CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 11

Natural Language Processing

Part of Speech Tagging

Many slides from: Joshua Goodman, L. Kosseim, D. Klein, D. Jurafsky, M. Hearst, K. McCoy, Y. Halevi, C. Manning, M. Poesio

Outline

- What is POS and POS tagging
  - POS = part of speech
- Why we need POS tagging
- Different Approaches to POS
  1. rule-based tagging
  2. statistical tagging

What is a Part of Speech ?

- Words that behave alike
  - appear in similar contexts
  - perform similar functions in sentences
  - undergo similar transformations
- Terminology
  - **POS** (part-of-speech tag)
  - also called
    - grammatical tag
    - grammatical category
    - syntactic word class

Substitution Test

- Two words belong to the same part of speech if replacing one with another does not change the grammaticality of a sentence

The {sad, big,green, …} dog is barking.

Origin

- Perhaps started with Aristotle (384–322 BCE)
- From Dionysius Thrax of Alexandria (c. 100 BCE) the idea that is still with us
  - 8 main parts of speech
- Those 8 are not exactly the ones taught today
  - Thrax: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
  - School grammar: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection

How Many POS are there?

- A basic set:
  - N(oun), V(erb), Adj(ective), Adv(erb), Prep(osition), Det(erminer), Aux(ilaries), Part(icle), Conj(unction)
- A simple division: open/content vs. closed/function
  - Open: N, V, Adj, Adv
    - new members are added frequently
  - Closed: Prep, Det, Aux, Part, Conj, Num
    - new members are added rarely
- Many subclasses, e.g.
  - eats/V $\Rightarrow$ eat/VB, eat/VBP, eats/VBZ, ate/VBD, eaten/VBN, eating/VBG, ...

POS tagging

- Goal: assign POS tag (noun, verb, …) to text

  The/AT girl/NN put/VBD chairs/NNS on/IN the/AT table/NN.

- What set of parts of speech do we use?
  - various standard tagsets to choose from, some have a lot more tags than others
  - choice of tagset is based on application
  - accurate tagging possible with even large tagsets

Why do POS Tagging?

- Word sense disambiguiaton (semantics)
  - limits the range of meanings: *deal* as noun vs. *deal* as verb
- Speech recognition and synthesis
  - how to recognize/pronounce a word:
  - *content*/**noun** vs. *content*/**adj**
- Stemming: which morphological affixes word can take
  - adverb - *ly* = noun: *friendly* - *ly* = friend
  - cannot apply to adjectives, example: *sly*
- Partial parsing/chunking
  - to find noun phrases/verb phrases
- Information extraction
  - helps identify useful terms and relationships between them

Common Tagged Datasets

- 45 tags in Penn Treebank
- 62 tags in CLAWS with BNC corpus
- 79 tags in Church (1991)
- 87 tags in Brown corpus
- 147 tags in C7 tagset
- 258 tags in Tzoukermann and Radev (1995)

Penn Treebank

- First syntactically annotated corpus
- 1 million words from Wall Street Journal
- Part of speech tags and syntax trees

Important Penn Treebank tags

- 45 tags total

IN	preposition or subordinating conjunct.
JJ	adjective or numeral, ordinal
JJR	adjective, comparative
NN	noun, common, singular or mass
NNP	noun, proper, singular
NNS	noun, common, plural
TO	"to" as preposition or infinitive marker
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle or gerund
VBN	verb, past participle
VBP	verb, present tense, not 3rd p. singular
VBZ	verb, present tense, 3rd p. singular
…	

Verb inflection tags

VBP	base present	*take*
VB	infinitive	*take*
VBD	past	*took*
VBG	present participle	*taking*
VBN	past participle	*taken*
VBZ	present 3sg	*takes*
MD	modal	*can, would*

The entire Penn Treebank tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*’s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Terminology

- Given text

  The cat decided to jump on the couch to play with another cat

- Terminology
  - **Word type**
    - Distinct words in the text (vocabulary)
    - text above has 10 word types
      - *the, cat, decided, to, jump, on, couch, play, with, another*
  - **Word token**
    - any word occurring in the text
    - text above has 13 word tokens

