CSC 439/539
Statistical Natural Language Processing
Lecture 6: Sequence Models, (Visible) Markov Models

Mihai Surdeanu
Fall 2017

1

Take-away

- Maximum entropy Markov models (MEMM)
- Visible Markov models (MM) for POS tagging
- Training by counting
- Smoothing probabilities
- Handling unknown words
- Viterbi algorithm

2

Sequences in Information Extraction

Fully furnished condo in the beautiful Catalina
what
Foothills! Equipped with everything you need -
location
house wares, linens, full-size washer & dryer, cable
and Wi-Fi. Relax on your private covered patio or
take a dip in the sparkling pool! Close to fine dining
features
and shopping, too. List price is average, please call
neighborhood
price
for exact pricing and availability.
contact

3

Sequences Are Everywhere!

- Speech recognition
  - Group phonemes into words
- Natural language processing
  - Part of speech tagging
  - Named entity recognition
  - Information extraction
  - Question answering
- Bioinformatics
  - Protein folding

- We will use POS tagging as a use case, for its simplicity

One of the main motivations for interdisciplinarity in ML!

4

Why POS Tagging Must Model Sequences

Secretariat is expected to **race** tomorrow.

- We want to generate all POS tags *jointly* given *all* words in the sentence!

5

Approach 0: Rule-based baseline

1. Assign each word a list of potential POS labels using the dictionary
2. Winnow down the list to a single POS label for each word using lists of hand-written disambiguation rules

6

Example: adverbial-that winnow rule

Given input: "that"
if
(+1 A/ADV/QUANT); /* *if next word is adj, adverb, or quantifier* */
(+2 SENT-LIM); /* *and following which is a sentence boundary.* */
(NOT-1 SVOC/A); /* *and the previous word is not a verb like* */
/* *'consider' which allows adjs as object complements* */
then eliminate non-ADV tags
else eliminate ADV tag

- You can learn these rules: see Transformation-based Learning: https://dl.acm.org/citation.cfm?id=218367

Slide by Jungyeul Park

7

Approach 1: Maximum entropy Markov models

- Maximum entropy = logistic regression
- Markov models
  - Discovered by Andrey Markov
  - Limited horizon
- How would you implement sequence models in the logistic regression algorithm that we know?
  - Let's assume we scan the text left to right.

8

Approach 1 continued

- Add the previously seen tags as features!
  - Use gold tags in training
  - Use predicted tags in testing
- Other common features
  - Words, lemmas in a window [-k, +k]
  - Casing info, prefixes, suffixes of these words
  - Bigrams containing the current word
  - See also: https://github.com/clulab/processors/blob/master/main/src/main/scala/org/clulab/processors/clu/sequences/PartOfSpeechTagger.scala

9

Practice Exercise

- What are the features and labels for each word in your MEMM POS classifier for the sentence:
- John/NNP goes/VBZ to/TO China/NNP ./.

10

Approach 1: bidirectional MEMMs

- You can stack MEMMs that traverse the text in opposite directions:
  - Left-to-right direction (same as before)
  - Right-to-left: uses the prediction(s) of the above system as features!
    - What is the problem with the predictions of the left-to-right model here?
- Many state-of-the-art taggers use this approach: CoreNLP, processors, SVMTool

11

Approach 2

- Let's put the probability theory we covered in the previous lecture to use!
- The resulting approach is called (visible) Markov model
  - "Visible" to distinguish it from the *hidden* Markov models, where the tags are unknown
    - Imagine implementing a POS tagger for an unstudied language without POS annotations

12

Approach 2: (Visible) Markov models

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Sentence 1 contains *n* words
- t_1^n - an assignment of POS tags to this sentence
- w_1^n - the words in this sentence
- $\hat{t}_1^n$ - the estimate of optimal tag assignment

(The function $\operatorname{argmax}_x f(x)$ means "the x such that f(x) is maximized".)

13

Let's Formalize This

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

What's this?

Why can we ignore that?

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

These two are still way too sparse!

