

Gli umani processano il linguaggio in modo parziale e incrementale: iniziano ad analizzare delle frasi a livelli successivi: prima che sia finita la frase, e di continuo fanno critiche cercando di prevedere come andrà la frase.
Se queste vengono smentite, c'è una sorta di riebalanzamento.

Le stesse frasi lavorate i punti andranno.

Tecnologie del Linguaggio Naturale

veri task

- Lemmatizzazione
- stemming
- PoS Tagging ← sequenze labeling

↑ è uno dei
task

Parte Prima

! non è un semplice probl. di classificazione (dove le RN vennero forte), qui vuoi sapere a che categ. appartiene una sequenza di ogg., e cosa c'è prima e cosa c'è dopo

Lezione n. 03

Part of Speech (PoS) Tagging

08 Marzo 2021

PoS tagging



<u>Input:</u>	Plays	well	with	others
<u>Ambiguity:</u>	NNS/VBZ	UH/JJ/NN/RB	IN	NNS
<u>Output:</u>	Plays/VBZ	well/RB	with/IN	others/NNS

il compito
del PoS è
rimuovere questa
ambiguità

ha diversi PoS

Uses:

- Text-to-speech (how do we pronounce “lead”?)
- Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
- As input to or to speed up a full parser
- Moreover, if you know the tag, you can back off to it in other tasks

PoS tagging performance

How many tags are correct? (Tag accuracy)

- About 97% currently → anche 98

→ ve salvato nel rispetto alla performance (neanche gli umani raggiungono il 100%), ma rispetto alla baseline

- ! • But baseline is already 90%: Baseline is performance of stupidest possible method

- Tag every word with its most frequent tag

→ e se non entro? Ne crea 1 rendendo + stupido
il mio algoritmo (ne faccio solo il primo passo)

- Tag unknown words as nouns

- Partly easy because

- Many words are unambiguous

- You get points for them (the, a, etc.) and for punctuation marks!

↑
closed class

PoS tagging performance

How many tags are correct? (**Tag accuracy**)

- About 97% currently
- But baseline is already 90%. **Baseline** is performance of stupidest possible method
 - Tag every word with its most frequent tag
 - Tag unknown words as nouns
- Partly easy because
 - Many words are unambiguous
 - You get points for them (the, a, etc.) and for punctuation marks!

!

Baseline

How difficult is PoS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words, e.g. that
 - *I know that he is honest* = IN
 - *Yes, that play was nice* = DT
 - *You can't go that far* = RB
- 40% of the word tokens are ambiguous

How difficult is PoS tagging?

- About 11% of the word **types** in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words, e.g. that
 - *I know that he is honest* = IN
 - *Yes, that play was nice* = DT
 - *You can't go that far* = RB
- 40% of the word **tokens** are ambiguous

Penn TreeBank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	+%, &
CD	cardinal number	<i>one, two, three</i>	TO	“to”	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential ‘there’	<i>there</i>	VB	verb, base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb, past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb, gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb, past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb, non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb, 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, singular	<i>IBM</i>	\$	dollar sign	\$
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	#
PDT	predeterminer	<i>all, both</i>	“	left quote	‘ or “
POS	possessive ending	<i>'s</i>	”	right quote	’ or ”
PRP	personal pronoun	<i>I, you, he</i>	(left parenthesis	[, (, {, <
PRP\$	possessive pronoun	<i>your, one's</i>)	right parenthesis],), }, >
RB	adverb	<i>quickly, never</i>	,	comma	,
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	. ! ?
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	: ; ... --
RP	particle	<i>up, off</i>			

How Hard is POS Tagging?

Ambiguity

	87-tag Original Brown	45-tag Treebank Brown	
Unambiguous (1 tag)	44,019	38,857	<i>80% → 38,857</i>
Ambiguous (2–7 tags)	5,490	8844	<i>word type (category) sopra parole del dizionario</i>
Details:			
2 tags	4,967	6,731	
3 tags	411	1621	
4 tags	91	357	
5 tags	17	90	
6 tags	2 (<i>well, beat</i>)	32	
7 tags	2 (<i>still, down</i>)	6 (<i>well, set, round, open, fit, down</i>)	
8 tags		4 ('s, half, back, a)	
9 tags		3 (<i>that, more, in</i>)	

le 3 parole + ambigue

Outline

algoritmi
↓

1 algoritmo simbolico

● Rule-Based Tagger: ENGTWOL (ENGLISH TWO Level

analysis)

↓
2 Step:
1 prendo tutte le regole
2 disambiguo

● Stochastic Tagger

2. HMM ← statistica con linguistica semplice

3. MEMM ← " " " " + completa

- PoS tagging conclusion

Outline

- Rule-Based Tagger: ENGTWOL (ENGLISH TWO Level analysis)
- Stochastic Tagger
 - HMM
 - MEMM
- PoS tagging conclusion

① Rule-Based Tagging

Basic Idea:

- ① Assign all possible tags to words (morphological analysis!)
- ② Remove tags according to set of rules of type. For example:
 - if word+1 is an adj, adv, or quantifier and the following is a sentence boundary and word-1 is not a verb like "consider"
 - then eliminate non-adv
 - else eliminate adv.
- Typically more than 1000 hand-written rules, but it may be machine-learned.

Non rimuove必然mente le ambiguità, ma le faune non leggono
↳ "la vecchia porta la sbarra" rimarrebbe con tutte e 2 le sue interpretazioni anche dopo l'algoritmo

Sample ENGTWOL Lexicon

metre
run
morph-it

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

Stage 1 of ENGTWOL Tagging

Run words through a morphological analyzer to get all parts of speech.

Example: *Pavlov had shown that salivation ...*

Pavlov	PAVLOV N NOM SG PROPER
had	HAVE V PAST VFIN SVO
	HAVE PCP2 SVO
shown	SHOW PCP2 SVOO SVO SV
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	N NOM SG

Stage 2 of ENGTWOL Tagging

Apply constraints in a negative way

Example: Adverbial “that” rule -> Given input: “that”

IF (and

(+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

Outline

- Rule-Based Tagger: ENGTWOL (ENGLISH TWO Level analysis)
 - Stochastic Tagger
 - HMM *"etichettatori nascosti"*
 - MEMM
 - PoS tagging conclusion
- hidden Markov Model*
- ci sono dei parametri nascosti che vengono valutati sulla base di parametri espliciti → ci sono degli osservabili (sequenze di parole) e stati Markoviani da stimare

POS Tagging as Sequence Labelling

- We are given a sentence (an “observation” or “sequence of observations”)
 - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view
 - Consider all possible sequences of tags
 - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \dots w_n$

Terminology

- 1. • **Modelling** -> give a formal model

- 2. • **Learning** -> an algorithm for setting the parameters of the model

- 3. • **Decoding** -> algorithm for applying the model in order to compute results

Modelling HMMs

- We want, out of all sequences of n tags $t_1 \dots t_n$ the single tag sequence such that

$P(t_1 \dots t_n | w_1 \dots w_n)$ is highest.

$\rightarrow P(NVN | PaF)$
 ↑↑↑
 nube
 subo
 "Paolo una
 francese"

Modello il prob. come
 un probl. probabilistico,
 in cui voglio trovare la
 sequenza cercando quella
 che ha la maggiore
 probabilità

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

- Hat ^ means “our estimate of the best one”
- Argmax_x f(x) means “the x such that f(x) is maximized”

la sequenza che rende
 massimo argmax

argmax $P(t_1^n | PaF) \rightarrow NVN$
 ↑
 in output

Modelling HMMs

- This equation should give us the best tag sequence

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

è una distribuz. di probabilità

- But how to make it operational? How to compute this value?
- Intuition of Bayesian inference:

- Use Bayes rule to transform this equation into a set of probabilities that are easier to compute (and give the right answer)

si vogliono fare delle approssimazioni

Using Bayes Rule

non c'è approssimazione!

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

"probabilità x condizionata da y"

"probabilità a priori di x"

"probabilità di verosimiglianza"

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

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

! sto invertendo il problema

"la probabilità di parola data il tag"

• è + facile da calcolare

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

language model:
la probab. delle parole

~~P(t₁ⁿ)~~

• ponendo al denominatore non condizionare il numeratore

Io posso fare solo le variazioni su cui varia argmax e' t_1^n , il prob. non ci sono le parole.
 $P(w_1^n)$ non influenza la sequenza che maximizza questa prob.

al denominatore per qualsiasi sequenza di tag, avrò sempre lo stesso valore
il denominatore NON contiene i tag

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

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

la sequenza di Tag che minimizza questo è la stessa
sequenza che minimizza questo

me è una costante
Questo probabilità NON è sempre 1: la prob. di "Paolo Paolo
Paolo Paolo" è molto + bassa di "Paolo amo Francesco"

è la prob. a priori di una sequenza di parole (che interessa
ad esempio nella speech recognition)

$$\frac{\dots P(\text{nome verbo nome})}{P(\text{Paolo amo Fre})} \quad \text{oppure} \quad \frac{\dots P(\text{articolo verbo articolo})}{P(\text{Paolo amo Fre})} \quad \text{oppure} \quad \frac{\dots P(\text{nome nome nome})}{P(\text{Paolo amo Fre})}$$

