Lecture 6 Companion: Part-of-Speech Tagging

Hidden Markov Models & The Viterbi Algorithm

NLP Course Companion

Contents

1 Introduction: The Problem of Ambiguity

Part-of-Speech (POS) tagging is the process of assigning a lexical class (Noun, Verb, Adjective, etc.) to every word in a text.

Why is this hard? Because English is ambiguous.

- "*I can **can** the **can**.*"
- Here, the word "can" appears three times but acts as a Modal Verb, a Main Verb, and a Noun.

A POS Tagger must look at the **context** to decide which tag is correct. This is the first step for many downstream tasks like Parsing, Named Entity Recognition, and Machine Translation.

2 The Tagset

We need a standard list of tags. The most common is the **Penn Treebank Tagset** (45 tags).

Tag	**Meaning**	**Example**
NN	Noun, singular	*dog, rain*
NNS	Noun, plural	*dogs, clouds*
VB	Verb, base form	*eat*
VBZ	Verb, 3rd person singular	*eats*
DT	Determiner	*the, a*
JJ	Adjective	*yellow*

3 Hidden Markov Models (HMM)

To solve tagging probabilistically, we use a **Hidden Markov Model**.

Intuition: The Ice Cream & Weather Analogy

Imagine you are a climatologist in the future. You want to know the weather (Hot or Cold) from the year 2007, but all records are lost.

- **Hidden State (The Weather):** You cannot see this directly.
- **Observation (Ice Cream):** You find a diary listing how many ice creams Jason ate each day (1, 2, or 3).

If Jason ate 3 ice creams, it was probably Hot. If he ate 1, it was probably Cold. **In NLP:**

- **Hidden States = POS Tags** (We want to find these).
- **Observations = Words** (We see these).

3.1 The Two Probabilities of HMM

An HMM relies on two matrices to make predictions:

1. Transition Probabilities (A): $P(t_i|t_{i-1})$

- The probability of a tag following another tag.
- Example: A Noun (NN) is very likely to follow a Determiner (DT), like "the **cat**".
- Matrix A is size $N \times N$ (where N is number of tags).

2. Emission Probabilities (B): $P(w_i|t_i)$

- The probability of a word appearing given a tag.
- Example: If the tag is VBZ (verb 3rd person), "is" is very likely, "race" is less likely.
- Matrix B is size $N \times V$ (Tags $\times$ Vocabulary).

4 The Viterbi Algorithm: A Step-by-Step Walkthrough

We want to find the sequence of tags that maximizes the probability for a sequence. We cannot check every combination (that would be exponential). Instead, we use **Viterbi**, a dynamic programming algorithm.

4.1 Scenario Setup: The Ice Cream Task

We will use the specific numbers from the lecture slides (Slide 55).

- **Observation Sequence:** 3, 1 (Jason ate 3 ice creams Day 1, 1 ice cream Day 2).
- **Hidden States:** H (Hot), C (Cold).
- **Goal:** Find the weather sequence (e.g., H H, or H C).

Model Parameters:

- **Start Probs (π):** $P(H) = 0.8$, $P(C) = 0.2$.
- **Transitions (A):** $P(H|H) = 0.6, \quad P(C|H) = 0.4$ $P(H|C) = 0.5, \quad P(C|C) = 0.5$
- **Emissions (B):** $P(3|H) = 0.4, \quad P(1|H) = 0.2$ $P(3|C) = 0.1, \quad P(1|C) = 0.5$

4.2 Step 1: Initialization (Day 1: Obs "3")

We calculate the probability of starting Hot vs Cold given we saw 3 ice creams.

Formula

$$V[t, 1] = P(t|\text{Start}) \times P(\text{obs}_1|t)$$

Calculations:

- **Path to Hot ($v_1(H)$):**
$$0.8(\text{Start}) \times 0.4(P(3|H)) = \mathbf{0.32}$$
- **Path to Cold ($v_1(C)$):**
$$0.2(\text{Start}) \times 0.1(P(3|C)) = \mathbf{0.02}$$

Winner for Day 1: Hot (0.32).

4.3 Step 2: Recursion (Day 2: Obs "1")

Now we calculate the score for Day 2 being Hot or Cold, coming from the best path of Day 1.

Formula

$$V[t,i] = \max_{prev}(V[prev,i-1] \times P(t|prev)) \times P(\text{obs}_i|t)$$

1. Take previous path score. 2. Multiply by Transition (Previous → Current). 3. Multiply by Emission (Current Tag → Current Obs).

Option A: Day 2 is HOT We could have come from Hot (Day 1) or Cold (Day 1).

- From H: $0.32 \times 0.6(H \to H) \times 0.2(P(1|H)) = \mathbf{0.0384}$
- From C: $0.02 \times 0.5(C \to H) \times 0.2(P(1|H)) = 0.002$

Best path to Hot(2) is from Hot(1). Score: 0.0384.

Option B: Day 2 is COLD

- From H: $0.32 \times 0.4(H \to C) \times 0.5(P(1|C)) = \mathbf{0.064}$
- From C: $0.02 \times 0.5(C \to C) \times 0.5(P(1|C)) = 0.005$

Best path to Cold(2) is from Hot(1). Score: 0.064.

4.4 Step 3: Termination & Backtrace

We compare the final scores for Day 2:

- Score ending in Hot: 0.0384
- Score ending in Cold: **0.064 (WINNER)**

We backtrack from the winner. 1. The winner was **Cold(2)**. 2. The best path to Cold(2) came from **Hot(1)**. 3. The best path to Hot(1) was Start.

Result: The most likely weather sequence is **HOT → COLD**.

5 Maximum Entropy Markov Models (MEMM)

HMMs are **Generative Models**: they calculate $P(w|t)$. They try to model how the data was created. **MEMMs** are **Discriminative Models**: they calculate $P(t|w)$ directly.

5.1 Why switch to MEMM?

HMMs are limited. They only look at the current tag and the current word. MEMMs can look at **Features**. When tagging "race", an MEMM can look at:

- Is the previous word "the"? (Likely Noun).
- Is the previous word "to"? (Likely Verb).
- Does the word end in "-ing"?
- Is the word capitalized?

The Mathematical Shift

HMM (Bayes Rule):

$$\hat{t} = \text{argmax} \prod P(w_i|t_i)P(t_i|t_{i-1})$$

MEMM (Logistic Regression):

$$\hat{t} = \text{argmax} \prod P(t_i|w_i, t_{i-1}, \text{features})$$

6 Summary

- **POS Tagging** resolves ambiguity in language.
- **HMMs** use Transition and Emission probabilities to find the most likely sequence.
- The **Viterbi Algorithm** efficiently calculates the best path using dynamic programming ($O(N^2 \cdot T)$ complexity instead of exponential).
- **MEMMs** improve on HMMs by allowing rich feature extraction (like capitalization and suffixes) to discriminate between tags.