Stochastic models for learning language models

R. Basili

Corso di *Web Mining* e *Retrieval*
a.a. 2013-14

March 23, 2014

Outline

Outline

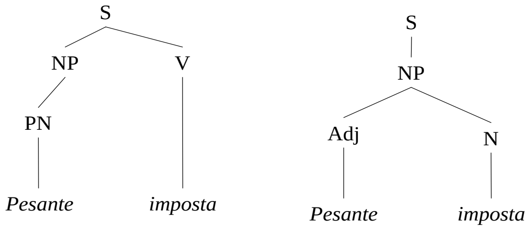
Quantitative Models of language structures

Linguistic structures are example of structures where syntagmatic information is crucial for machine learning. The most used modeling here are grammars:

```
1.    S   -> NP  V
2.    S   -> NP
3.    NP  -> PN
4.    NP  -> N
5.    NP  -> Adj N
6.    N   -> "imposta"
7.    V   -> "imposta"
8.   Adj  -> "pesante"
9.    PN  -> "Pesante"
...
```

The role of Quantitative Approaches

“*Pesante imposta*”



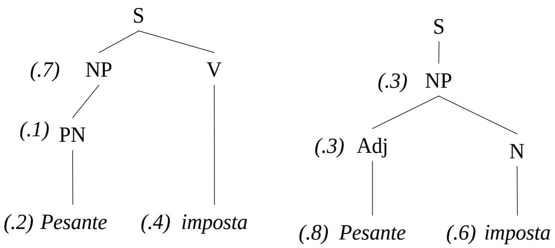
The role of Quantitative Approaches

Weighted grammars are models of (possibly limited) *degrees of grammaticality*. They are meant to deal with a large range of ambiguity problems:

```
1.    S  -> NP  V        .7
2.    S  -> NP           .3
3.    NP -> PN           .1
4.    NP -> N            .6
5.    NP -> Adj N        .3
6.    N  -> imposta      .6
7.    V  -> imposta      .4
8.   Adj -> Pesante      .8
9.    PN -> Pesante      .2
```

Linguistic Ambiguity and weighted grammars

"Pesante imposta"



Linguistic Ambiguity and weighted grammars

Weighted grammars allow to compute the degree of grammaticality of different ambiguous derivations, thus supporting disambiguation:

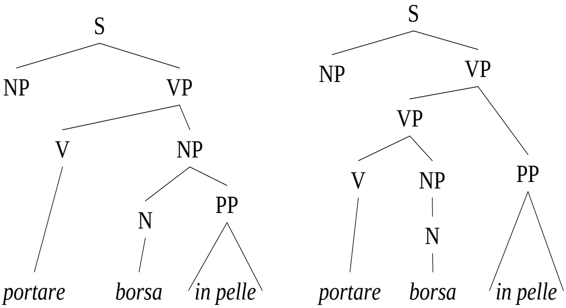
```
1.    S   -> NP  V        .7
2.    S   -> NP           .3
3.    NP  -> PN           .1
4.    NP  -> N            .6
5.    NP  -> Adj N        .3
6.    N   -> imposta      .6
7.    V   -> imposta      .4
8.   Adj  -> Pesante      .8
9.    PN  -> Pesante      .2
...
```

prob(((Pesante)$_{PN}$ (imposta)$_V$)$_S$)= $(.7 \cdot .1 \cdot .2 \cdot .4) = 0.0084$

prob(((Pesante)$_{Adj}$ (imposta)$_N$)$_S$)= $(.3 \cdot .3 \cdot .8 \cdot .6) = 0.0432$

Syntactic Disambiguation

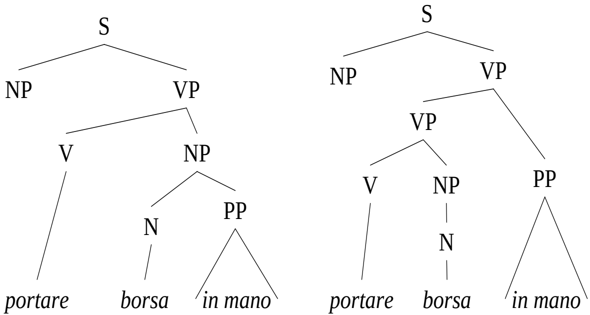
"portare borsa in pelle"



