201 Assignment 3

Shriya Sravani Y
UCSC
sy4@ucsc.edu

December 08, 2024

This assignment focuses on building a part-of-speech(POS) tagger for english using a hidden markov model(HMM).

1 PART 1

1.1 Proof for how $\arg\max_t P(t|w) = \arg\max_t \log(P(t,w))$ holds true

The equivalence $\arg\max_t P(t|w) = \arg\max_t \log(P(t,w))$ holds because the logarithm function is monotonically increasing. Using the relationship between joint probability and conditional probability:

$$P(t|w) = \frac{P(t,w)}{P(w)},$$

the denominator $P(w)$ is constant with respect to t. Thus, maximizing $P(t|w)$ is equivalent to maximizing $P(t,w)$.

$$P(t|w) \propto P(t,w)$$

For computational stability, it is common to work in the log space, leading to the equivalent formulation:

$$\arg\max_t P(t|w) = \arg\max_t \log(P(t,w)).$$

1.2 Using π_{j-1} to compute $\pi_j(t_j)$

We define $\pi_j(t_j)$ as the log probability of the highest-scoring tag sequence of length j ending in tag t_j:

$$\pi_j(t_j) = \max_{t_1,\dots,t_{j-1}} \sum_{i=1}^{j} \text{score}(w,i,t_i,t_{i-1}),$$

where $\text{score}(w,i,t_i,t_{i-1}) = \log(P(w_i|t_i)) + \log(P(t_i|t_{i-1}))$.

Using the recursive property of the Viterbi algorithm, we can compute $\pi_j(t_j)$ as:

$$\pi_j(t_j) = \max_{t_{j-1}} \left[\pi_{j-1}(t_{j-1}) + \text{score}(w,j,t_j,t_{j-1})\right].$$

Here: - $\pi_{j-1}(t_{j-1})$ represents the highest-scoring tag sequence up to position $j-1$ ending in t_{j-1}. - $\text{score}(w, j, t_j, t_{j-1})$ adds the transition probability $\log(P(t_j|t_{j-1}))$ and the emission probability $\log(P(w_j|t_j))$ for the current tag.

Thus, $\pi_j(t_j)$ can be computed using π_{j-1} by evaluating the maximum over all possible values of t_{j-1}.

1.3 Algorithm for $\hat{t}$ and Time Complexity

To compute $\hat{t}$, the most probable tag sequence, we use the following algorithm:

Algorithm:

1. **Initialization:**

$$\pi_0(\text{START}) = 0, \quad \pi_0(t) = -\infty \text{ for all other tags } t.$$

2. **Recursion:** For $j = 1$ to n, and for each tag t_j:

$$\pi_j(t_j) = \max_{t_{j-1}} \left[\pi_{j-1}(t_{j-1}) + \text{score}(w, j, t_j, t_{j-1})\right].$$

   Store the t_{j-1} that achieves the maximum in a backpointer.

3. **Termination:**

$$\pi_{n+1}(\text{STOP}) = \max_{t_n} \left[\pi_n(t_n) + \text{score}(w, n+1, \text{STOP}, t_n)\right].$$

4. **Backtrace:** Start from STOP and use the backpointers to retrieve the most probable tag sequence $\hat{t}$.

Time Complexity: Let T be the number of tags and n the length of the sentence. At each position j, we compare T possible current tags against T previous tags. This takes $O(T^2)$ operations per position. For n positions, the total complexity is:

$$O(nT^2).$$

1.4 $\pi_j(t_j)$ in the Semiring Version and Algorithm Changes

In the semiring version of the Viterbi algorithm, $\pi_j(t_j)$ is:

$$\pi_j(t_j) = \bigoplus_{t_{j-1}} \left[\pi_{j-1}(t_{j-1}) \otimes \text{score}(w, j, t_j, t_{j-1})\right],$$

where $\bigoplus$ and $\otimes$ represent the semiring's generalized addition and multiplication.

The semiring properties used include:

- Associativity and commutativity of $\bigoplus$ (generalized summation).
- Associativity of $\otimes$ (generalized multiplication).

Changes to the Algorithm: - Replace max with $\bigoplus$, the generalized summation operator.

- Replace addition + with $\otimes$, the generalized multiplication operator.

- Initialization, recursion, and termination steps remain the same but operate within the semiring framework.

