nlp201 hw3

Yifei Gan

December 2024

1

This equivalence is due to the monotonicity of the logarithmic function. The logarithm is a strictly increasing function, so maximizing P(t,w) is equivalent to maximizing log(P(t,w)).

2

Define $\pi_j(t_j)$ as: $\pi_j(t_j) = \max_{t_1,\ldots,t_{j-1}} \sum_{i=1}^{j} \text{score}(w, i, t_i, t_{i-1})$.

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

where: $\text{score}(w, j, t_j, t_{j-1}) = \log(P(w_j \mid t_j)) + \log(P(t_j \mid t_{j-1}))$.
$\pi_{j-1}(t_{j-1})$ represents the maximum score for the tag sequence ending in t_{j-1} at step $j-1$.
The current score $\text{score}(w, j, t_j, t_{j-1})$ includes the emission probability $\log(P(w_j \mid t_j))$ and the transition probability $\log(P(t_j \mid t_{j-1}))$.
Thus, $\pi_j(t_j)$ depends only on $\pi_{j-1}(t_{j-1})$ and the scores of the current step.

3

Initialization:

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

Recursion:

For each $j = 1, \ldots, n$ (word positions) and 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 backpointer: $\text{bp}_j(t_j) = \arg\max_{t_{j-1}} \left[\pi_{j-1}(t_{j-1}) + \text{score}(w, j, t_j, t_{j-1})\right]$.

Termination:

Compute the final score including the STOP tag: $\pi_{n+1}(\text{STOP}) = \max_{t_n} \left[\pi_n(t_n) + \text{score}(w, n+1, \text{STOP}, t_n)\right]$.
Store the final backpointer: $\text{bp}_{n+1}(\text{STOP}) = \arg\max_{t_n} \left[\pi_n(t_n) + \text{score}(w, n+1, \text{STOP}, t_n)\right]$.

Backtracking:

Recover the best sequence by tracing back from $\mathrm{bp}_{n+1}(\mathrm{STOP})$ to $\mathrm{bp}_0(\mathrm{START})$.

Time Complexity:

At each step j, we compute $\pi_j(t_j)$ for all tags t_j, requiring a maximization over all possible t_{j-1}. Let T denote the number of tags:
Per step: $O(T^2)$, as we evaluate all $T \times T$ pairs of current and previous tags.
Total for n words: $O(n \cdot T^2)$.
The time complexity is $O(n \cdot T^2)$.

4

From the definition of $\pi_j(t_j)$: $\pi_j(t_j) = \bigoplus_{t_{j-1}} \left[\pi_{j-1}(t_{j-1}) \otimes \mathrm{score}(w, j, t_j, t_{j-1})\right]$.
This recursive formulation expresses $\pi_j(t_j)$ in terms of $\pi_{j-1}(t_{j-1})$. Because of the computation using the semiring properties, the recursion relies on the above semiring properties to generalize the computation of $\pi_j(t_j)$.
To adapt the algorithm from question 2 for the semiring version:

1. Replace the standard max operation with $\oplus$, which represents summation in the semiring.

2. Replace the addition $(+)$ operation with $\otimes$, which represents multiplication in the semiring.

3. Initialize $\pi_0(\mathrm{START}) = 1_s$ (multiplicative identity) and $\pi_0(t) = 0_s$ (additive identity) for all other tags.

4. Use $\oplus$ and $\otimes$ for recursive updates:

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

5. Terminate with:

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

Conclusion

The semiring properties ensure that the algorithm is valid for any semiring $S = \langle A, \oplus, \otimes, 0_s, 1_s \rangle$. The changes to the standard Viterbi algorithm involve replacing max and $+$ with $\oplus$ and $\otimes$, making the algorithm compatible with a generalized scoring framework.