Distribution of Tags

- POS follow typical frequency-based behavior
  - most word types have only one part of speech
  - of the rest, most have two
  - only a small number of word types have lots of parts of speech
    - but these occur with high frequency

Most Word Types not Ambiguous but

	num. word types	
Unambiguous (1 tag)	**35 340**	
Ambiguous (>1 tag)	**4 100**	
2 tags	3760	
3 tags	264	
4 tags	61	
5 tags	12	
6 tags	2	
7 tags	1	*"still"*

- but most word types are rare
- Brown corpus (Francis&Kucera, 1982):
  - 11.5% **word types** are ambiguous (>1 tag)
  - 40% **word tokens** are ambiguous (>1 tag)

Tagging is a Type of Disambiguation

1. Book/**VB** that/**DT** flight/**NN**
   - ***book*** can also be **NN**
   - Can I read a ***book*** on this flight?
2. Does/**VBZ** that/**DT** flight/**NN** serve/**VB** dinner/**NN** ?
   - ***that*** can also be a **complementizer**
   - My travel agent said ***that*** there would be a meal on this flight.

Potential Sources of Disambiguation

1. Lexical information:
   - look up all possible POS for a word in a dictionary
   - *“table”*: {noun, verb} but not a {adj, prep,…}
   - *“rose”*: {noun, adj, verb} but not {prep, ...}
2. Syntagmatic information:
   - some tag sequences are more probable than others:
   - **DET** + **N** occur frequently but **DET**+**V** never occurs
   - **ART**+**ADJ**+**N** is more probable than **ART**+**ADJ**+**VB**
   - Can find the syntagmatic information
     - by talking to the experts
     - or, better, from training corups

Syntagmatic Information from Corpus

- For a is a sequence of tags $\mathbf{t_1, t_2,.., t_k}$ compute

$$\mathbf{P(t_1, t_2,.., t_k)}$$

  - tells us how likely this tag sequence is
  - similar to computing probability of a sequence of words $\mathbf{P(w)}$
  - make the same approximation as before

$$\mathbf{P(t_n | t_1, t_2,.., t_{n-1}) = P(t_n | t_{n-k}...t_{n-1})}$$

  - for computational efficiency, our assumption is

$$\mathbf{P(t_n | t_1, t_2,.., t_{n-1}) = P(t_n | t_{n-1})}$$

POS Tagging Techniques

1. rule-based tagging
   - uses hand-written rules
2. statistical tagging
   - uses probabilities computed from training corpus
     - Charniak
     - Markov Model based

Rule-based POS Tagging

- Step 1: assign each word with all possible tags
  - use dictionary

- Step 2: use if-then rules to identify the correct tag in context (disambiguation rules)

Rule-based POS Tagging: Sample rules

- **ART-V rule**:

  tag ART (article) cannot be followed by a tag V (verb)

  ...the book...

  - the: {ART}
  - book: {N, V} --> {N}

- **N-IP rule**:

  tag N (noun) cannot be followed by tag IP (interrogative pronoun)

  ... man who ...

  - man: {N}
  - who: {RP, IP} --> {RP} relative pronoun

Rule-based Tagger

- using only syntagmatic patterns
  - Green & Rubin (1971)
  - accuracy of 77%
- In addition
  - very time consuming to come up with the rules
  - need an expert in English to come up with the rules

Statistical POS Tagger: Charniak 1993

- Simplest statistical tagger
- From corpus, calculate most probable tag for each word
- that is the one maximizing

  count(word has tag t)/count(word)

- Equivalent to maximizing

  count(word has tag t)

- Charniak tagger assigns most probable POS tag to a word
- Given a word to tag,
  1. for each possible tag t for this word, compute

     count(word has tag t)