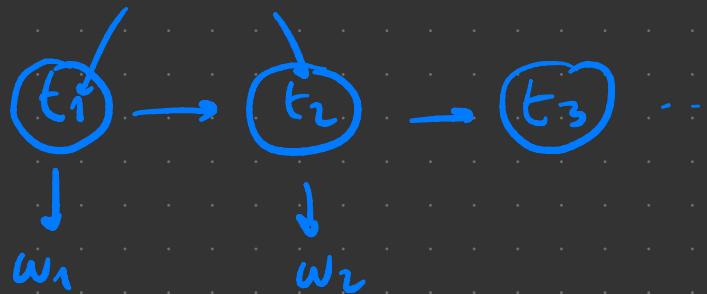
quindi " $P(\text{Paolo amo Fre})$ " si può eliminare

Ma perché si vuole arrivare a

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

Perché quelle z prob. sono + facili da calcolare, in particolare da approssim.

variabili nascoste

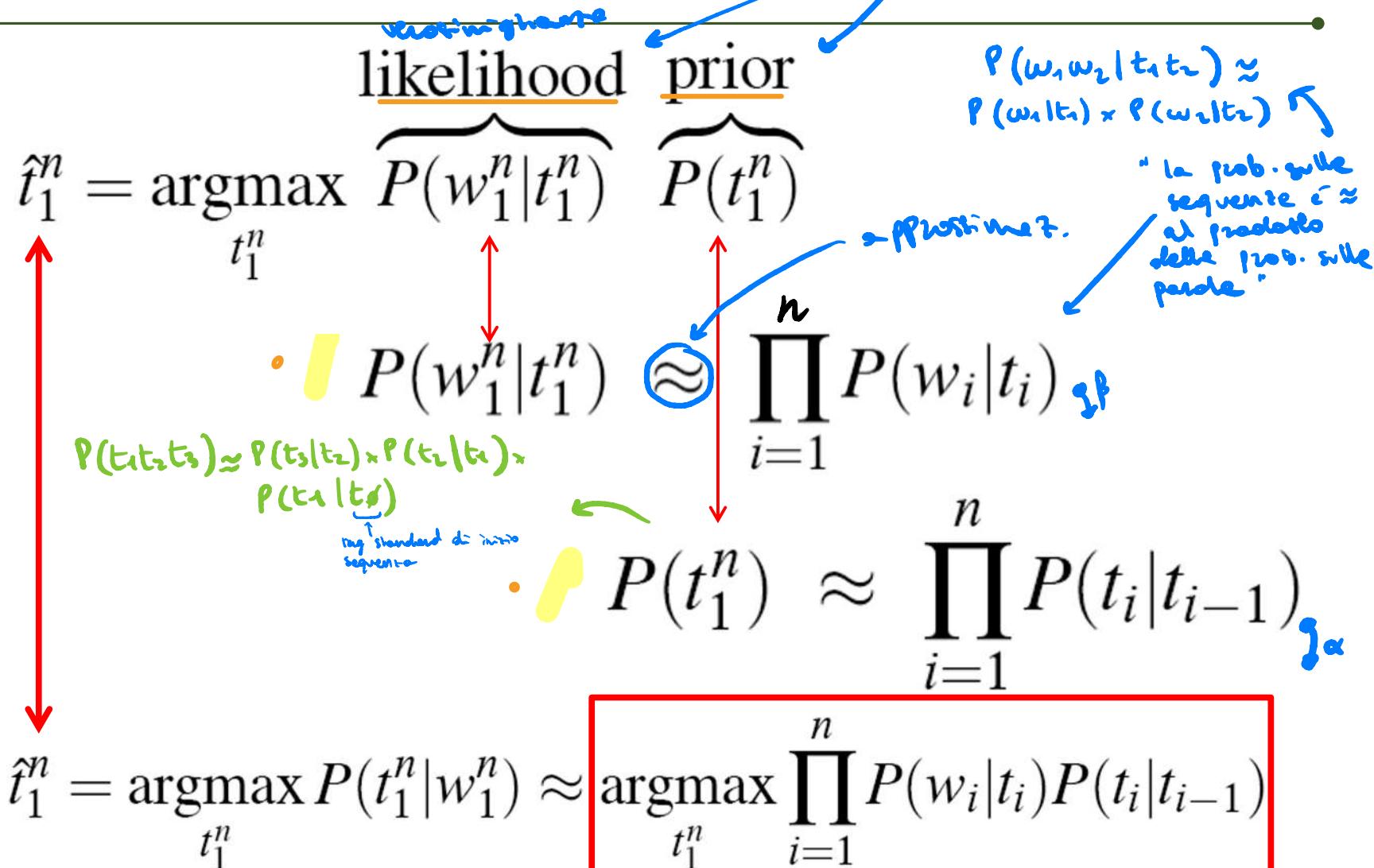


] modello nascosto

] parole in output:
osservabili

Likelihood and Prior

ipotesi Markoviane



2a

Learning Two Kinds of Probabilities: PoS->PoS

Tag transition probabilities $P(t_i|t_{i-1})$

- Determiners likely to precede adjs and nouns
 - That/DT flight/NN
 - The/DT yellow/JJ hat/NN
- So we expect $P(NN|DT)$ and $P(JJ|DT)$ to be high
- Compute $P(NN|DT)$ by counting in a labeled corpus: Learning

abilità un tagset, n' fa annotare a un "linguista" le sequenze → conosce un labeled corpus



probab. di
riunione

→ 3rd

→ 3rd

!.

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \quad \begin{array}{l} \text{→ "tag segue il precedente"} \\ \text{C(t_{i-1})} \end{array}$$

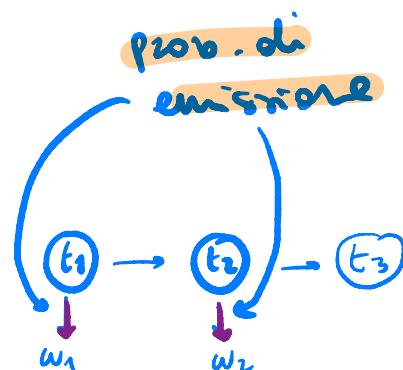
$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

nel ~50% dei casi i DT sono seguiti da nomi

Learning Two Kinds of Probabilities: PoS->word

Word likelihood probabilities $P(w_i|t_i)$

- VBZ (3sg Pres Verb) likely to be “is”
- Compute $P(is|VBZ)$ by counting in a labeled corpus: Learning



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

quante volte “is” è taggato
come verbo

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

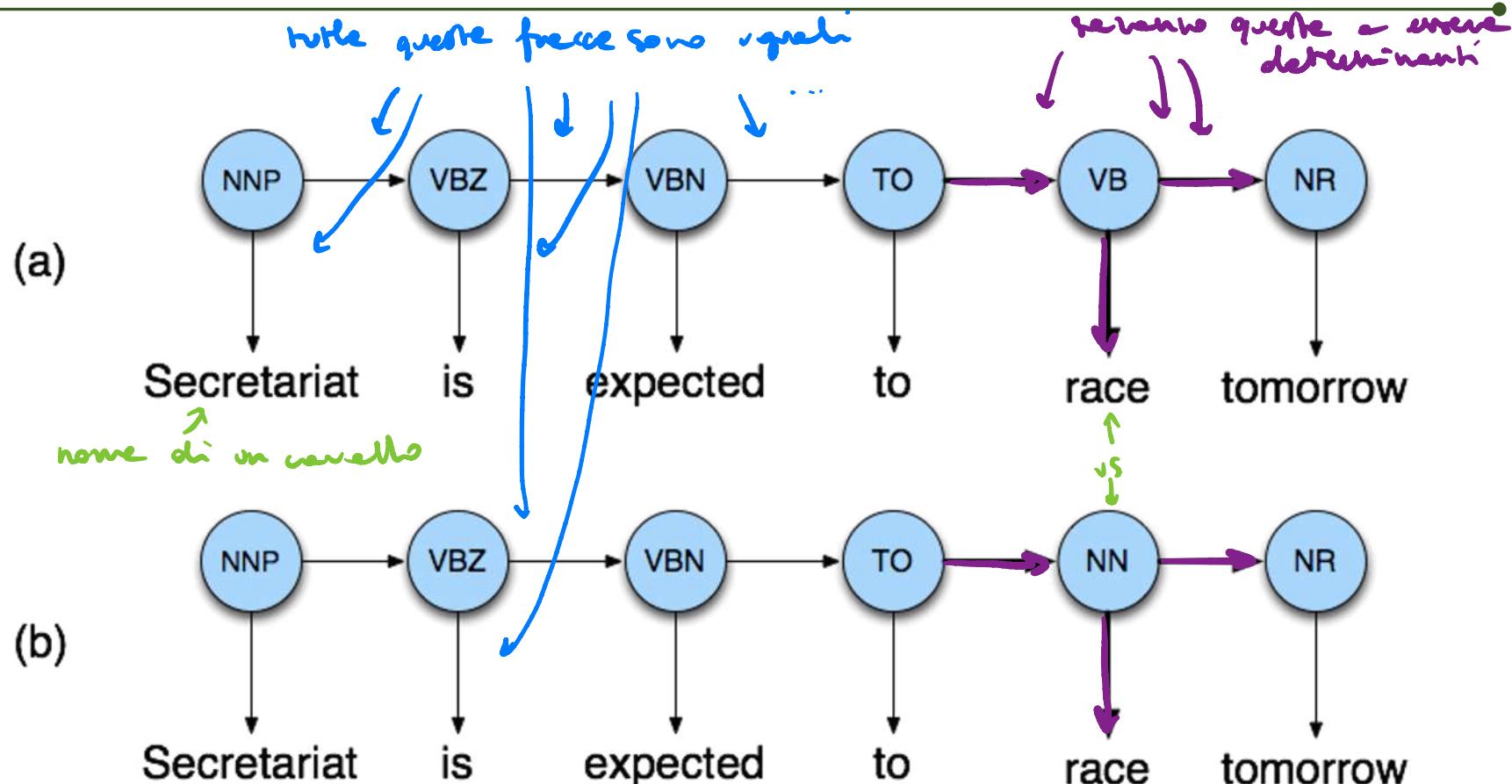
le prob. che un verbo sia “is”