Derivation Trees for a structurally ambiguous sentence

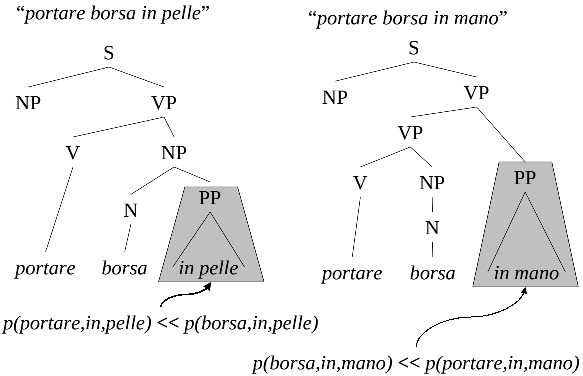
Syntactic Disambiguation (cont'd)

"portare borsa in mano"



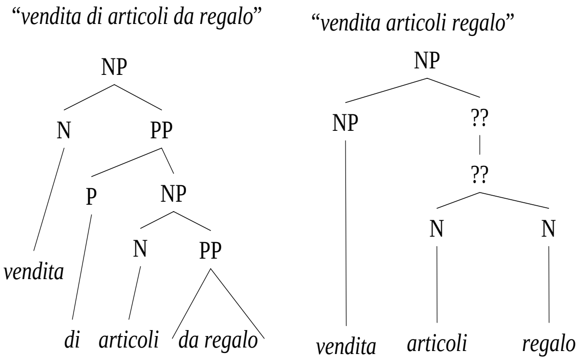
Derivation Trees for a second structurally ambiguous sentence.

Structural Disambiguation (cont'd)



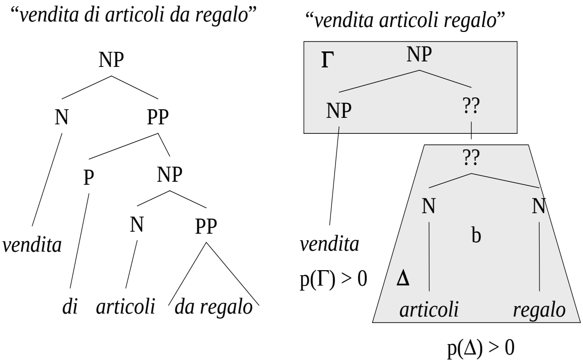
Disambiguation of structural ambiguity.

Tolerance to errors



An example of ungrammatical but meaningful sentence

Error tolerance (cont'd)



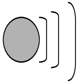
Modeling of ungrammatical phenomena

Probability and Language Modeling

- Aims
  - to extend grammatical (i.e. rule-based) models with predictive and disambiguation capabilities
  - to offer theoretically well founded *inductive methods*
  - to develop (not merely) quantitative models of linguistic phenomena
- Methods and Resources:
  - Methematical theories (e.g. Markov models)
  - Systematic testing/evaluation frameworks
  - Extended repositories of examples of *language in use*
  - Traditional linguistic resources (e.g. "models" like dictionaries)

Probability and Language Modeling

- Signals are abstracted via symbols that are not known in advance
- Emitted signals belong to an alphabet A
- Time is discrete: each time point corresponds to an emitted signal
- Sequences of symbols $(w_1, \ldots, w_n)$ correspond to sequences of time points $(1, \ldots, n)$



$\ldots,\ \underline{8,\ 7,\ 6,\ 5,\ 4,\ 3,\ 2,\ 1}$

$\ldots,\ W_{i8},\ W_{i7},\ W_{i6},\ W_{i5},\ W_{i4},\ W_{i3},\ W_{i2},\ W_{i1}$

Probability and Language Modeling

A generative language model

A random variable X can be introduced so that

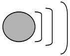
- It assumes values w_i in the alfabet A
- Probability is used to describe the uncertainty on the emitted signal

$$p(X = w_i) \qquad w_i \in A$$

Probability and Language Modeling

- A random variable X can be introduced so that
  - X assumes values in A at each step i, i.e. $X_i = w_j$
  - probability is $p(X_i = w_j)$
- Constraints: the total probability is for each step:

$$\sum_j p(X_i = w_j) = 1 \qquad \forall i$$

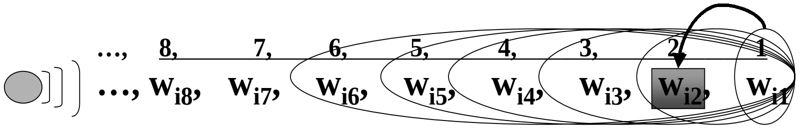


..., 8, 7, 6, 5, 4, 3, 2, 1

..., w_{i8}, w_{i7}, w_{i6}, w_{i5}, w_{i4}, w_{i3}, w_{i2}, w_{i1}

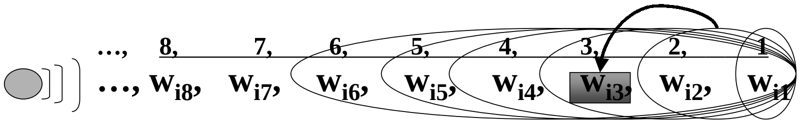
Probability and Language Modeling

- Notice that time points can be represented as **states** of the emitting source
- An output w_i can be considered as emitted in a *given state* X_i by the source, and *given a certain* **history**