14

Three Approximations

- Words are independent of the words around them
- Words depend only on their POS tags, not on the neighboring POS tags

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

Which rule was used to generate these?

- A tag is dependent only on the previous tag

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

15

So…

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i|t_i)}_{\text{Word likelihoods}} \underbrace{P(t_i|t_{i-1})}_{\text{Tag transition probabilities}}$$

16

Computing Tag Transition Probabilities

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

- In the Brown corpus (1M words)
  - DT occurs 116,454 times
  - DT is followed by NN 56,509 times

$$P(NN|DT) = \frac{C(DT,NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

17

Computing Word Likelihoods

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

- In the Brown corpus (1M words)
  - VBZ occurs 21,627 times
  - VBZ is the tag for “is” 10,073 times

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

18

Example

Secretariat/NNP is/BEZ expected/VBN to/TO **race**/VB tomorrow/NR

People/NNS continue/VB to/TO inquire/VB the/AT reason/NN for/IN the/AT **race**/NN for/IN outer/JJ space/NN

- “race” is ambiguous
- Let’s see why VB is preferred in the first case

19

Example

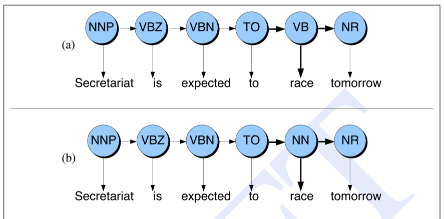


Figure 5.12 Two of the possible sequences of tags corresponding to the Secretariat sentence, one of them corresponding to the correct sequence, in which *race* is a VB. Each arc in these graphs would be associated with a probability. Note that the two graphs differ only in 3 arcs, hence in 3 probabilities.

20

Example

- The first tag transition
  - P(NN|TO) = 0.00047
  - P(VB|TO) = .83
- The word likelihood for “race”
  - P(race|NN) = 0.00057
  - P(race|VB) = 0.00012
- The second tag transition
  - P(NR|VB) = 0.0027
  - P(NR|NN) = 0.0012

21

Example

- P(VB|TO)P(NR|VB)P(race|VB) = 0.00000027
- P(NN|TO)P(NR|NN)P(race|NN) = 0.00000000032
- VB is more likely than NN, even though “race” appears more commonly as a noun!

22

Practice Exercise

- You are implementing a POS tagger for Klingon. In your corpus you see the word “vjIjatlh” a total of 1,000 times, out of which 100 times it is seen with the POS tag VB. In the corpus there is a total of 10,000 words tagged as VB. What is P(vjIjatlh|VB)?

23

Practice Exercise (1/2)

- Given the following MM, in a hypothetical language containing 5 POS tags and 4 words:

from/to	tag1	tag2	tag3	tag4	tag5
tag1	0.2	0.2	0.2	0.2	0.2
tag2	0.1	0.15	0.5	0.15	0.1
tag3	0.3	0.1	0.3	0.1	0.2
tag4	0.4	0.05	0.1	0.4	0.05
tag5	0.2	0.3	0.1	0.1	0.3

Table 1: Transition Matrix

state/obsv.	w1	w2	w3	w4
tag1	0.25	0.25	0.25	0.25
tag2	0.30	0.20	0.30	0.20
tag3	0.30	0.10	0.30	0.30
tag4	0.10	0.20	0.30	0.40
tag5	0.20	0.30	0.20	0.30

Table 2: Emissions Matrix

state	prob.
tag1	0.15
tag2	0.25
tag3	0.20
tag4	0.25
tag5	0.15

Table 3: Start State Probabilities

24

Practice Exercise (2/2)

- What is the best sequence of POS tags for the sentences below, in a greedy left-to-right tagging approach using this MM? That is, in each step choose the POS tag that maximizes the current $P(t_i|t_{i-1})$.
- What is the overall probability for each best sequence?