Example: The Verb “race”

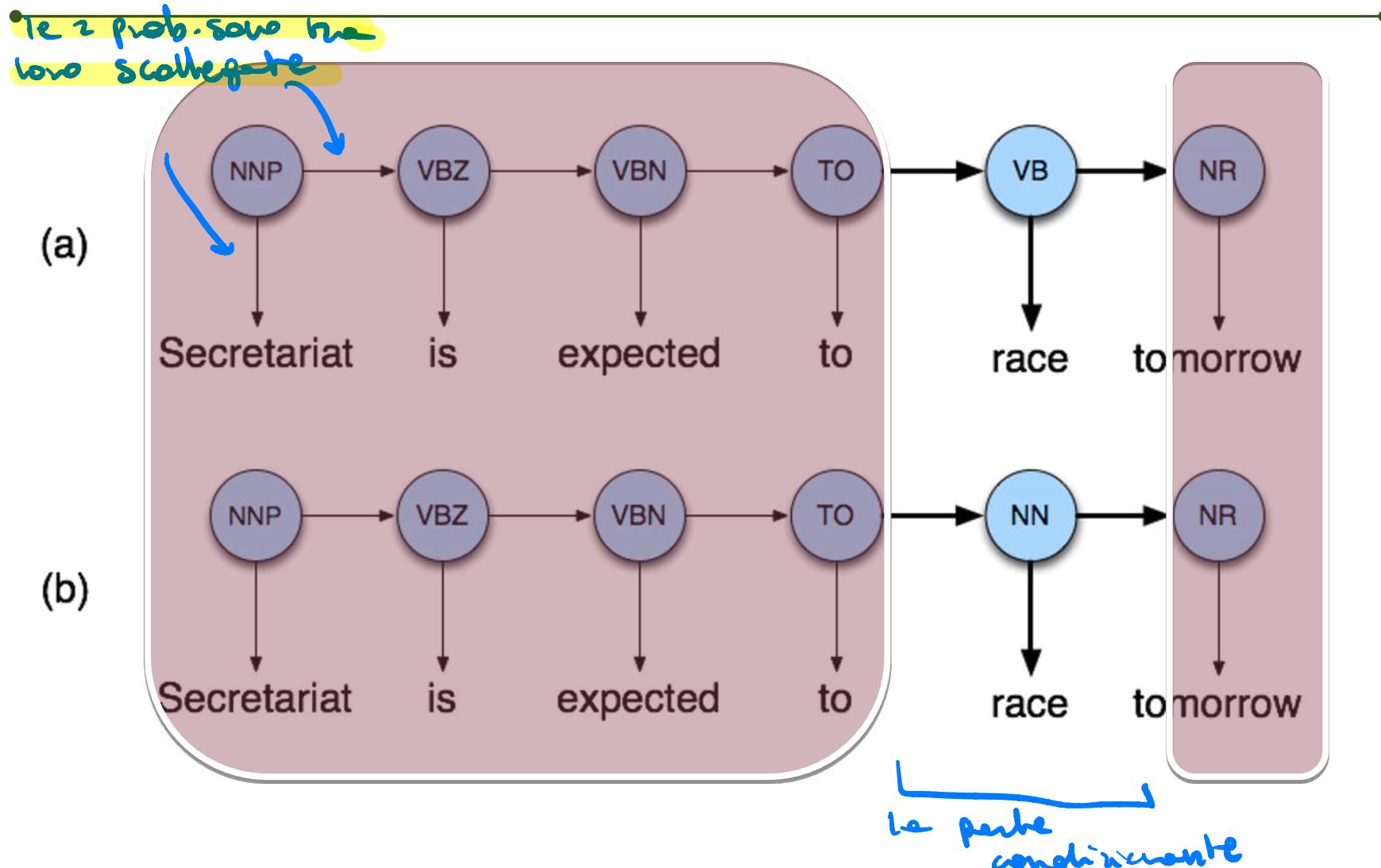
- Secretariat_NNP is_VBZ expected_VBN to_TO race_VB tomorrow_NR
- People_NNS continue_VB to_TO inquire_VB the_DT reason_NN for_IN the_DT race_NN for_IN outer_JJ space_NN

How do we pick the right tag?

Disambiguating “race”



Disambiguating “race”



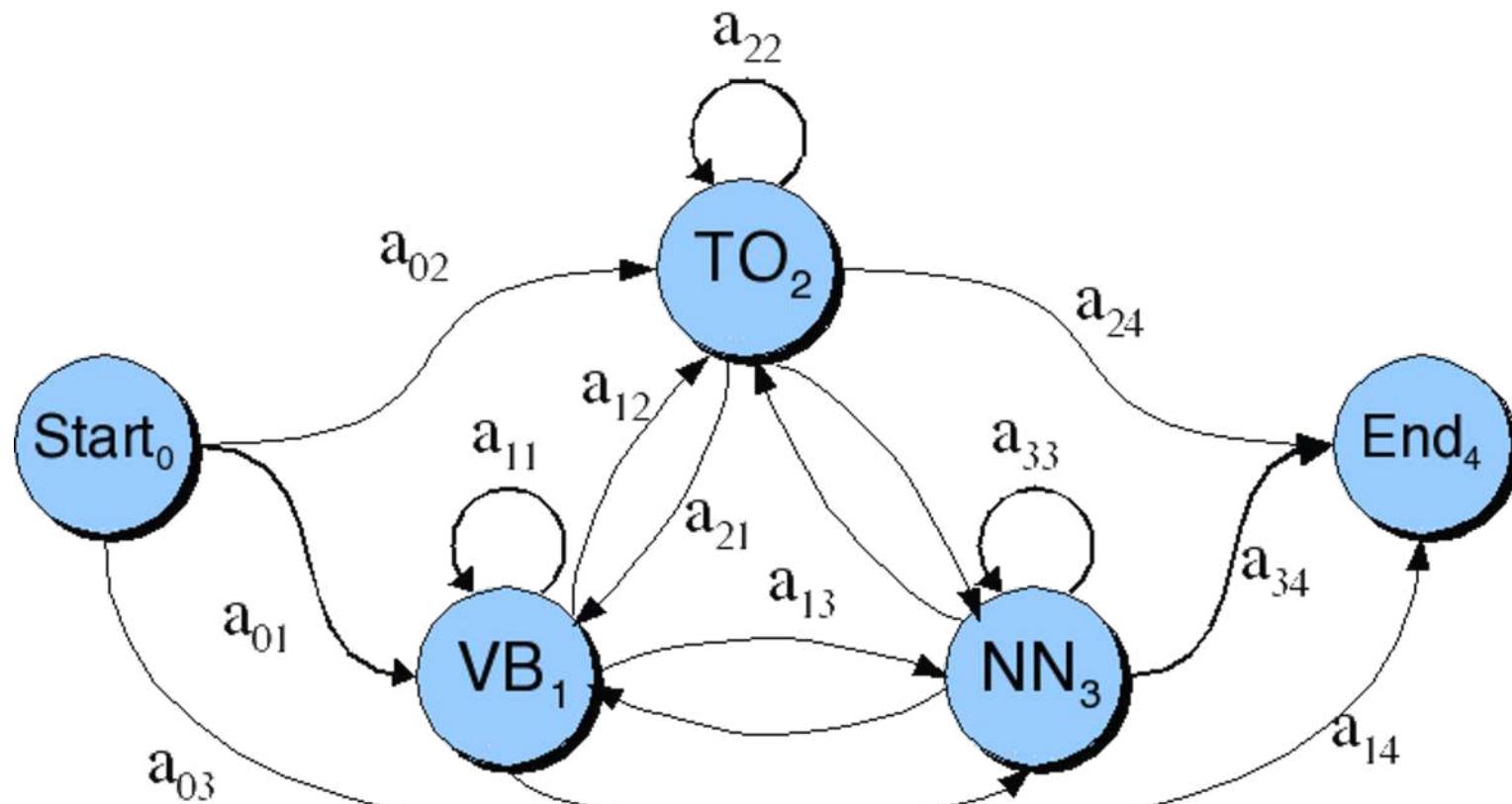
Example

- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(race|NN) = .00057$
- $P(race|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$
- $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(race|NN)=.0000000032$