Probability and Language Modeling

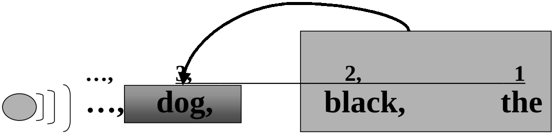
- Formally:
  - $P(X_i = w_i, X_{i-1} = w_{i-1}, \ldots X_1 = w_1) =$
    $= P(X_i = w_i | X_{i-1} = w_{i-1}, X_{i-2} = w_{i-2}, \ldots, X_1 = w_1) \cdot$
    $P(X_{i-1} = w_{i-1}, X_{i-2} = w_{i-2}, \ldots, X_1 = w_1)$



Probability and Language Modeling

What's in a state

$n-1$ preceding words $\Rightarrow$ n**-gram language models**

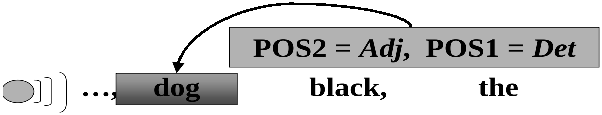


$$p(the, black, dog) = p(dog|the, black)p(black|the)p(the)$$

Probability and Language Modeling

What's in a state

preceding POS tags ⇒ **stochastic taggers**

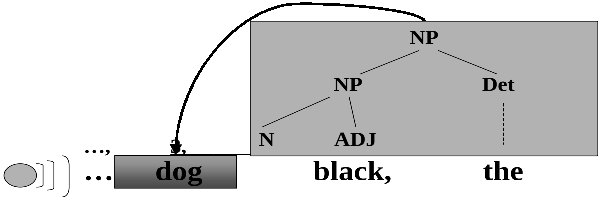


$$p(the_{DT}, black_{ADJ}, dog_N) = p(dog_N | the_{DT}, black_{ADJ}) \ldots$$

Probability and Language Modeling

What's in a state

preceding *parses* ⇒ **stochastic grammars**



$$p((the_{Det},(black_{ADJ},dog_N)_{NP})_{NP}) =$$
$$p(dog_N|((the_{Det}),(black_{ADJ},\_)))\ldots$$

Probability and Language Modeling (2)

- Expressivity
  - The predictivity of a statistical grammar can provide a very good explanatory model of the source language (string)
  - Acquiring information from data has a clear definition, with simple and sound induction algorithms
  - Simple but richer descriptions (e.g. grammatical preferences)
  - Optimal Coverage (i.e. better on *more important phenomena*)
- Integrating Linguistic Description
  - Start with poor assumptions and approximate as much as possible *what is known* (early evaluate only performance)
  - *Bias* the statistical model since the beginning and check the results on a *linguistic ground*

Probability and Language Modeling (3)

Advantages: Performances

- Faster Processing
- Faster Design
- Linguistic Adequacy
  - Acceptance
  - Psychological Plausibility
  - Explanatory power
- Tools for further analysis of Linguistic Data

Markov Models

Markov Models

Suppose $X_1, X_2, ..., X_T$ form a sequence of random variables taking values in a countable set $W = p_1, p_2, ..., p_N$ (State space).

- Limited Horizon Property:
  $P(X_{t+1} = p_k | X_1, ..., X_t) = P(X_{t+1} = k | X_t)$
- Time invariant:
  $P(X_{t+1} = p_k | X_t = p_l) = P(X_2 = p_k | X_1 = p_l) \qquad \forall t (> 1)$

It follows that the sequence of $X_1, X_2, ..., X_T$ is a **Markov chain**.

Representation of a Markov Chain

Markov Models: Matrix Representation

- A (transition) matrix A:

$$a_{ij} = P(X_{t+1} = p_j | X_t = p_i)$$

Note that $\forall i,j \quad a_{ij} \geq 0$ and $\forall i \quad \sum_j a_{ij} = 1$

- Initial State description (i.e. probabilities of initial states):

$$\pi_i = P(X_1 = p_i)$$

Note that $\sum_{j=1}^{n} \pi_{ij} = 1$.

Representation of a Markov Chain

Graphical Representation (i.e. Automata)

- States as nodes with names
- Transitions from states i-th and j-th as arcs labelled by conditional probabilities $P(X_{t+1} = p_j | X_t = p_i)$
  Note that 0 probability arcs are omitted from the graph.