w1 w2 w3 w4
w3 w1 w4 w1
w2 w1 w2 w4

25

Formalization

An **HMM** is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A sequence of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

tags = states; words = observations

26

Training/Testing an HMM

- Just like with any machine learning algorithm, there are two important issues one needs to do to build an HMM:
- **Training**:
  - Estimating $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
- **Testing** (predicting):
  - Estimating the best sequence of tags for a sentence (or sequence or words)

27

Training: Two Types of Probabilities

- *A*: transition probabilities
  - Used to compute the prior probabilities
- *B*: observation likelihoods
  - Used to compute the likelihood probabilities

A Transition Probabilities

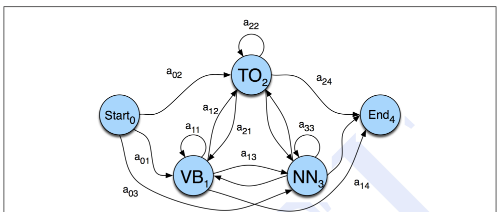


Figure 5.13 The Markov chain corresponding to the hidden states of the HMM. The *A* transition probabilities are used to compute the prior probability.

B Observation Likelihoods

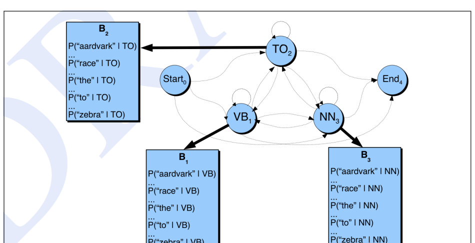


Figure 5.14 The *B* observation likelihoods for the HMM in the previous figure. Each state (except the non-emitting Start and End states) is associated with a vector of probabilities, one likelihood for each possible observation word.

Testing: Viterbi Algorithm

This is expensive to do...

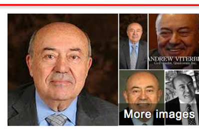
$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

because of this.

- Viterbi algorithm
  - Computes the argmax efficiently
  - Example of dynamic programming

31

Viterbi



Andrew Viterbi

Engineer

Andrew James Viterbi is an American electrical engineer and businessman who co-founded Qualcomm Inc. and invented the Viterbi algorithm. Wikipedia

Born: March 9, 1935 (age 78), Bergamo, Italy

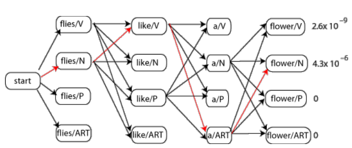
Books: CDMA, Principles of digital communication and coding

Education: University of Southern California (1963), More

Awards: IEEE Medal of Honor, Claude E. Shannon Award, More

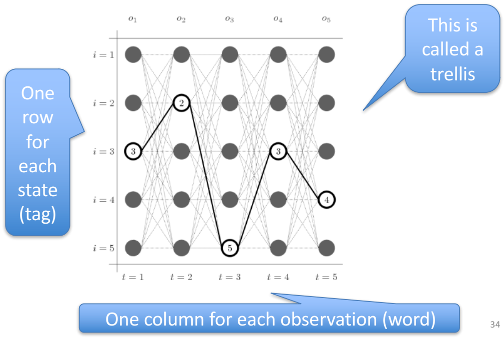
32

Illustration of Search Space



33

Illustration of Search Space



34

Viterbi Algorithm

- Input
  - State (or tag) transition probabilities (A)
  - Observation (or word) likelihoods (B)
  - An observation sequence O = ($o_1 o_2 \ldots o_T$)
- Output
  - Most probable state sequence Q = ($q_1 q_2 \ldots q_T$) together with its probability

35

Viterbi Algorithm

function VITERBI(*observations* of len T, *state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ;initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ;recursion step
for each state s **from** 1 **to** N **do**
$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(o_t)$
$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s', t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F, T]$

Note that states 0 (virtual start) and q_F (virtual end) do not emit words.