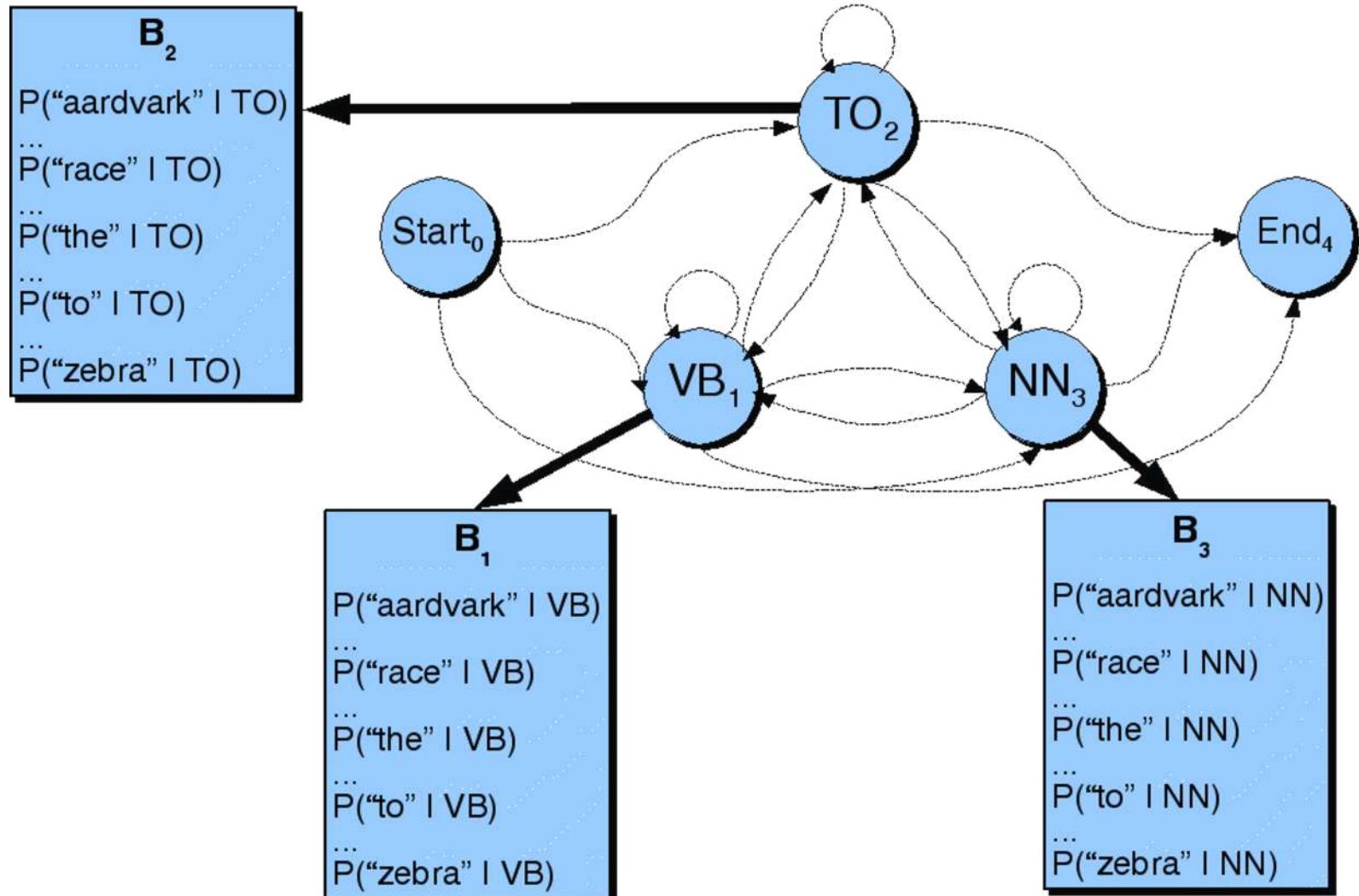
← ha una prob. + altre

So we (correctly) choose the verb tag for “race”

POS Transition Probabilities



Observation Likelihoods





Decoding HMMs

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

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

- We could just enumerate all paths given the input and use the model to assign probabilities to each.

Example: Janet will back the bill

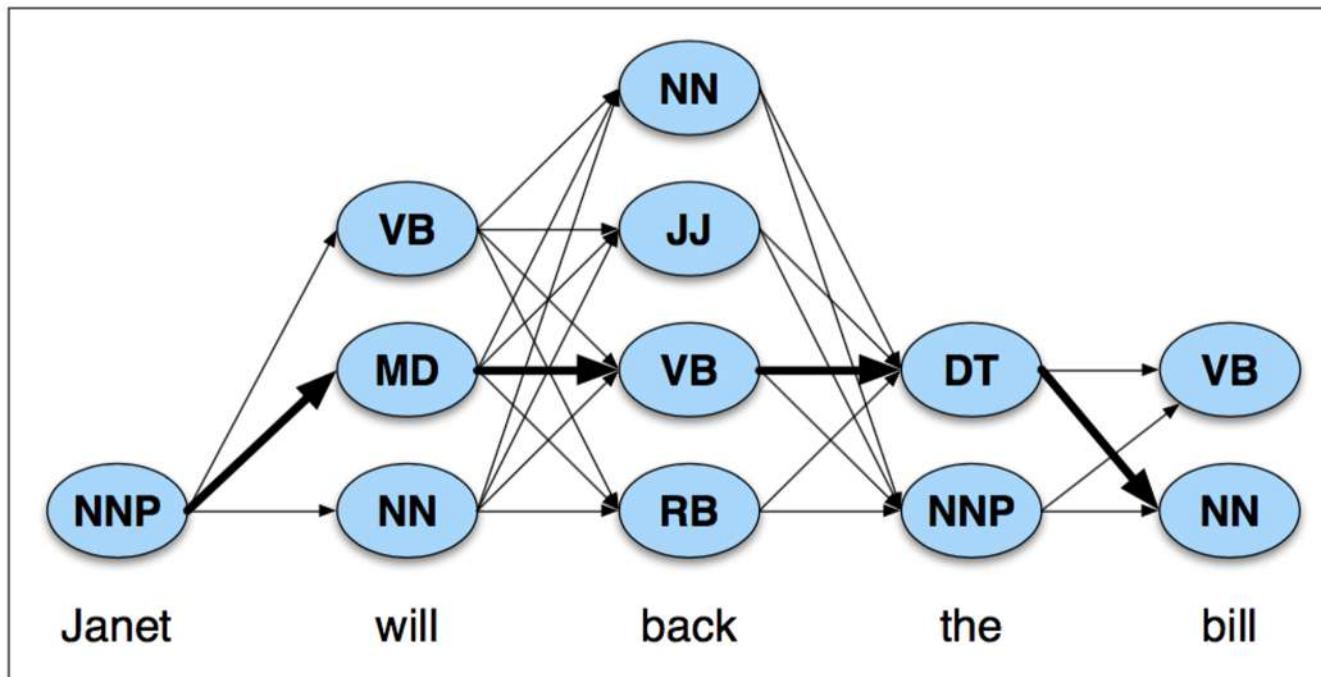
Janet/NNP will/MD back/VB the/DT bill/NN

	NNP	MD	VB	JJ	NN	RB	DT
<i>< s ></i>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Example: Janet will back the bill

Janet/NNP will/MD back/VB the/DT bill/NN



Question

- If there are 30 or so tags in the Penn set
- And the average sentence is around 20 words...
- How many tag sequences do we have to enumerate to argmax over in the worst case scenario?

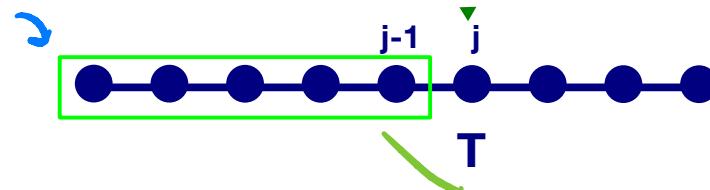
↓ 30^{20}

"se ho calcolato qualcosa, invece di bilanciare via, lo conservo"
Lo memorizzazione

Dynamic Programming idea

- Consider a state sequence (tag sequence) that ends at state j with a particular tag T .
- The probability of that tag sequence can be broken into two parts
 - The probability of the BEST tag sequence up through $j-1$
 - Multiplied by the transition probability from the tag at the end of the $j-1$ sequence to T (and the observation probability of the word given tag T)

differisce solo da quanto c'era prima, non da quanto c'è dopo
prefisso



Possette di passare da una complessità esponenziale a una polinomiale

Markov assumption

“Punto una francese dolcemente”
NV N AV oppure NN N Adj?
avverbio aggettivo
I'ho già calcolato,
me lo farò e non lo
rifacerò?

$P(\text{flies}|N) P(\text{like}|V) P(N|V)$

>

$P(\text{flies}|V) P(\text{like}|V) P(V|V)$

follows:

$P(\text{flies}|N) P(\text{like}|V) P(N|V) P(a|\text{ART}) P(V|\text{ART})$

>

$P(\text{flies}|V) P(\text{like}|V) P(V|V) P(a|\text{ART}) P(V|\text{ART})$

Store ONLY the MAX value and its path! -> $O(D \times |V|^2)$

Viterbi Summary

- Create an array matrix
 - With columns corresponding to inputs
 - Rows corresponding to possible states -> (PoS_TAGS + S_{ini} + S_{fin})
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, and not all paths.