	S_1	S_2
S_1	0.70	0.30
S_2	0.50	0.50

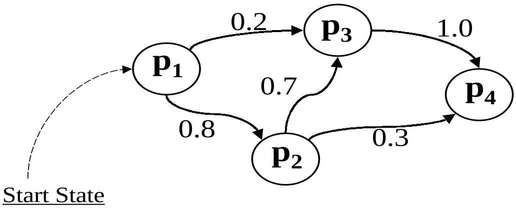
Representation of a Markov Chain

Graphical Representation

$$P(X_1 = p_1) = 1 \qquad \leftarrow StartState$$

$$P(X_k = p_3 | X_{k-1} = p_2) = 0.7 \qquad \forall k$$

$$P(X_k = p_4 | X_{k-1} = p_1) = 0 \qquad \forall k$$



A Simple Example of Hidden Markov Model

Crazy Coffee Machine

- Two states: Tea Preferring (*TP*), Coffee Preferring (*CP*)
- Switch from one state to another randomly
- Simple (or visible) Markov model:
  Iff the machine output *Tea* in *TP* AND *Coffee* in *CP*

What we need is a description of the random event of switching from one state to another. More formally we need for each time step n and couple of states p_i and p_j to determine following conditional probabilities:

$$P(X_{n+1} = p_j | X_n = p_i)$$

where p_t is one of the two states *TP*, *CP*.

A Simple Example of Hidden Markov Model

Crazy Coffee Machine

Assume, for example, the following state transition model:

	TP	CP
TP	0.70	0.30
CP	0.50	0.50

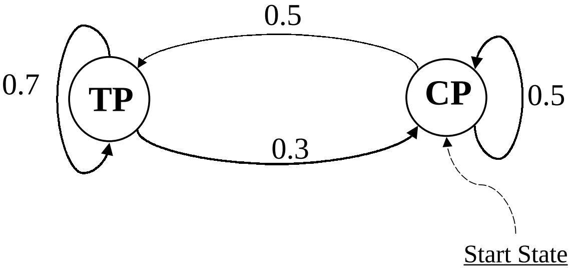
and let CP be the starting state (i.e. $\pi_{CP} = 1$, $\pi_{TP} = 0$).

Potential Use:

1. What is the probability at time step 3 to be in state TP?
2. What is the probability at time step n to be in state TP?
3. What is the probability of the following sequence in output: $(Coffee, Tea, Coffee)$?

Crazy Coffee Machine

Graphical Representation



Crazy Coffee Machine

Solution to Problem 1:

$P(X_3 = TP) =$ (given by (CP, CP, TP) *and* (CP, TP, TP))
$= P(X_1 = CP) \cdot P(X_2 = CP|X_1 = CP) \cdot P(X_3 = TP|X_1 = CP, X_2 = CP) +$
$+ \quad P(X_1 = CP) \cdot P(X_2 = TP|X_1 = CP) \cdot P(X_3 = TP|X_1 = CP, X_2 = TP) =$
$= P(CP)P(CP|CP)P(TP|CP, CP) + P(CP)P(TP|CP)P(TP|CP, TP) =$
$= P(CP)P(CP|CP)P(TP|CP) + P(CP)P(TP|CP)P(TP|TP) =$
$= 1 \cdot 0.50 \cdot 0.50 + 1 \cdot 0.50 \cdot 0.70 = 0.25 + 0.35 = 0.60$

Crazy Coffee Machine

Solution to Problem 2

$P(X_n = TP) =$
$\sum_{CP,p_2,p_3,\ldots,TP} P(X_1 = CP)P(X_2 = p_2|X_1 = CP)P(X_3 = p_3|X_1 = CP, X_2 = p_2) \cdot \ldots \cdot P(X_n = TP|X_1 = CP, X_2 = p_2, \ldots, X_{n-1} = p_{n-1}) =$
$= \sum_{CP,p_2,p_3,\ldots,TP} P(CP)P(p_2|CP)P(p_3|p_2) \cdot \ldots \cdot P(TP|p_{n-1}) =$
$= \sum_{CP,p_2,p_3,\ldots,TP} P(CP) \cdot \prod_{t=1}^{n-1} P(p_{t+1}|p_t) =$
$= \sum_{p_1,\ldots,p_n} P(p_1) \cdot \prod_{t=1}^{n-1} P(p_{t+1}|p_t)$

Crazy Coffee Machine

Solution to Problem 3:

$$P(Cof, Tea, Cof) =$$
$$= P(Cof) \cdot P(Tea|Cof) \cdot P(Cof|Tea) = 1 \cdot 0.5 \cdot 0.3 = 0.15$$

A Simple Example of Hidden Markov Model (2)

Crazy Coffee Machine

- **Hidden** Markov model: If the machine output *Tea*, *Coffee* or *Capuccino* **independently** from *CP* and *TP*.