36

Example: A and B Matrices

A: The rows are labeled with the conditioning event, e.g., P(PPSS|VB) = .0070

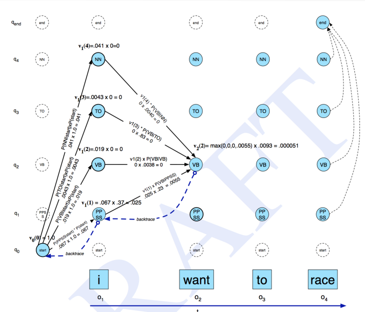
	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

B:

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

37

Example Trace



What's the complexity of this algorithm?

38

Summary of Viterbi Algorithm

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

- $v_{t-1}(i)$ – the **previous Viterbi path probability** from the previous time step t – 1 (i.e., the previous word)
- a_{ij} – the **transition probability** from previous state q_i (i.e., the previous word having POS tag i) to current state q_j (i.e., the current word having POS tag j)
- $b_j(o_t)$ – the **state observation likelihood** of the observation symbol o_t (i.e., word at position t) given the current state j (i.e., the j POS tag)

39

Practice Exercise

- What is the Viterbi path in the trellis for the MM in the previous practice exercise for each of these sequences:

  w1 w2 w3 w4
  w3 w1 w4 w1
  w2 w1 w2 w4

40

Extending the HMM Algorithm to Trigrams

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

This is pretty limiting for POS tagging
Let's extend it to trigrams of tags!

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1},t_{i-2})\right] P(t_{n+1}|t_n)$$

This is better

- t_{n+1} – end of sentence tag
- We also need virtual tags, t_0 and t_{-1}, to be set to the beginning of sentence value.

41

TnT

- This is what the TnT (Trigrams'n'Tags) tagger does
  - Probably the fastest POS tagger in the world
  - Not the best, but pretty close (96% acc)
- http://www.coli.uni-saarland.de/~thorsten/tnt/

42

One Problem

$$P(t_i|t_{i-1},t_{i-2}) = \frac{C(t_{i-2},t_{i-1},t_i)}{C(t_{i-2},t_{i-1})}$$

Very sparse!

Backoff model: linear interpolation

$$P(t_i|t_{i-1}t_{i-2}) = \lambda_1 \hat{P}(t_i|t_{i-1}t_{i-2}) + \lambda_2 \hat{P}(t_i|t_{i-1}) + \lambda_3 \hat{P}(t_i)$$

λ1 + λ2 + λ3 = 1, to guarantee that result is a probability. How to set them?

43

Deleted Interpolation

function DELETED-INTERPOLATION(*corpus*) **returns** $\lambda_1, \lambda_2, \lambda_3$

$\lambda_1 \leftarrow 0$
$\lambda_2 \leftarrow 0$
$\lambda_3 \leftarrow 0$
foreach trigram t_1, t_2, t_3 with $f(t_1,t_2,t_3) > 0$
  depending on the maximum of the following three values
    case $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: increment λ_3 by $C(t_1,t_2,t_3)$
    case $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: increment λ_2 by $C(t_1,t_2,t_3)$
    case $\frac{C(t_3)-1}{N-1}$: increment λ_1 by $C(t_1,t_2,t_3)$
  end
end
normalize $\lambda_1, \lambda_2, \lambda_3$
return $\lambda_1, \lambda_2, \lambda_3$

What is the normalization count here, to guarantee that these 3 values sum up to 1?

- Notes:
  - N – total number of words in the corpus
  - When denominator is 0, result is 0

44

Other Types of Smoothing

- Add one:
  - $P(w \mid t) = \dfrac{C(w,t)+1}{C(t)+K}$
  - Where K is the number of words with POS tag t
- Variant of add one (Charniak's):
  - $P(t_i \mid t_{i-1}) = (1-\varepsilon)\dfrac{C(t_i,t_{i-1})}{C(t_{i-1})} + \varepsilon$
  - Not a proper probability distribution!

