Faculty of Computer Science, Dalhousie University *26-Oct-2023*

CSCI 4152/6509 — Natural Language Processing

Lecture 16: Efficient Inference with HMM

Location: Rowe 1011 Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- Reminders: no Lab 7, A2, P1
- **POS tagging: Introduction**
- Reading: [JM] Ch5 Part-of-Speech Tagging
- Open word categories
- Closed word categories
- Other word categories
- **Hidden Markov Model (HMM):**
  - idea, definition, graphical representation
  - HMM assumption

15.2 POS Tagging using HMM

We will examine now Hidden Markov Model (HMM) in more details, including computational tasks in this moden on the example of POS tagging application.

HMM use in POS Tagging

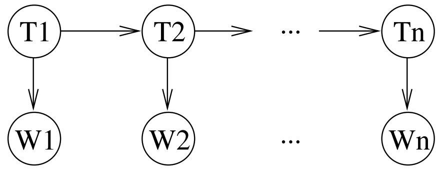
- Hidden states = POS Tags
- Observable variables = words
- In practice: higher-order HMM taggers are used, where the nodes keep a bit longer history (e.g., two previous tags)
- Described in [JM] Sec 5.5 (HMM POS Tagging)

Computational Tasks for HMM

- Evaluation: use HMM assumption formula
- Generation: generate in the order dictated by the "unrolled" graphical representation
- Inference:
  - marginalization, conditioning, completion
  - need for an efficient method (will discuss it)
- Learning: MLE if labeled examples are given

HMM POS Example

To understand better issues involved in efficient HMM inference, we will use a very small, walk-through example in POS tagging. We assume that the hidden internal states of the HMM correspond to correct POS tags of words, while the words correspond to generated observed variables. According to this, a sentence of n words would be associated with the following HMM graph in the unrolled form:



The variables $W_1, \ldots, W_n$ are assigned to words in the sentence, while variables $T_1, \ldots, T_n$ are assigned POS tags. The three probability tables that we mentioned in the definition of HMM are: $\mathrm{P}(T_1)$, $\mathrm{P}(T_{i+1}|T_i)$, and $\mathrm{P}(W_i|T_i)$

Having this in mind, suppose that we are given the following training data:

```
swat V flies N like P ants N
time N flies V like P an D arrow N
```

To accommodate for unseen words, we can assign a special symbol $\star$ to unknown words, and assume that it occurred 0.5 "times" with each tag.

First, we can "learn" the probability of initial states π ($= \mathrm{P}(T_1)$ by counting how many times each state was the first state in a sequence, and obtain the following counts and the resulting probabilities:

q	counts for $\pi(q)$
D	0
N	1
P	0
V	1

$\Rightarrow$

q	$\pi(q)$
D	0
N	0.5
P	0
V	0.5

We will also denote the initial probability $\pi(q)$ as $\pi(q) = \mathrm{P}(T_1 = q)$.

Similarly, we count the transitions in order to estimate transition probabilities a:

counts for $a(p,q)$	D	N	P	V	sum
D	0	1	0	0	1
N	0	0	1	1	2
P	1	1	0	0	2
V	0	1	1	0	2

$\Rightarrow$

$a(p,q)$	D	N	P	V
D	0	1	0	0
N	0	0	0.5	0.5
P	0.5	0.5	0	0
V	0	0.5	0.5	0

We will also denote the transitional probability $a(p,q)$ as $a(p,q) = PP(T_{i+1} = q|T_i = p)$, which more clearly presents meaning of this probability table.

We will incorporate our smoothing method into learning of the output probabilities by giving a count of 0.5 to any unseen words, marked with '*', to be generated from any tag. This is how we obtain the following counts:

counts for $b(q,o)$	an	ants	arrow	flies	like	swat	time	*	sum
D	1	0	0	0	0	0	0	0.5	1.5
N	0	1	1	1	0	0	1	0.5	4.5
P	0	0	0	0	2	0	0	0.5	2.5
V	0	0	0	1	0	1	0	0.5	2.5

which leads to the following estimated probabilities obtained by dividing numbers in each row with the corresponding sum:

$b(q,o)$	an	ants	arrow	flies	like	swat	time	*
D	$2/3$	0	0	0	0	0	0	$1/3$
N	0	$2/9$	$2/9$	$2/9$	0	0	$2/9$	$1/9$
P	0	0	0	0	$4/5$	0	0	$1/5$
V	0	0	0	$2/5$	0	$2/5$	0	$1/5$