What we need is a description of the random event of output(ting) a drink.

Crazy Coffee Machine

A description of the random event of output(ting) a drink. Formally we need (for each time step n and for each kind of output $O = \{Tea, Cof, Cap\}$), the following conditional probabilities:

$$P(O_n = k | X_n = p_i, X_{n+1} = p_j)$$

where k is one of the values *Tea*, *Coffee* or *Capuccino*. This matrix is called the **output matrix** of the machine (or of its Hidden markov Model).

A Simple Example of Hidden Markov Model (2)

Crazy Coffee Machine
Given the following output probability for the machine

	Tea	Coffee	Capuccino
TP	0.8	0.2	0.0
CP	0.15	0.65	0.2

and let CP be the starting state (i.e. $\pi_{CP} = 1$, $\pi_{TP} = 0$).

- Find the following probabilities of output from the machine
  1. $(Cappuccino, Coffee)$ given that the state sequence is (CP, TP, TP)
  2. $(Tea, Coffee)$ for any state sequence
  3. a generic output $O = (o_1, \ldots, o_n)$ for *any* state sequence

A Simple Example of Hidden Markov Model (2)

Solution for the problem 1 For the given state sequence
$X = (CP, TP, TP)$
$P(O_1 = Cap, O_2 = Cof, X_1 = CP, X_2 = TP, X_3 = TP) =$
$P(O_1 = Cap, O_2 = Cof | X_1 = CP, X_2 = TP, X_3 = TP)P(X_1 = CP, X_2 = TP, X_3 = TP)) =$
$P(Cap, Cof | CP, TP, TP)P(CP, TP, TP))$ Now:
$P(Cap, Cof | CP, TP, TP)$ is the probability of output Cap, Cof *during* transitions from CP to TP and TP to TP
and $P(CP, TP, TP)$ is the probability of the transition chain.
Therefore,
$= P(Cap|CP, TP)P(Cof|TP, TP)$ =(in our simplified model)
$= P(Cap|CP)P(Cof|TP) = 0.2 \cdot 0.2 = 0.04$

A Simple Example of Hidden Markov Model (2)

Solutions for the problem 2

In general, for any sequence of three states $X = (X_1, X_2, X_3)$

$P(Tea, Cof|X_1, X_2, X_3) =$

$P(Tea, Cof) =$ (as sequences are a partition for the sample space)

$= \sum_{X_1,X_2,X_3} P(Tea, Cof|X_1, X_2, X_3)P(X_1, X_2, X_3)$ where

$P(Tea, Cof|X_1, X_2, X_3) = P(Tea|X_1, X_2)P(Cof|X_2, X_3) =$

(for the simplified model of the coffee machine)

$= P(Tea|X_1)P(Cof|X_2)$ and (for the Markov constraint)

$P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_2)$

The simplified model is concerned with only the following transition chains

(CP, CP, CP), (CP, TP, CP), (CP, CP, TP)

(CP, TP, TP)

A Simple Example of Hidden Markov Model (2)

Solutions for the problem 2

In general, for any sequence of three states $X = (X_1, X_2, X_3)$

The following probability is given

$P(Tea, Cof) =$

$$
\begin{aligned}
& P(Tea|CP)P(Cof|CP)P(CP)P(CP|CP)P(CP|CP)+ && \text{st.: } (CP,CP,CP)) \\
& P(Tea|CP)P(Cof|TP)P(CP)P(TP|CP)P(CP|TP)+ && \text{st.: } (CP,TP,CP)) \\
& P(Tea|CP)P(Cof|CP)P(CP)P(CP|CP)P(TP|CP)+ && \text{st.: } (CP,CP,TP)) \\
& P(Tea|CP)P(Cof|TP)P(CP)P(TP|CP)P(TP|TP) = && \text{st.: } (CP,TP,TP))
\end{aligned}
$$

$$
\begin{aligned}
&= 0.15 \cdot 0.65 \cdot 1 \cdot 0.5 \cdot 0.5 + \\
&+ \quad 0.15 \cdot 0.2 \cdot 1 \cdot 0.5 \cdot 0.3 + \\
&+ \quad 0.15 \cdot 0.65 \cdot 1 \cdot 0.5 \cdot 0.5 + \\
&+ \quad 0.15 \cdot 0.2 \cdot 1.0 \cdot 0.5 \cdot 0.7 =
\end{aligned}
$$

$$
\begin{aligned}
&= 0.024375 + 0.0045 + 0.024375 + 0.0105 = \\
&= 0.06375
\end{aligned}
$$

A Simple Example of Hidden Markov Model (2)