45

Practice Exercise

- You are implementing a POS tagger for Klingon. In your corpus you see the word “vjIjatlh” a total of 1,000 times, out of which 100 times it is seen with the POS tag VB. In the corpus there is a total of 10,000 words tagged as VB. What is P(vjIjatlh|VB)
  - Using “add one” smoothing?
  - Using Charniak smoothing for epsilon = 0.01?

46

Another Problem for All HMMs

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Work in log space to avoid underflow!

47

Yet Another Problem: Unknown Words

- Solution 0 (not great): assume uniform emission probabilities (this is what “add one” smoothing does)
  - You can exclude closed-class POS tags such as…
  - This does not use any lexical information such as suffixes
- Solution 1: capture lexical information:
  - $P(w^l|t^j) = \frac{1}{Z} P(\text{unknown word}|t^j) P(\text{capitalized}|t^j) P(\text{endings/hyph}|t^j)$
  - This reduces error rate for unknown words from 40% to 20%

48

Main Disadvantage of HMMs

- Hard to add features in the model
  - Capitalization, hyphenated, suffixes, etc.
- It's possible but every such feature must be encoded in the p(word|tag)
  - Redesign the model for every feature!
  - MEMMs avoid this limitation, but they take longer to train

49

HMM vs. MEMM

(H)MM

$$\hat{T} = \operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T P(W|T)P(T)$$

MEMM

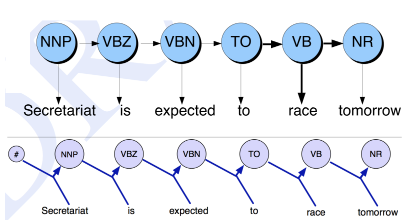
$$\hat{T} = \operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T \prod_i P(tag_i|word_i, tag_{i-1})$$

50

Illustration

(H)MM

MEMM



51

Formalization

(H)MM

$$P(Q|O) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

MEMM

$$P(Q|O) = \prod_{i=1}^{n} P(q_i|q_{i-1}, o_i)$$

52

Testing (Decoding)

- A slight variation of Viterbi:

(H)MM

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i)\, P(o_t|s_j) \quad 1 \leq j \leq N, 1 < t \leq T$$

MEMM

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i, o_t) \quad 1 \leq j \leq N, 1 < t \leq T$$

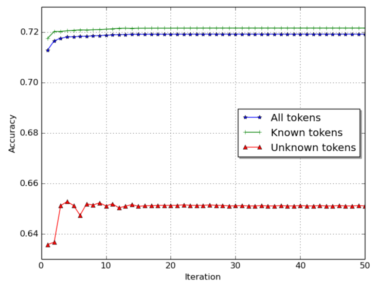
53

Evaluation

- POS tagging accuracy = 100 x (number of correct tags) / (number of words in dataset)
- Accuracy numbers currently reported for POS tagging are most often between 95% and 97%
  - But they are much worse for "unknown" words

54

Evaluation example



Persian POS tagging results using cross-lingual projection

55

Evaluation for NER/IE

- Accuracy does not work. Why?
- We need precision, recall, F1:
  - P = TP/(TP + FP)
  - R = TP/(TP + FN)
  - F1 = 2PR/(P + R)
- Micro vs. macro F1 measures
  - What are these?
  - What are the advantages/disadvantages of each?

56

Hidden Markov Models

- What if you want to build a POS tagger for a language where you have no annotations?
- Hidden Markov models
  - The states are unknown
  - Kinda like clustering for sequences
  - Not extremely useful in practice…
  - Skipping for now. We may revisit this idea at the end of the course, time permitting

57

Readings

- FSNLP chapter 10
- Optional, using MEMM for information extraction: http://cseweb.ucsd.edu/~elkan/254spring02/gidofalvi.pdf
- Optional, bi-directional MEMM for POS tagging: https://nlp.stanford.edu/~manning/papers/tagging.pdf

58

Take-away

- Maximum entropy Markov models (MEMM)
- Visible Markov models (MM) for POS tagging
- Training by counting
- Smoothing probabilities
- Handling unknown words
- Viterbi algorithm

59