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

prob. di Viterbi

The Viterbi Matrix

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

- $v_{t-1}(i)$ the **previous Viterbi path probability** from the previous time step
- a_{ij} the **transition probability** from previous state q_i to current state q_j
- $b_j(o_t)$ the **state observation likelihood** of the observation symbol o_t given the current state j

NON mi dà le prob. di tutte le possibili sequenze (tutto orario),
ma solo la prob. della sequenza + probabile



The 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

$$viterbi[s, 1] \leftarrow a_{0,s} * b_s(o_1)$$

$$backpointer[s, 1] \leftarrow 0$$

for each time step t from 2 to T do

for each state s from 1 to N do

$$viterbi[s, t] \leftarrow \max_{s'=1}^N viterbi[s', t-1] * a_{s', s} * b_s(o_t)$$

memorizza
un val.
numerico

$$backpointer[s, t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s', t-1] * a_{s', s}$$

legittima il
val. della colonna
preced. che
rende massima
la prob.
(è un fattore
a una riga
precedente)

$$viterbi[q_F, T] \leftarrow \max_{s=1}^N viterbi[s, T] * a_{s, q_F}$$

; 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 $backpointer[q_F, T]$

applica il modello matem. a tutte le possibilità ma l'affronta: valori di prob.
den prefissi e li memorizza all'interno della matrice