Another way to represent the learned table is as the following conditional probability tables (CPTs):

Generated Tables

T_1	$\mathrm{P}(T_1)$
N	0.5
V	0.5

,

T_{i-1}	T_i	$\mathrm{P}(T_i \mid T_{i-1})$
D	N	1
N	P	0.5
N	V	0.5
P	D	0.5
P	N	0.5
V	N	0.5
V	P	0.5

and

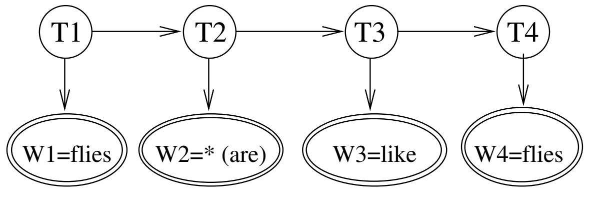
T_i	W_i	$\mathrm{P}(W_i \mid T_i)$
D	an	$2/3 \approx 0.666666667$
D	$\star$	$1/3 \approx 0.333333333$
N	ants	$2/9 \approx 0.222222222$
N	arrow	$2/9 \approx 0.222222222$
N	flies	$2/9 \approx 0.222222222$
N	time	$2/9 \approx 0.222222222$
N	$\star$	$1/9 \approx 0.111111111$
P	like	0.8
P	$\star$	0.2
V	flies	0.4
V	swat	0.4
V	$\star$	0.2

.

In the tables above we did not include zero-probabilities: for example, $\mathrm{P}(T_i = V | T_{i-1} = D)$ is not included since it is equal to 0. The symbol '*' is used to denote any unseen word that may appear in a testing sentence.

Reminder: Generated Tables

Let us use the Hidden Markov Model to POS tag the sentence "flies are like flies."



The problem of POS tagging in this case is the problem of finding the most probable values of the variables T_i given the values of variables W_i, i.e., it is the completion problem of finding

$$\begin{aligned}\arg\max_T \mathrm{P}(T|W = \text{sentence}) &= \arg\max_T \frac{\mathrm{P}(T, W = \text{sentence})}{\mathrm{P}(W = \text{sentence})} = \arg\max_T \mathrm{P}(T, W = \text{sentence}) \\ &= \arg\max_T \mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1) \cdot \mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2) \\ &\quad \cdot \mathrm{P}(T_3|T_2) \cdot \mathrm{P}(W_3 = \text{like}|T_3) \cdot \mathrm{P}(T_4|T_3) \cdot \mathrm{P}(W_4 = \text{flies}|T_4)\end{aligned}$$

where T and W denote arrays of variables T_i and W_i. One way to find the values of T that maximize the given probability is to test all variations of their values. This number is exponential in general, and in this case it is $4^4 = 256$. A much more efficient solution can be obtained by applying a dynamic programming approach, known as the Viterbi algorithm.

"Brute-Force" Approach

– Try all combinations of variable values T_1, T_2, T_3, and T_4
– Calculate the overall probability for each of them using the formula

$$\begin{aligned}&\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1) \\ &\cdot \mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2) \\ &\cdot \mathrm{P}(T_3|T_2) \cdot \mathrm{P}(W_3 = \text{like}|T_3) \\ &\cdot \mathrm{P}(T_4|T_3) \cdot \mathrm{P}(W_4 = \text{flies}|T_4)\end{aligned}$$

– Choose the maximal probability

15.3 Efficient Tagging with HMM

Efficient Tagging with HMM

– Rather than using the brute-force approach, we can incrementally optimize the product expression by partial maximization from left to right
– One way to represent this is by using a table, which leads to the dynamic programming solution, or the Viterbi algorithm
– The second way to represent this computation is using message passing, or product-sum algorithm

HMM Inference: Dynamic Programming Solution

– Brute-force approach is too inefficient
– Idea for more efficient calculation: maximize sub-products first
– Dynamic Programming approach: divide problem into sub-problems
    – with a manageable number of sub-problems
– Find maximal partial configurations up to T_1, then T_2, T_3, and T_4

Viterbi Algorithm Example

The idea of the Viterbi algorithm is to incrementally calculate maximal values of the following parts of the above product:

$$\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1)$$

for all possible values of T_1, then

$$\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1) \cdot \mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2)$$

for all possible values of T_2 and so on. The computation can be summarized in the following table:

	T_1 (W_1 = flies)	T_2 (W_2 = *)	T_3 (W_3 = like)	T_4 (W_4 = flies)
	$\mathrm{P}(T_1)\mathrm{P}(W_1\vert T_1)$	$p\cdot\mathrm{P}(T_2\vert T_1)\mathrm{P}(W_2\vert T_2)$	$p\cdot\mathrm{P}(T_3\vert T_2)\mathrm{P}(W_3\vert T_3)$	$p\cdot\mathrm{P}(T_4\vert T_3)\mathrm{P}(W_4\vert T_4)$
D	$0\times 0=0$	DD: $0\times 0\times\frac{1}{3}=0$ ND: $\frac{1}{9}\times 0\times\frac{1}{3}=0$ PD: 0 VD: 0 max: 0	DD: $0\times 0\times 0=0$ ND: $\frac{1}{90}\times 0\times=0$ PD: $\frac{1}{50}\times\frac{1}{2}\times 0=0$ VD: $\frac{1}{90}\times 0\times 0=0$ max: 0	DD: $0\times 0\times 0=0$ ND: $0\times 0\times 0=0$ PD: $\frac{1}{225}\times 0.5\times 0=0$ VD: $0\times 0\times 0=0$ max: 0
N	$0.5\times\frac{2}{9}=\frac{1}{9}$	DN: $0\times 1\ldots=0$ NN: $\frac{1}{9}\times 0\ldots=0$ PN: $0\times\ldots=0$ VN: $0.2\times 0.5\times\frac{1}{9}=\frac{1}{90}$ max: $\frac{1}{90}$	DN: $0\times 1\times 0=0$ NN: $\frac{1}{90}\times 0\ldots=0$ PN: $\frac{1}{50}\times 0.5\times 0=0$ VN: $\frac{1}{90}\times 0.5\times 0=0$ max: 0	DN: $0\times 1\times\frac{2}{9}=0$ NN: $0\times 0\times\frac{2}{9}=0$ PN: $\frac{1}{225}\times 0.5\times\frac{2}{9}=\frac{1}{2025}$ VN: $0\times 0.5\times\frac{2}{9}=0$ max: $\frac{1}{2025}$
P	$0\times 0=0$	DP: $0\times\ldots=0$ NP: $\frac{1}{9}\times 0.5\times 0.2=\frac{1}{90}$ PP: $0\times\ldots=0$ VP: $0.2\times 0.5\times 0.2=\frac{1}{50}$ max: $\frac{1}{50}$	DP: $0\times 0\times 0.8=0$ NP: $\frac{1}{90}\times 0.5\times 0.8=\frac{1}{225}$ PP: $\frac{1}{50}\times 0\times 0.8=0$ VP: $\frac{1}{90}\times 0.5\times 0.8=\frac{1}{225}$ max: $\frac{1}{225}$	DP: $0\times 0\times 0=0$ NP: $0\times 0.5\times 0=0$ PP: $\frac{1}{225}\times 0\times 0=0$ VP: $0\times 0.5\times 0=0$ max: 0
V	$0.5\times 0.4=0.2$	DV: $0\times\ldots=0$ NV: $\frac{1}{9}\times 0.5\times 0.2=\frac{1}{90}$ PV: $0\times\ldots=0$ VV: $0.2\times 0\ldots=0$ max: $\frac{1}{90}$	DV: $0\times 0\times 0=0$ NV: $\frac{1}{90}\times 0.5\times 0=0$ PV: $\frac{1}{50}\times 0\times 0=0$ VV: $\frac{1}{90}\times 0\times 0=0$ max: 0	DV: $0\times 0\times 0.4=0$ NV: $0\times 0.5\times 0.4=0$ PV: $\frac{1}{225}\times 0\times 0.4=0$ VV: $0\times 0\times 0.4=0$ max: 0

The table is filled column by column. We can see now that the largest value that the expression

$$\mathrm{P}(T_1)\cdot\mathrm{P}(W_1=\text{flies}|T_1)\cdot\cdot\mathrm{P}(T_2|T_1)\cdot\mathrm{P}(W_2=*|T_2)\cdot\mathrm{P}(T_3|T_2)\cdot\mathrm{P}(W_3=\text{like}|T_3)\cdot\mathrm{P}(T_4|T_3)\cdot\mathrm{P}(W_4=\text{flies}|T_4)$$

can obtain is $\frac{1}{2025}$, and is achieved with $T_4=$ N. If we work backwards through the table, we can obtain the optimal values for previous variables as well: $T_3=$ P, $T_2=$ V, and $T_1=$ N. We can also choose $T_2=$ N, but in this case we have $T_1=$ V.