Solution to the problem 3 (*Likelihood*)
In the general case, a sequence of n symbols $O = (o_1, ..., o_n)$ out from any sequence of $n+1$ transitions $X = (p_1, ..., p_{n+1})$ can be predicted by the following probability:

$$P(O) = \sum_{p_1,...,p_{n+1}} P(O|X)P(X) =$$

$$= \sum_{p_1,...,p_{n+1}} P(CP) \prod_{t=1}^{n} P(O_t|p_t, p_{t+1})P(p_{t+1}|p_t)$$

Modeling linguistic tasks as Stochastic Processes

Advantages

There are several advantages to model a linguistic problem as an HMM

- It is a powerful mathematical framework for modeling
- It provides clear problems settings for different applications: **estimation**, **decoding** and **model induction**
- HMM-based models provides sound solutions for the above applications

We will see an example as the HMM modeling of POS tagging

Fundamental problems for HMM

Fundamental Questions for HMM

The complexity of training and decoding can be limited by the use of optimization techniques

- Given the observation sequence $O = O_1, ..., O_n$ and a model $\lambda = (E, T, \pi)$, how to efficiently compute $P(O|\lambda)$? (*Language Modeling*)
- Given the observation sequence $O = O_1, ..., O_n$ and a model $\lambda = (E, T, \pi)$, how do we choose the optimal state sequence $Q = q_1, ..., q_n$ responsible of generating O ? (*Tagging/Decoding*)
- How to adjust model parameters $\lambda = (E, T, \pi)$ so to maximize $P(O|\lambda)$? (*Model Induction*)

HMM: Mathematical Methods

All the above problems can be approached by several optimization techniques able to limit the complexity.

- Language Modeling via *dynamic programming* (**Forward algorithms**) ($O(n)$)
- Tagging/Decoding via *dynamic programming* ($O(n)$) (**Viterbi**)
- Parameter estimation via *entropy minimization* (*EM*)

A relevant issue is the availability of source data: supervised training cannot be applied always

The task of POS tagging

POS tagging

Given a sequence of morphemes $w_1, \ldots, w_n$ with ambiguous syntactic descriptions (i.e.part-of-speech tags) t_j, compute the sequence of n POS tags $t_{j_1}, \ldots, t_{j_n}$ that characterize correspondingly all the words w_i.

Examples:

- *Secretariat is expected to race tomorrow*
- $\Rightarrow$ `NNP VBZ VBN TO VB NR`
- $\Rightarrow$ `NNP VBZ VBN TO NN NR`

HMM and POS tagging

Given a sequence of morphemes $w_1, ..., w_n$ with ambiguous syntactic descriptions (i.e.part-of-speech tags), derive the sequence of n POS tags $t_1, ..., t_n$ that maximizes the following probability:

$$P(w_1, ..., w_n, t_1, ..., t_n)$$

that is

$$(t_1, ..., t_n) = argmax_{pos_1, ..., pos_n} P(w_1, ..., w_n, pos_1, ..., pos_n)$$

Note that this is equivalent to the following:
$(t_1, ..., t_n) = argmax_{pos_1, ..., pos_n} P(pos_1, ..., pos_n | w_1, ..., w_n)$
as: $\frac{P(w_1, ..., w_n, pos_1, ..., pos_n)}{P(w_1, ..., w_n)} = P(pos_1, ..., pos_n | w_1, ..., w_n)$
and $P(w_1, ..., w_n)$ is the same for all the sequencies $(pos_1, ..., pos_n)$.

HMM and POS tagging

How to map a POS tagging problem into a HMM

The above problem

$$(t_1, ..., t_n) = argmax_{pos_1,...,pos_n} P(pos_1, ..., pos_n | w_1, ..., w_n)$$

can be also written (Bayes law) as:

$$(t_1, ..., t_n) = argmax_{pos_1,...,pos_n} P(w_1, ..., w_n | pos_1, ..., pos_n) P(pos_1, ..., pos_n)$$

HMM and POS tagging

The HMM Model of POS tagging:

- **HMM States are mapped into POS tags** (t_i), so that $P(t_1,...,t_n) = P(t_1)P(t_2|t_1)...P(t_n|t_{n-1})$
- **HMM Output symbols are words**, so that $P(w_1,...,w_n|t_1,...,t_n) \quad = \quad \prod_{i=1}^{n} P(w_i|t_i)$
- Transitions represent moves from one word to another

Note that *the Markov assumption is used*

- to model probability of a tag in position i (i.e. t_i) only by means of the preceeding part-of-speech (i.e. t_{i-1})
- to model probabilities of words (i.e. w_i) based only on the tag (t_i) appearing in that position (i).

