

Viterbi Algorithm in NLP

```
# States and observations
states = ['Noun', 'Verb']
observations = ['I', 'run']

# Start probability
start_prob = {'Noun': 0.6, 'Verb': 0.4}

# Transition probability
trans_prob = {
    'Noun': {'Noun': 0.3, 'Verb': 0.7},
    'Verb': {'Noun': 0.8, 'Verb': 0.2}
}

# Emission probability
emit_prob = {
    'Noun': {'I': 0.5, 'run': 0.1},
    'Verb': {'I': 0.1, 'run': 0.7}
}

def viterbi(obs, states, start_p, trans_p, emit_p):
    V = [{}] # Probability table
    path = {} # Path table

    # Initialize base case
    for s in states:
        V[0][s] = start_p[s] * emit_p[s].get(obs[0], 0)
        path[s] = [s]

    # Run Viterbi for t > 0
    for t in range(1, len(obs)):
        V.append({})
        new_path = {}

        for s in states:
            (prob, state) = max(
                (V[t-1][s0] * trans_p[s0][s] * emit_p[s].get(obs[t],
0), s0)
                for s0 in states
            )
            V[t][s] = prob
            new_path[s] = path[state] + [s]

        path = new_path

    # Find the final most probable state
    n = len(obs) - 1
    (prob, state) = max((V[n][s], s) for s in states)
    return (prob, path[state])
```

```
prob, pos_sequence = viterbi(observations, states, start_prob,  
trans_prob, emit_prob)  
print(f"Probability: {prob}")  
print(f"POS tags: {pos_sequence}")
```

Probability: 0.147

POS tags: ['Noun', 'Verb']