n° di parole
delle nostre
frasi

n° di punti tg

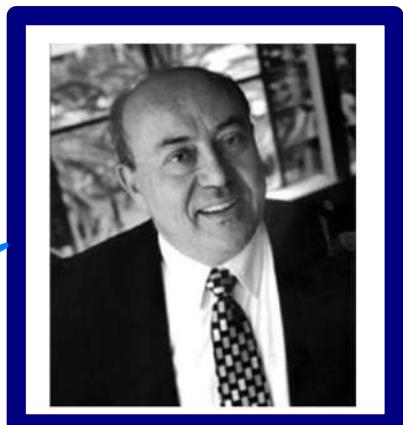
; initialization step

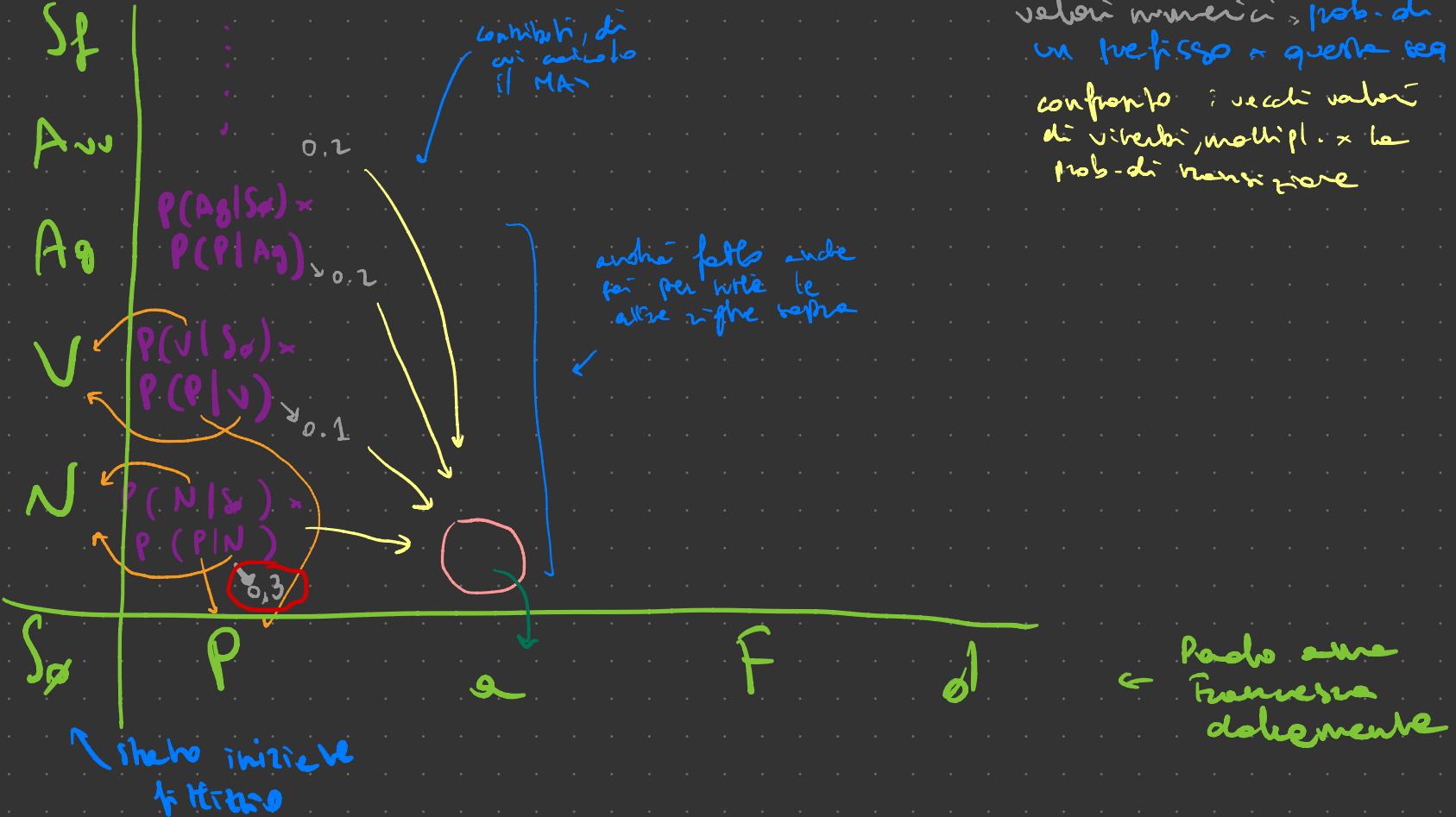
vecchio
nodo

prob. di transiz. tra
un rag e un altro

; recursion step

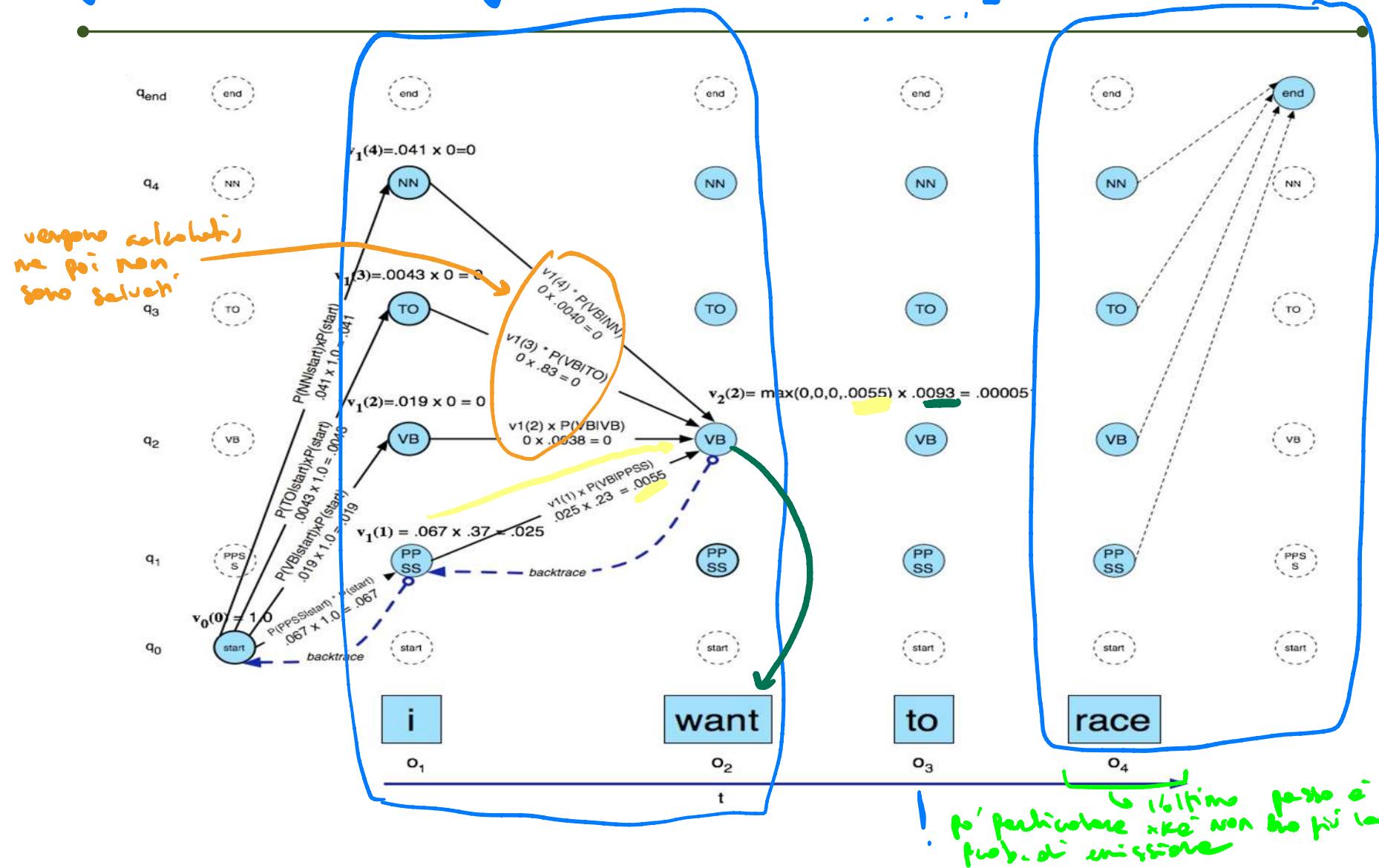
prob. di
emissione



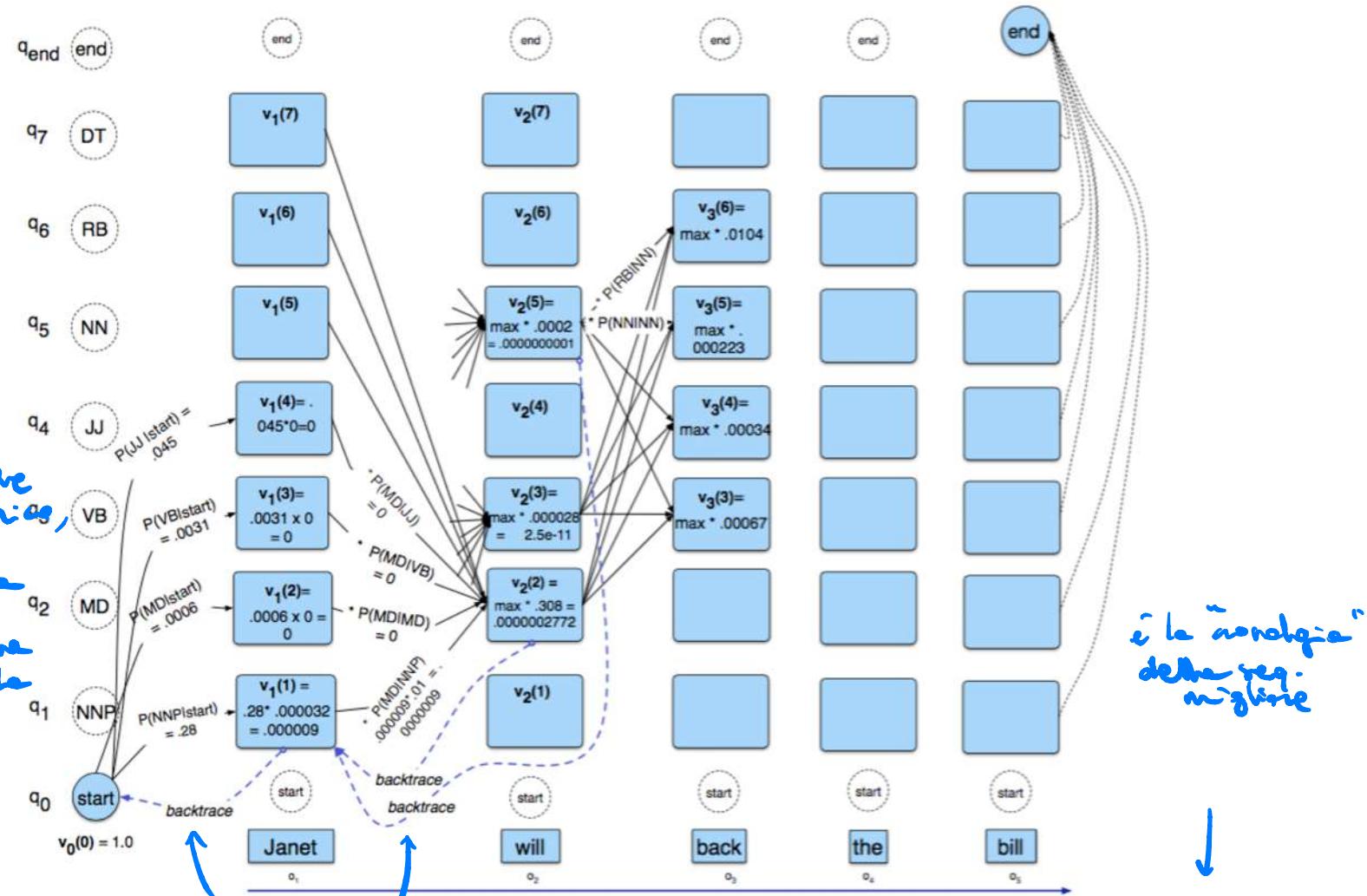


Example

fretta di viterbi: ogni volta mi interessa solo la colonna prec., non anche quelle prime (perché qui la memoria di Markov è 1. Forse anche 2, avrei preso in consid. fin a 2 colonne prima)



Example



Si considera una matrice parallela di back pointers per registrare a qual è stata la
seguita che ha portato ad (magimo) valore finale

Viterbi Summary

- Dynamic programming key is that we need only store the MAX prob path to each cell, and not all paths.
- “Life can only be understood backwards, but must be lived forwards”

Esercizio su HMM

possibile domande
all'ora

- Fare il learning di un modello HMM sul corpus:
 - Paolo/N pesca/V
 - Giovanni/N ama/V i/D cani/N
 - Francesca/N ama/N
determiner/anticab
 - Una/D pesca/N Francesca/A

training set / corpus
(3 tag)

- Fare il decoding, facendo girare a mano l'algoritmo di Viterbi, della frase:
 - Paolo ama Francesca

Extending the HMM Algorithm to Trigrams

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

dimens. delle
frequenze = 1

modelli più
complessi



memoria di
lunghezza 2

$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1}, t_{i-2})$$

diventano prob.
di transizione su
trigrammi

Extending the HMM Algorithm to Trigrams

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

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

Extending the HMM Algorithm to Trigrams

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

se non lo faccio, e ad esempio sbaglio
a scrivere una parola, quella non verrà
trovata nel corpus, e le sue prob. di
emissione, per tutte i
tag, saranno 0,
facendo fallire
tutto

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

Sparsness!!!

SMOOTHING

→ una possibilità è dare
→ questi casi una prob.
di default/di massa

← se i veri sono calcolate
anche prob. impossibili
in un corpus ($V \rightarrow V \rightarrow V$)

in realtà dipende dalla diversità
del corpus, però alcuni sono questi impossibili
in ogni caso. E i weight comunque
vanno aumentati esponenzialmente.

però: • applicare una prob. equidistribuita su tutti i tag: $\hat{P}(t_i) = 1/N$ ($N = \text{nº di tag}$)
• dire che la Prob. c'è per Nome, e per tutti gli altri tag

Sparsity of data -> combining n-grams

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

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

$$\text{Unigrams } \hat{P}(t_i) = \frac{C(t_i)}{N}$$

un'altra tecnica è applicare correzioni di Laplace
a quegli eventi che non si verificherebbero:
gli associa le prob. dei vari eventi che lo compongono

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

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Lambdas -> deleted interpolation algorithm

```
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  $C(t_1, t_2, t_3) > 0$ 
```

```
    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  $\lambda_3$  by  $C(t_1, t_2, t_3)$ 
```

```
        case  $\frac{C(t_2, t_3) - 1}{C(t_2) - 1}$ : increment  $\lambda_2$  by  $C(t_1, t_2, t_3)$ 
```

```
        case  $\frac{C(t_3) - 1}{N - 1}$ : increment  $\lambda_1$  by  $C(t_1, t_2, t_3)$ 
```

```
    end
```

```
end
```

```
normalize  $\lambda_1, \lambda_2, \lambda_3$ 
```

```
return  $\lambda_1, \lambda_2, \lambda_3$ 
```

Problems of HMM

60%

• No Rich Feature Information

- Rich information are required
 - When x_k is complex
 - When data of x_k is sparse

• Example: POS Tagging

• How to evaluate $P(w_k|t_k)$ for unknown words w_k ?

• Useful features

- Suffix, e.g., -ed, -tion, -ing, etc.

- Capitalization

← c'è difficoltà a inserire nel modello, che prevede solo parole e tag. Fatto ricrederà solo + spiegatā

→ verbi inerenti che finiscono in -are -ere -ire

← posso mapparle nel mio modello, ma solo considerandole come 2 parole diverse. Mentre c'è piacerebbe far capire che solo la 3^a lettera è diversa

R 110

on che dei buoni risultati, MA non
e' elastico / flessibile

↓

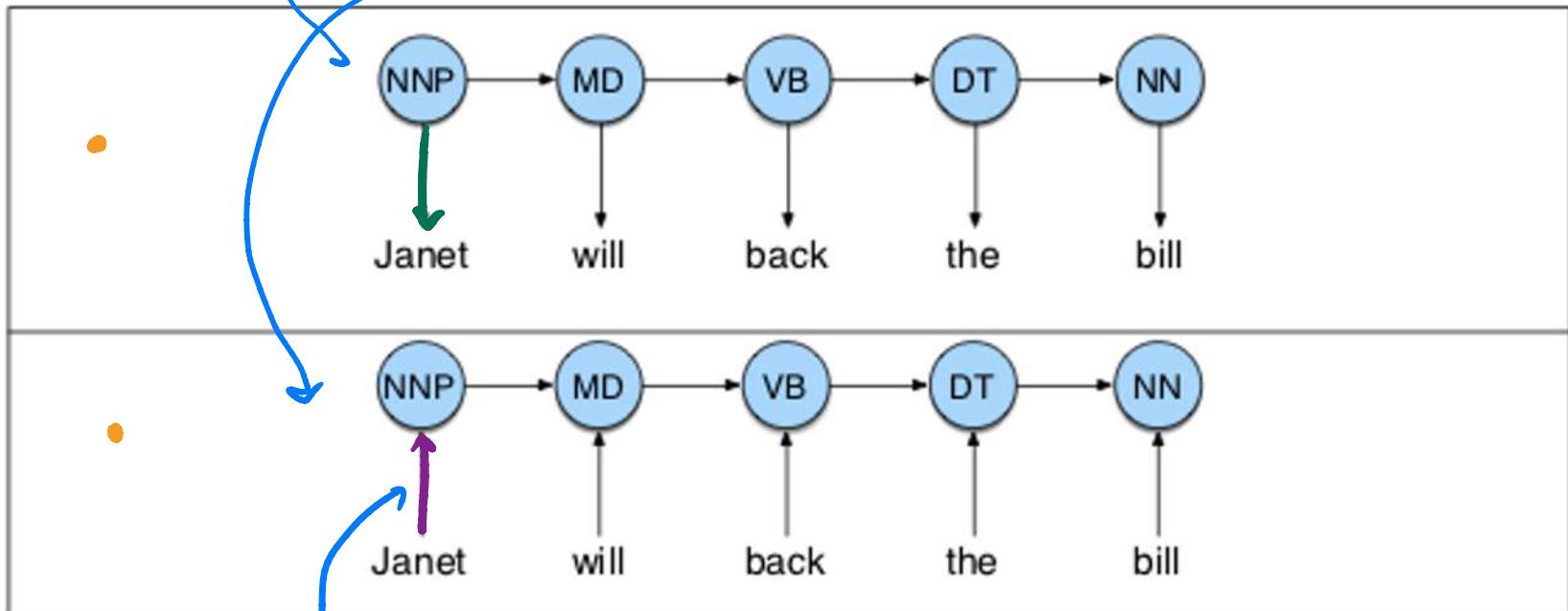
MA non

↓ VS

Outline

- Rule-Based Tagger: ENGTWOL (ENGLISH TWO Level analysis)
 - Stochastic Tagger
 - HMM
 - MEMM  *+ flessibile nella modellaz. delle features*
 - PoS tagging conclusion
- pro: learning molto facile da calcolare
con: il modelling è abbastanza primitivo*

HMM vs. MEMM (MAXIMUM ENTROPY MARKOV MODELS)



+ difficile de calcolare
rispetto a quelle !