HMM and POS tagging

The final equation is thus:

$$
\begin{aligned}
(t_1, ..., t_n) = \quad & argmax_{t_1,...,t_n} P(t_1, ..., t_n | w_1, ..., w_n) = \\
& argmax_{t_1,...,t_n} \prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1})
\end{aligned}
$$

Fundamental Questions for HMM in POS tagging

1. Given a model what is the probability of an output sequence, O:
   Computing Likelihood.
2. Given a model and an observable output sequence O (i.e. words), how to determine the sequence of states $(t_1, \ldots, t_n)$ such that it is the best explanation of the observation O:
   Decoding Problem
3. Given a sample of the output sequences and a space of possible models how to find out the best model, that is the model that best explains the data:
   how to estimate parameters?

Fundamental Questions for HMM in POS tagging

- 1. Not much relevant for POS tagging, where $(w_1, ..., w_n)$ are always known.
  Trellis and dynamic programming technique.
- 2. (Decoding) Viterbi Algorithm for evaluating $P(W|O)$.
  Linear in the sequence length.
- 1. Baum-Welch (or Forward-Backward algorithm), that is a special case of Expectation Maximization estimation.
  Weakly supervised or even unsupervised.
  Problems: Local minima can be reached when initial data are poor.

HMM and POS tagging

Advantages for adopting HMM in POS tagging

- An elegant and sound theory
- Training algorithms:
  - Estimation via EM (Baum-Welch)
  - Unsupervised (or possibly weakly supervised)
- Fast Inference algorithms: Viterbi algorithm
  Linear wrt the sequence length ($O(n)$)
- Sound methods for comparing different models and estimations
  (e.g. cross-entropy)

Forward algorithm

In computing the likelihood $P(O)$ of an observation we need to sum up the probability of all paths in a Markov model. Brute forse computation is not applicable in most cases.
The forward algorithm is an application of dynamic programming.

Forward algorithm

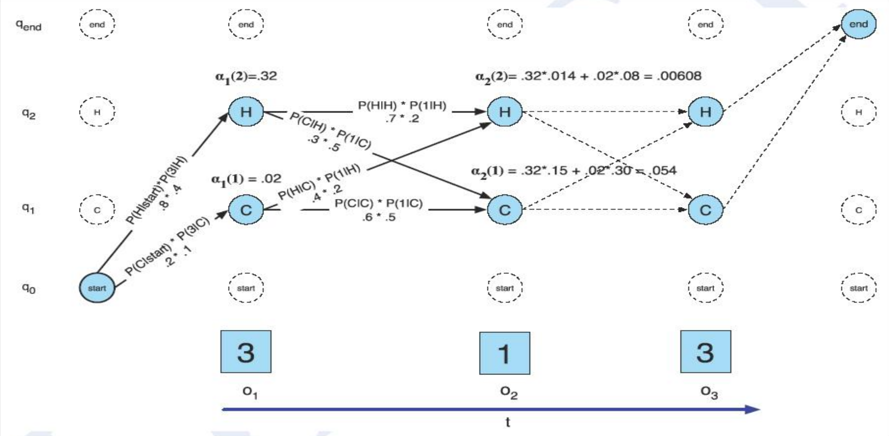


Figure 6.6 The forward trellis for computing the total observation likelihood for the ice-cream events *3 1 3*. Hidden states are in circles, observations in squares. White (unfilled) circles indicate illegal transitions. The figure shows the computation of $\alpha_t(j)$ for two states at two time steps. The computation in each cell follows Eq˙ 6.11: $\alpha_t(j) = \sum_{i=1}^{N-1} \alpha_{t-1}(i) a_{ij} b_j(o_t)$. The resulting probability expressed in each cell is Eq˙ 6.10: $\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$.

HMM and POS tagging: Forward Algorithm

function FORWARD(*observations* of len *T*, *state-graph*) **returns** *forward-probability*

num-states ← NUM-OF-STATES(*state-graph*)
Create a probability matrix *forward[num-states+2,T+2]*
forward[0,0] ← 1.0
for each time step *t* **from** 1 **to** *T* **do**
  for each state *s* **from** 1 **to** *num-states* **do**
    $forward[s,t] \leftarrow \sum_{1 \le s' \le num\text{-}states} forward[s', t-1] * a_{s',s} * b_s(o_t)$
return the sum of the probabilities in the final column of *forward*

Figure 6.8 The forward algorithm; we've used the notation *forward*$[s,t]$ to represent $\alpha_t(s)$.

1. Initialization:

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N \tag{6.12}$$

2. Recursion (since states 0 and N are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^{N-1} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 < j < N, 1 < t < T \tag{6.13}$$

3. Termination:

$$P(O|\lambda) = \alpha_T(N) = \sum_{i=2}^{N-1} \alpha_T(i)\, a_{iN} \tag{6.14}$$

Viterbi algorithm

In decoding we need to find the most likely state sequence given an observation O. The Viterbi algorithm follows the same approach (dynamic programming) of the Forward.