  2. choose tag t that maximizes the above

Statistical POS Tagger: Charniak 1993

- Accuracy of 90%
  - contrast with 77% accuracy of the rule-based tagger!
  - evidence of power of statistical over rule-based methods
  - MUCH better than rule based, but not very good...
    - 1 mistake every 10 words
  - funny fact: every word will have only one POS assigned to it
    - book will always be assigned the noun tag
- This tagger is used mostly as baseline for evaluation
- How do we improve it?
  - take the context of the surrounding words into account
  - some sequence of tags are much more likely than others

Statistical Tagger: Markov Model Based

- Tag sentence of words $\mathbf{w_{1,n} = w_1\ w_2\ \dots\ w_n}$
- Denote tag sequence as $\mathbf{t_{1,n} = t_1\ t_2\ \dots\ t_n}$
  - $\mathbf{t_i}$ is a tag for word $\mathbf{w_i}$
- Find the best tagging $\mathbf{t_{1,n}}$ out of all possible taggings
- Two sources of information
  1. $\mathbf{P}(\mathbf{t_{i+1}} | \mathbf{t_i})$: how likely is tag $\mathbf{t_{i+1}}$ after tag $\mathbf{t_i}$
  2. $\mathbf{P}(\mathbf{w_i} | \mathbf{t_i})$: if tag is $\mathbf{t_i}$ how likely word is $\mathbf{w_i}$
     - Example: **P**(book | **verb**) > P(book | **noun**)
       - there are many more nouns than verbs
       - say 1,000 verbs and 10,000 nouns

Markov Model Tagger

- The best tagging is the one that maximizes

$$\mathbf{P(t_{1,n} \mid w_{1,n})}$$

- Hard to estimate directly

- Using Bayes law

$$\mathbf{P(t_{1,n} \mid w_{1,n}) = \frac{P\,(w_{1,n} \mid t_{1,n}\,)P(t_{1,n}\,)}{P(w_{1,n})}}$$

- Bottom does not effect maximization,
  - constant over all possible taggings $\mathbf{t_{1,n}}$

- Find tagging that maximizes

$$\mathbf{P\,(w_{1,n} \mid t_{1,n}\,)P(t_{1,n}\,)}$$

Markov Model Tagger: First Assumption

$$\mathbf{P(w_{1,n} \mid t_{1,n})P(t_{1,n})}$$

- We will make two simplifying assumptions
- First simplifying assumption:
  1. given its tag, probability of word is independent of tags of other words in a sentence:

$$\mathbf{P(w_{1,n} \mid t_{1,n}) = \prod_{i=1}^{n} P(w_i \mid t_i)}$$

  - **P**(book|**verb**) is independent of what are the tags of other words in the sentence
  - Reasonable assumption. For example, if the next tag is **adverb**, does not change much about **P**(book|**verb**)

Markov Model Tagger: First Assumption

$$\mathbf{P}\,(\mathbf{w}_{1,n} \mid \mathbf{t}_{1,n}) = \prod_{i=1}^{n} \mathbf{P}\,(\mathbf{w}_i \mid \mathbf{t}_i) = \mathbf{P}(\mathbf{w}_1 \mid \mathbf{t}_1)\,\mathbf{P}(\mathbf{w}_2 \mid \mathbf{t}_2) \dots \mathbf{P}(\mathbf{w}_n \mid \mathbf{t}_n)$$

- $\mathbf{P}(\mathbf{w}_i \mid \mathbf{t}_k)$ estimated from tagged corpus:

$$\frac{\mathbf{C}(\,\mathbf{w}_i \text{ has tag } \mathbf{t}_k\,)}{\mathbf{C}(\mathbf{t}_k)}$$

- i.e. **P**(book | **verb**) is count of how many times book has tag **verb** divided by how many times tag **verb** occurs in corpus

Markov Model Tagger: Second Assumption

$$P(w_{1,n} | t_{1,n}) P(t_{1,n})$$

2. Each tag depends only on one previous tag:

$$P(t_{1,n}) = \prod_{i=1}^{n} P(t_i | t_{i-1}) = P(t_1 | t_0) \, P(t_2 | t_1) \ldots P(t_n | t_{n-1})$$

- this is **Markov** assumption we saw in language modeling
- estimate as in language modeling:

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

- $\mathbf{P(t_1 | t_0)}$ stands for $\mathbf{P(t_1)}$, estimated by $P(t_1) = \dfrac{C(t_1)}{N}$

Markov Model Tagger

- Using these 2 assumptions, find tagging that maximizes

$$\prod_{i=1}^{n} \mathbf{P}(\mathbf{w_i} \mid \mathbf{t_i})\mathbf{P}(\mathbf{t_i} \mid \mathbf{t_{i-1}}) \qquad (1)$$

- Naïve algorithm: given sentence $\mathbf{w_{1,n}}$ go over all possible tag assignments $\mathbf{t_{1,n}}$ and compute (1)
- Choose final tagging $\mathbf{t_{1,n}}$ which maximizes (1)
  - efficiency: for each word try only tags given by the dictionary
  - example: for fly, possible tags are **noun**, **verb** and also **adjective** (meaning keen or artful, mainly in England)

Markov Model Tagger

- Naïve algorithm: given sentence $\mathbf{w}_{1,n}$ go over all possible tag assignments $\mathbf{t}_{1,n}$
- 40 % words have more than 1 tag
- too many tag assignments to try
- if 2 tags per word, then 2^n possible assignments
- exhaustive search is exponential

Markov Model Tagger

- Side note: Markov tagger becomes Charniak's tagger if tags are assumed independent, i.e.

$$P(t_i|t_{i-1})= P(t_i)$$

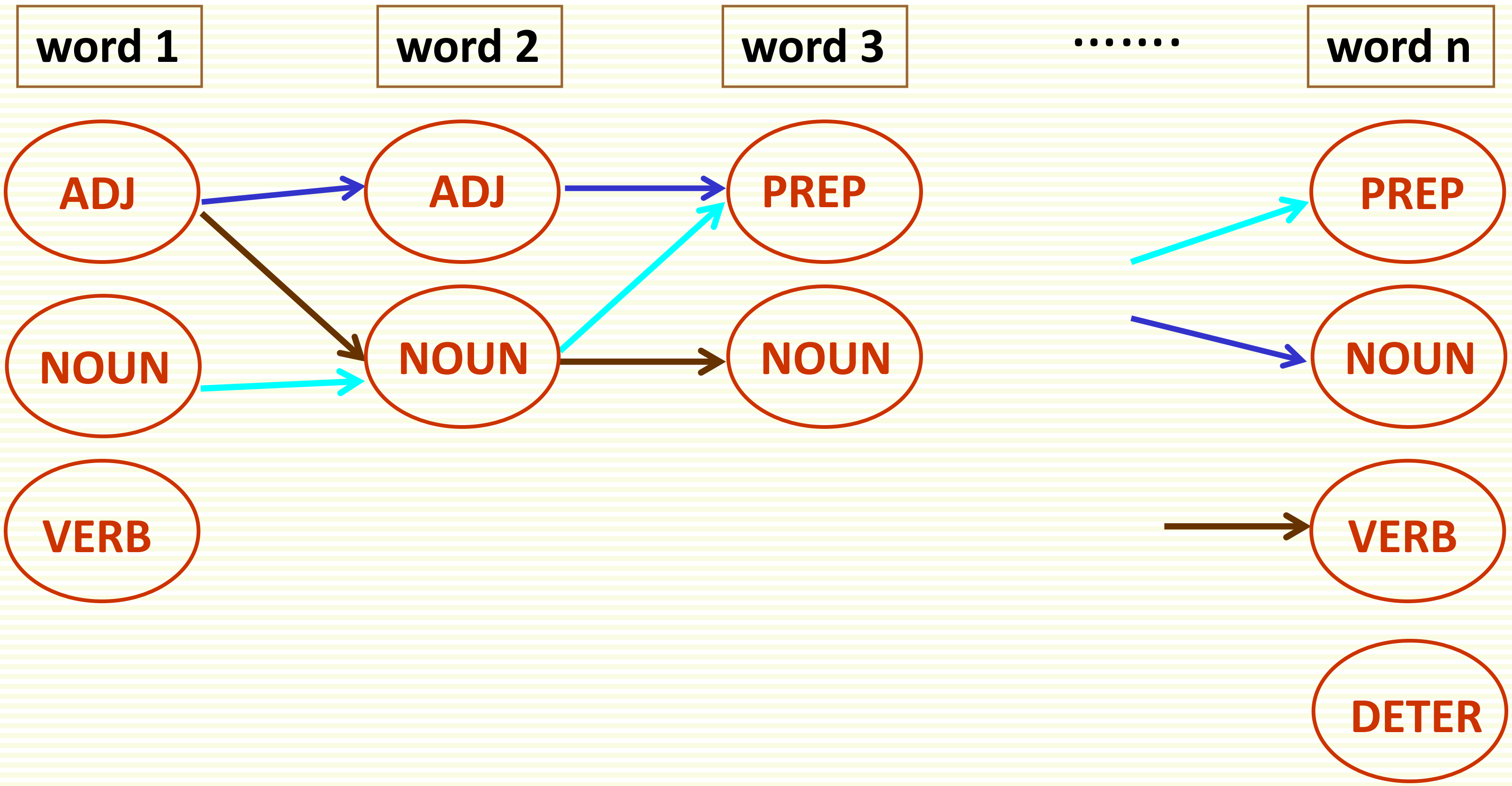
$$\prod_{i=1}^{n} P(w_i \mid t_i)\, P(t_i \mid t_{i-1}) = \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i)$$