HMM vs. MEMM (MAXIMUM ENTROPY MARKOV MODELS)

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} P(W|T)P(T) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(\text{word}_i|\text{tag}_i) \prod_i P(\text{tag}_i|\text{tag}_{i-1})\end{aligned}$$

qui sarebbe estremam. + complesso

VS.

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

fatto aggiungere:
• le parole finite in "-are"
• le parole che sono iniziate con una maiuscula/minuscola
...

Generative vs. Discriminative (Ch. 5&8, 3rd ed.)

- ... **generative model**: a model that is trained to generate the data x from the class y . The likelihood term $P(x|y)$ expresses that we are given the class y and are trying to predict which features we expect to see in the input x .
- A **discriminative model** takes this direct approach, computing $P(y|x)$ by discriminating among the different possible values of the class y rather than first computing a likelihood:

MEMM

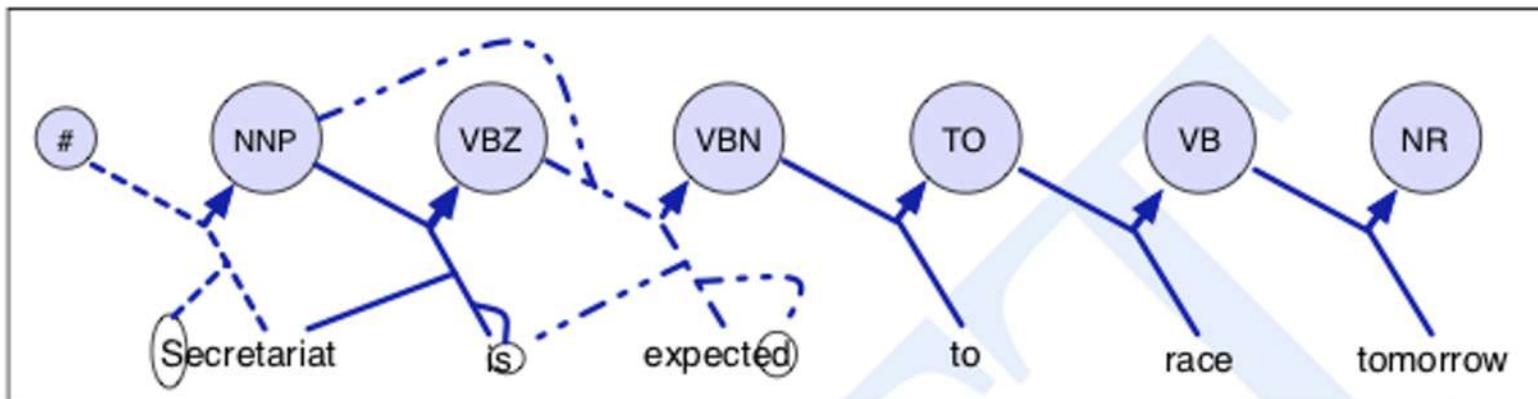


Figure 6.21 An MEMM for part-of-speech tagging, augmenting the description in Fig. 6.20 by showing that an MEMM can condition on many features of the input, such as capitalization, morphology (ending in *-s* or *-ed*), as well as earlier words or tags. We have shown some potential additional features for the first three decisions, using different line styles for each class.

Maximum Entropy modeling

- We may identify a more heterogeneous set of features which contribute in some way to the choice of the current word.
- Maxent combines these features in a probabilistic model. The given features provide a constraint on the model.
- We would like to have a probability distribution which, outside of these constraints, is as uniform as possible – has the maximum entropy among all models that satisfy these constraints.

$$P(w_i | F_1, F_2, \dots, F_n)$$

features
...
che sono cose parole,
che finiscono in -are,

*tutte le prob. delle features che
NON compaiono nella formula sono UNIFORMI*

Maxent P(tag|word)

- Can do surprisingly well just looking at a word by itself:

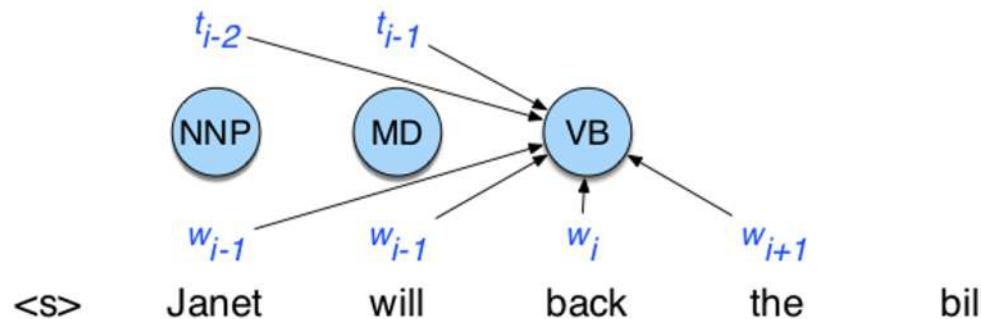
Word	the:	the	DT
Prefixes	unfathomable:	un-	JJ
Suffixes	Importantly:	-ly	RB
Cap.	Meridian:	CAP	NNP
Shapes	35-year:	d-x	JJ

- Then build a classifier to predict tag

Maxent P(tag|word): 93.7% overall / 82.6% unknown

MEMM (Maximum Entropy Markov Model)

- A sequence version of the maximum entropy classifier



$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

MEMM tagger

One step up: also condition on previous tags: **Modelling**

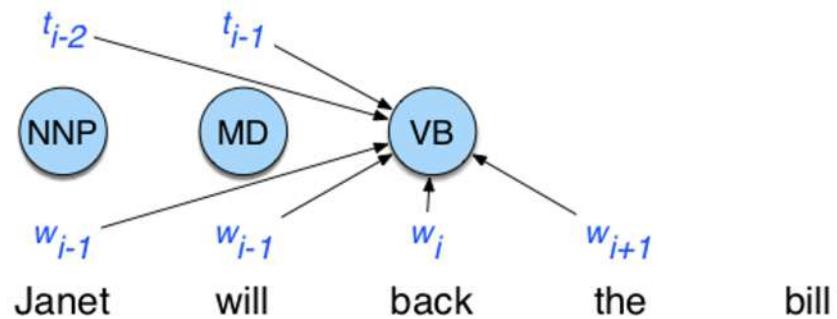
$$\begin{aligned} p(y_1 \dots y_n | x_1 \dots x_n) &= \prod_{i=1}^n p(y_i | y_1 \dots y_{i-1}, x_1 \dots x_n) \\ &= \prod_{i=1}^n p(y_i | y_{i-1}, x_1 \dots x_n) \end{aligned}$$

- **Learning**
 - Train $p(y_i | y_{i-1}, x_1 \dots x_n)$ as a discrete log-linear (MaxEnt) model
- **Decoding**
$$p(y_i | y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$
- This is referred to as an MEMM tagger [Ratnaparkhi 96]

Features

MEMMs

$\langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle$
 $\langle t_i, t_{i-1} \rangle, \langle t_i, t_{i-2}, t_{i-1} \rangle,$
 $\langle t_i, t_{i-1}, w_i \rangle, \langle t_i, w_{i-1}, w_i \rangle \langle t_i, w_i, w_{i+1} \rangle,$



- $t_i = \text{VB}$ and $w_{i-2} = \text{Janet}$
- $t_i = \text{VB}$ and $w_{i-1} = \text{will}$
- $t_i = \text{VB}$ and $w_i = \text{back}$
- $t_i = \text{VB}$ and $w_{i+1} = \text{the}$
- $t_i = \text{VB}$ and $w_{i+2} = \text{bill}$
- $t_i = \text{VB}$ and $t_{i-1} = \text{MD}$
- $t_i = \text{VB}$ and $t_{i-1} = \text{MD}$ and $t_{i-2} = \text{NNP}$
- $t_i = \text{VB}$ and $w_i = \text{back}$ and $w_{i+1} = \text{the}$

More features

- w_i contains a particular prefix (from all prefixes of length ≤ 4)
- w_i contains a particular suffix (from all suffixes of length ≤ 4)
- w_i contains a number
- w_i contains an upper-case letter
- w_i contains a hyphen
- w_i is all upper case
- w_i 's word shape
- w_i 's short word shape
- w_i is upper case and has a digit and a dash (like CFC-12)
- w_i is upper case and followed within 3 words by Co., Inc., etc.

MEMM Modelling

$$P(y|x) = \sum_{i=1}^N w_i f_i$$

Importanza del peso per il valore della feature

$$= w \cdot f$$

peso feature, normalm. booleane
(non c'è la word)

"prob. che un certo output appartenga a una certa classe degli osservabili"



$$p(y=c|x) = p(c|x) = \frac{1}{Z} \exp \sum_i w_i f_i$$

fattore di
normalizzazione/
perturbazione

MEMM Modelling

define le feature che vogliamo

le feature le abbiamo già definite. Come calcolare i giusti testi?