Viterbi scores are attached to each possible state in the sequence.

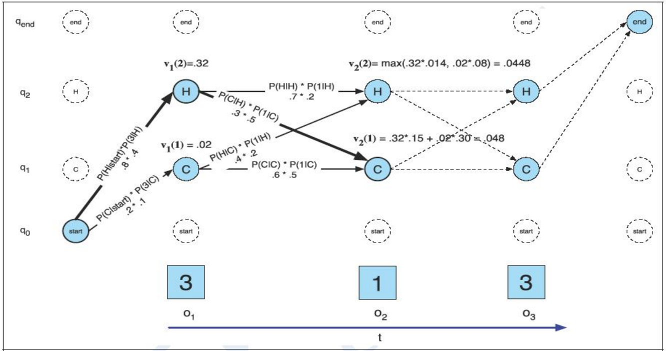


Figure 6.9 The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events *3 1 3*. Hidden states are in circles, observations in squares. White (unfilled) circles indicate illegal transitions. The figure shows the computation of $v_t(j)$ for two states at two time steps. The computation in each cell follows Eq˙ 6.10: $v_t(j) = \max_{1\leq i\leq N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$ The resulting probability expressed in each cell is Eq˙ 6.16: $v_t(j) = P(q_0, q_1, \ldots, q_{t-1}, o_1, o_2, \ldots, o_t, q_t = j|\lambda)$.

HMM and POS tagging: the Viterbi Algorithm

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

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step *t* **from** 1 **to** *T* **do**
  for each state *s* **from** 1 **to** *num-states* **do**
    $viterbi[s,t] \leftarrow \max_{1 \leq s' \leq num\text{-}states} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{1 \leq s' \leq num\text{-}states} viterbi[s',t-1] * a_{s',s}$
Backtrace from highest probability state in final column of *viterbi[]* and return path

Figure 6.10 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence. Note that states 0 and N+1 are non-emitting *start* and *end* states.

HMM and POS tagging: Parameter Estimation

Supervised methods in tagged data sets:

- Output probs: $P(w_i|p^j) = \frac{C(w_i, p^j)}{C(p^j)}$
- Transition probs: $P(p^i|p^j) = \frac{C(p^i \quad \text{follows} \quad \text{p}^j)}{C(p^j)}$
- Smoothing: $P(w_i|p^j) = \frac{C(w_i, p^j)+1}{C(p^j)+K^i}$
  (see Manning& Schutze, Chapter 6)

HMM and POS tagging: Parameter Estimation

Unsupervised (few tagged data available):

- With a dictionary: $P(w_i|p^j)$ are early estimated from D, while $P(p^i|p^j)$ are randomly assigned
- With equivalence classes u_L, (Kupiec92):
  $$P(w^i|p^L) = \frac{\frac{1}{|L|}C(u^L)}{\sum_{u_{L'}} \frac{C(u^{L'})}{|L'|}}$$
  For example, if L ={noun, verb} then
  $u_L = \{cross, drive, \ldots\}$

Other Approaches to POS tagging

- Church (1988):
  $\prod_{i=n}^{3} P(w_i|t_i)P(t_{i-2}|t_{i-1},t_i)$ (backward)
  Estimation from tagged corpus (Brown)
  No HMM training
  Performances: $> 95\%$
- De Rose (1988):
  $\prod_{i=1}^{n} P(w_i|t_i)P(t_{i-1}|t_i)$ (forward)
  Estimation from tagged corpus (Brown)
  No HMM training Performance: 95%
- Merialdo et al.,(1992), ML estimation vs. Viterbi training
  Propose an incremental approach: small tagging and then
  Viterbi training
- $\prod_{i=1}^{n} P(w_i|t_i)P(t_{i+1}|t_i,w_i)$???

POS tagging: References

- F. Jelinek, Statistical methods for speech recognition, Cambridge, Mass.: MIT Press, 1997.
- Manning & Schutze, Foundations of Statistical Natural Language Processing, MIT Press, Chapter 6.
- Church (1988), A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text, http://acl.ldc.upenn.edu/A/A88/A88-1019.pdf
- Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.
- Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, IT-13(2), 260-269.
- Parameter Estimation (slides): `http://jan.stanford.edu/fsnlp/statest/henke-ch6.ppt`
- "*Introduction to Information Retrieval*", Christopher D. Manning, Prabhakar Raghavan and Hinrich Schųtze, Cambridge University Press. 2008. Chapter 12. `http://www-csli.stanford.edu/ hinrich/information-retrieval-book.`