote went to 'NN'.
Approach 2: To statistically assign probabilities to each tag and randomly assign when we encounter any OOV word.

3. Approach 1 gave better score as compared to approach 2
Approach 1: 0.766 Approach 2: 0.763

4. I also tried doing Laplacian Smoothing on transition probability matrices or emission probability matrix or the common probability matrix but it resulted in a few errors in the training set and also a lower accuracy on submission (0.693).

5. My highest score (0.808) has come from the correct implementation of POS tagging using HMM and assigning the OOV words as 'NN'.

0.5 Implementations

HMM: POS Tagging Implementation

MEMM: MEMM Tagging Implementation

0.6 Conclusion

1. One of the techniques I thought of implementing while assigning tags for OOV words was doing rule-based tagging. Since the OOV words formed only less than 2 percent of all the words in the vocabulary, I didn't find it worthy enough to spend time on this idea.

2. I found the idea of implementing HMM but assuming second order Markov independence. Where tag assigned to current word will not only depend on previous tag assigned but also the previous's previous tag.

0.7 References

1. Penn Treebank 2. Medium article 1 3. Medium article 2