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\ &= \underset{T}{\operatorname{argmax}} \prod_i \frac{\exp \left(\sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)}{\sum_{t' \in \text{tagset}} \exp \left(\sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)}\end{aligned}$$

prob. che si vuole minimizzare

MEMM Learning

- MaxEnt learning -> Multinomial Logistic Regression
- Conditional maximum likelihood estimation: we choose the parameters w that maximize the (log) probability of the y labels in the training data given the observations x.

$$\hat{w} = \operatorname{argmax}_w \sum_j \log P(y^{(j)} | x^{(j)})$$

dove calcol. la prob. di massima verosimiglianza considerando il training data

MEMM Learning

si tratta di calcolare il valore
dei pesi che rendono massime le
prob. w training set

$$\begin{aligned} L(w) &= \sum_i \log P(y^{(j)} | x^{(j)}) \\ &= \sum_j \log \frac{\exp \left(\sum_{i=1}^N w_i f_i(y^{(j)}, x^{(j)}) \right)}{\sum_{y' \in Y} \exp \left(\sum_{i=1}^N w_i f_i(y'^{(j)}, x^{(j)}) \right)} \\ &= \sum_j \log \exp \left(\sum_{i=1}^N w_i f_i(y^{(j)}, x^{(j)}) \right) - \sum_j \log \sum_{y' \in Y} \exp \left(\sum_{i=1}^N w_i f_i(y'^{(j)}, x^{(j)}) \right) \end{aligned}$$

Convex optimization problem, so we use hill-climbing methods like stochastic gradient ascent, L-BFGS (Nocedal 1980, Byrd et al. 1995)

(o non c'è una soluz. esatta, non pu' essere risolto analiticamente)

MEMM Decoding

- Simplest algorithm
 - Greedy: at each step in sequence, select tag that maximizes $P(\text{tag} \mid \text{nearby words, nearby tags})$
- In practice
 - Viterbi algorithm
 - Beam search

$$p(y_i | y_{i-1}, x_1 \dots x_n) = \frac{e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y_i)}}{\sum_{y'} e^{w \cdot \phi(x_1 \dots x_n, i, y_{i-1}, y')}}$$

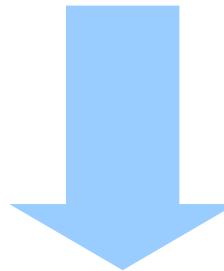
piv' importante capire i motivi che : dettagli matematici ↴



	HMM	vs	MEMM
Modelling	moltò semplice, ma anche debole (non si poss. aggiungere features)		abb. semplice (si poss. agg. features qualitativi)
Training	tecniche moltò brave		severo approssimatore qui stile vere diff.
Decoding		utendo le stesse tecniche (complicate)	
generativo		vs	discriminativo

MEMM Decoding: Viterbi!

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



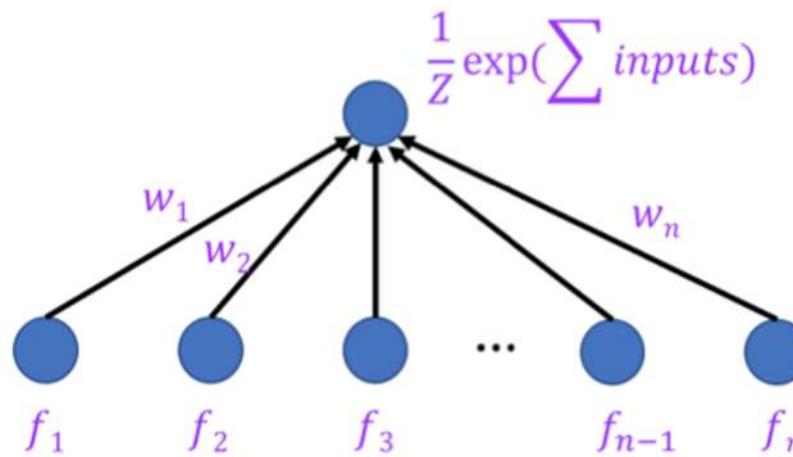
$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(s_j | s_i, o_t) \quad 1 \leq j \leq N, 1 < t \leq T$$

MEMM/LogisticRegression Summary

- model $P(y|x)$ only, have no generative process
- can use arbitrary local/global features, including correlated ones
- can use for classification, or choosing from n-best list (reranking)
- Learning involves iteratively updating the weights, so typically slower than for generative models
- Decoding -> Greedy/Viterbi

Extension to neural network (Sharon Goldwater)

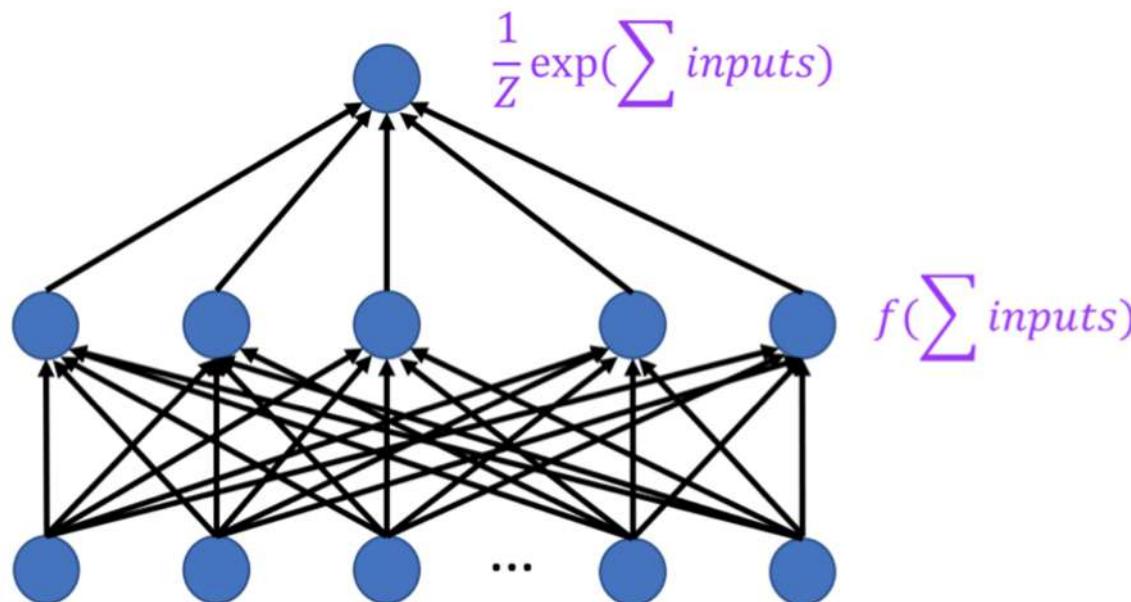
- Logistic regression can be viewed as a building block of neural networks -> perceptron



è un prob1. che associa gli input a quello di una rete neurale

Extension to neural network (Sharon Goldwater)

- Adding a fully-connected layer creates one of the simplest types of neural network: a **multi-layer perceptron** (MLP).



Extension to neural network (Sharon Goldwater)

- Contains one or more hidden layers
 - Each node applies a non-linear function to the sum of its inputs
 - Hidden layers can be viewed as learned representations (embeddings) of the input
- A non-linear classifier: more powerful than logistic regression.
- Also trained using gradient-based methods, but vulnerable to local optima, so can be more difficult to train.
- Like other NNet architectures, really just a complex function computed by multiplying weights by inputs and passing through non-linearities. Not magic.

Outline

- Rule-Based Tagger: ENGTWOL (ENGLISH TWO Level analysis)
- Stochastic Tagger
 - HMM
 - MEMM
- PoS tagging conclusion

POS Tagging Accuracies

- Rough accuracies:

- Baseline: most freq tag: ~90%
 - Trigram HMM: (senza smoothing) ~95%
 - Maxent $P(t|w)$: (senza memoria)
↳ non ho bisogno di vettori 93.7%
 - MEMM tagger: 96.9%
 - Bidirectional MEMM: 97.2%
 - Upper bound: ~98%
- (human agreement)

↑↑↑
dipendono in maniera
cruciale come si modella
lo smoothing delle parole
sensibili |

Tagging Unknown Words

- New words added to (newspaper) language 20+ per month
- Plus many proper names ...
- Increases error rates by 1-2%

Possible approaches for UW:

- Method 1: assume they are nouns
- Method 2: assume the unknown words have a probability distribution similar to words only occurring once in the training set.
- Method 3: Use morphological information, e.g., words ending with -ed tend to be tagged VBN
- Method 4: Dictionary (*morph-it*)
- Method 5: Assume uniform distribution over PoS

Evaluation

- The result is compared with a manually coded “Gold Standard”
- Corpus tagged by-hand = Gold standard
- Usually:
 - Training data: 80%
 - Development data: 10%
 - Test data: 10%

il corpus è diviso
in 3 insiemni

percentuale di parole che rientrano in questo insieme

Evaluation

- Typically accuracy reaches 96-97% on WSJ
 - Train: 02-21 Dev: 00/24 Test: 23
 - Important: 100% is impossible even for human annotators.
- Factors that affects the performance
 - The amount of training data available
 - The tag set
 - The difference between training corpus and test corpus
 - Dictionary
 - Unknown words
 - Domain! -> tweet -> EVALITA 2016 -> Neural Features!!!