$$= \prod_{i=1}^{n} \frac{P(w_i, t_i)}{P(t_i)} P(t_i)$$

$$= \prod_{i=1}^{n} P(w_i, t_i)$$

Markov Model Tagger

$$\prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1})$$



Markov Model Tagger: DP

- Use DP (dynamic programming) to significantly speed up
  - also called Viterbi algorithm
- If **k** tags per word and **n** words, can find best tagging in O($\mathbf{k}^2\mathbf{n}$)
- To avoid floating point underflows, take logarithms

$$\log\left[\prod_{\mathbf{i}=1}^{\mathbf{n}} \mathbf{P}(\mathbf{w}_\mathbf{i} \mid \mathbf{t}_\mathbf{i})\mathbf{P}(\mathbf{t}_\mathbf{i} \mid \mathbf{t}_{\mathbf{i}-1})\right] = \sum_{\mathbf{i}=1}^{\mathbf{n}} \left(\log \mathbf{P}(\mathbf{w}_\mathbf{i} \mid \mathbf{t}_\mathbf{i}) + \log \mathbf{P}(\mathbf{t}_\mathbf{i} \mid \mathbf{t}_{\mathbf{i}-1})\right)$$

how likely word $\mathbf{w}_\mathbf{i}$ is for tag $\mathbf{t}_\mathbf{i}$

how likely tag $\mathbf{t}_\mathbf{i}$ to follow tag $\mathbf{t}_{\mathbf{i}-1}$

Markov Model Tagger: DP

- Turn maximizing:

$$\sum_{i=1}^{n} \log P(w_i \mid t_i) + \sum_{i=1}^{n} \log P(t_i \mid t_{i-1})$$

- Into equivalent minimizing

$$-\sum_{i=1}^{n} \log P(w_i \mid t_i) - \sum_{i=1}^{n} \log P(t_i \mid t_{i-1})$$

Markov Model Tagger: DP

- Find a sequence of tags $\mathbf{t_1, t_2, \ldots, t_n}$ to minimize

$$\sum_{i=1}^{n} -\log P(w_i \mid t_i) + \sum_{i=1}^{n} -\log P(t_i \mid t_{i-1})$$