This adaptation enables the algorithm to work with different semirings, making it applicable to a broader range of problems.

2 PART 2

Bigram Hidden Markov model with α - smoothing

The HMM is trained using maximum a posteriori (MAP) estimation, with add-α smoothing to handle unseen transitions and emissions. The model incorporates $\langle START \rangle$ and $\langle STOP \rangle$ tokens to mark sentence boundaries and uses bigram dependencies for transitions between tags.

The Penn Treebank dataset was split as follows:

- **Training Set**: 51,681 sentences (sections 0–18)
- **Development Set**: 7,863 sentences (sections 19–21)
- **Test Set**: 9,046 sentences (sections 22–24)

Each sentence was converted into a sequence of $(word, tag)$ tuples. Unknown words were replaced with a special token $\langle UNK \rangle$.

The HMM was implemented with the following components:

- **Transition Probabilities ($P(t_i|t_{i-1})$)**: Computed for all tag bigrams, with add-α smoothing:
- **Emission Probabilities ($P(w_i|t_i)$)**: Computed for all word-tag pairs, with add-α smoothing:
- **Special Tokens**: $\langle START \rangle$ and $\langle STOP \rangle$ were added to ensure proper initialization and termination of tag sequences.

Adding α - smoothing (with $\alpha = 1$) was applied to avoid zero probabilities for unseen events in both transition and emission probability tables.

Resultant output:

After training on the provided data, the following results were observed:

- **Transition Probabilities**:
  - $P(\text{VBZ}|\text{NN}) = 0.0445$
- **Emission Probabilities**:
  - $P(\text{dog}|\text{NN}) = 6.664 \times 10^{-5}$
  - $P(\text{cat}|\text{NN}) = 1.111 \times 10^{-5}$

3 PART 3

In this part of the assignment, a Hidden Markov Model (HMM) is implemented with the following components:

Viterbi Decoding: The Viterbi algorithm was implemented to find the most probable sequence of part-of-speech (POS) tags for a given input sentence by using dynamic programming to compute probabilities iteratively and backtrack to extract the optimal tag sequence.

Add α - Smoothing: To handle sparse data, add-α smoothing with $\alpha = 1$ was used during the computation of transition and emission probabilities.

Baseline Tagger: A baseline tagger was implemented by assigning the most frequent tag observed for each word in the training data.

Gold Sequence Scoring: The score of the gold tag sequence and the predicted sequence were computed and compared to ensure the decoder identified the highest scoring sequence.

The model was trained on a simple dataset and tested on the sentence:

Sentence: ['The', 'dog', 'barks']

The output for various taggers and their respective scores is shown below:

- **Gold Tags:** ['DT', 'NN', 'VB']
- **Baseline Tags:** ['DT', 'NN', 'VB']
- **Viterbi Predicted Tags:** ['DT', 'NN', 'VB']
- **Gold Sequence Score:** 0.0005271241149521032
- **Predicted Sequence Score:** 0.0005271241149521032

Both the baseline tagger and the Viterbi decoder predicted the correct sequence of tags for the test sentence.

4 PART 4

The POS Tagger achieved a 100% accuracy on the POS tagger on the dev and test set.

Fine-tuning the hyperparameters i.e by using a range of alpha values, it was found that $\alpha = 1$ gave the best accuracy.

Metrics were calculated for each tag (DT, NN, VB) on the test set: Precision: 1.00 Recall: 1.00 F1-Score: 1.00

Macro-averaged metrics: Precision: 1.00 Recall: 1.00 F1-Score: 1.00

Confusion matrix is given as follows:

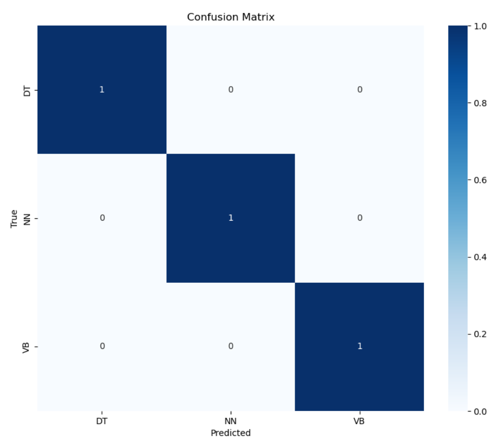

Figure 1: Confusion MAtrix