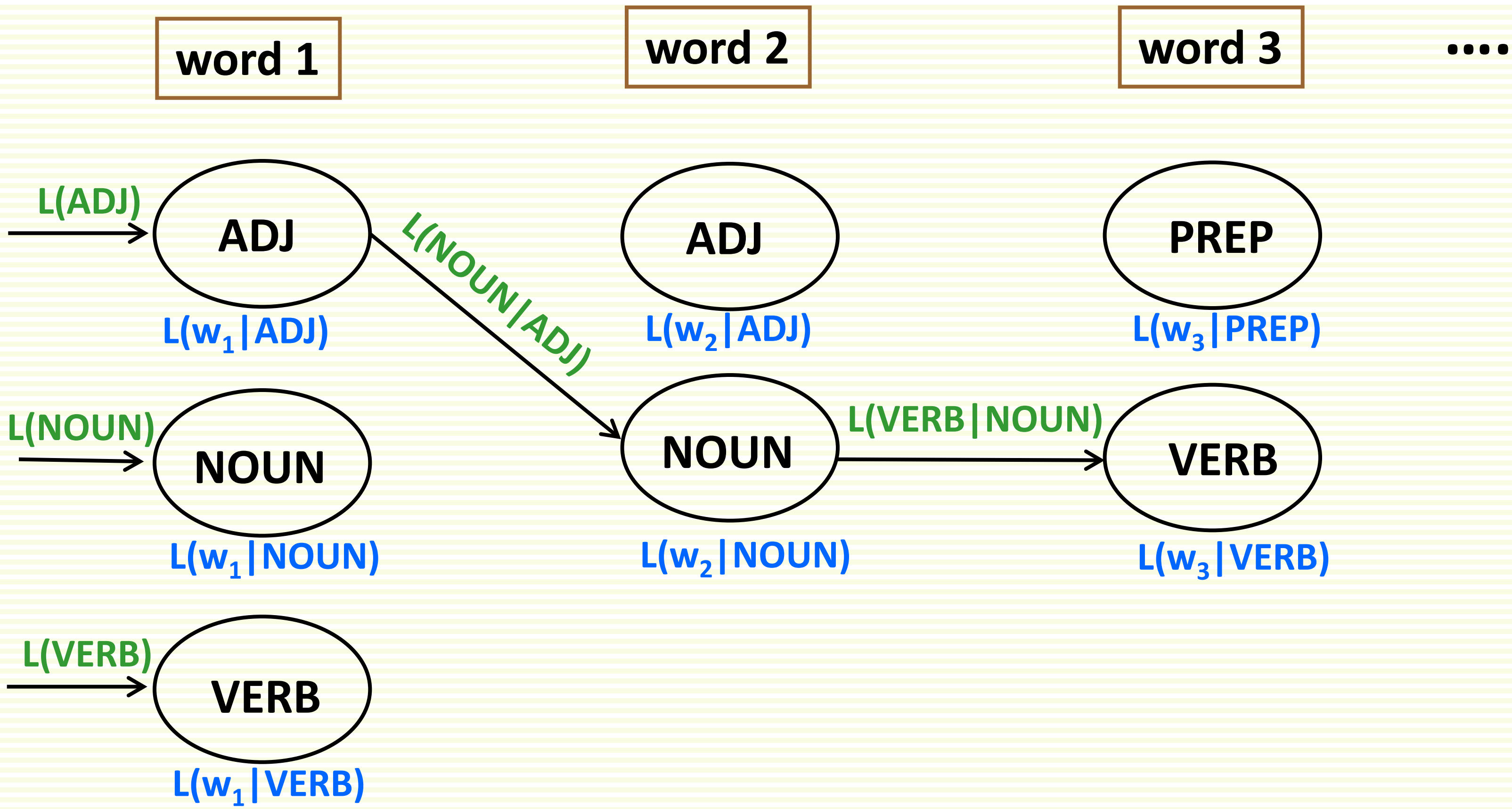
$$\underbrace{\qquad}_{L(w_i \mid t_i)} \qquad \underbrace{\qquad}_{L(t_i \mid t_{i-1})}$$

- In the new notation, find tags $\mathbf{t_1, t_2, \ldots, t_n}$ to minimize:

$$\sum_{i=1}^{n} L(w_i \mid t_i) + \sum_{i=1}^{n} L(t_i \mid t_{i-1})$$

Markov Model Tagger: DP

$$\sum_{i=1}^{n} L(w_i \mid t_i) + \sum_{i=1}^{n} L(t_i \mid t_{i-1})$$

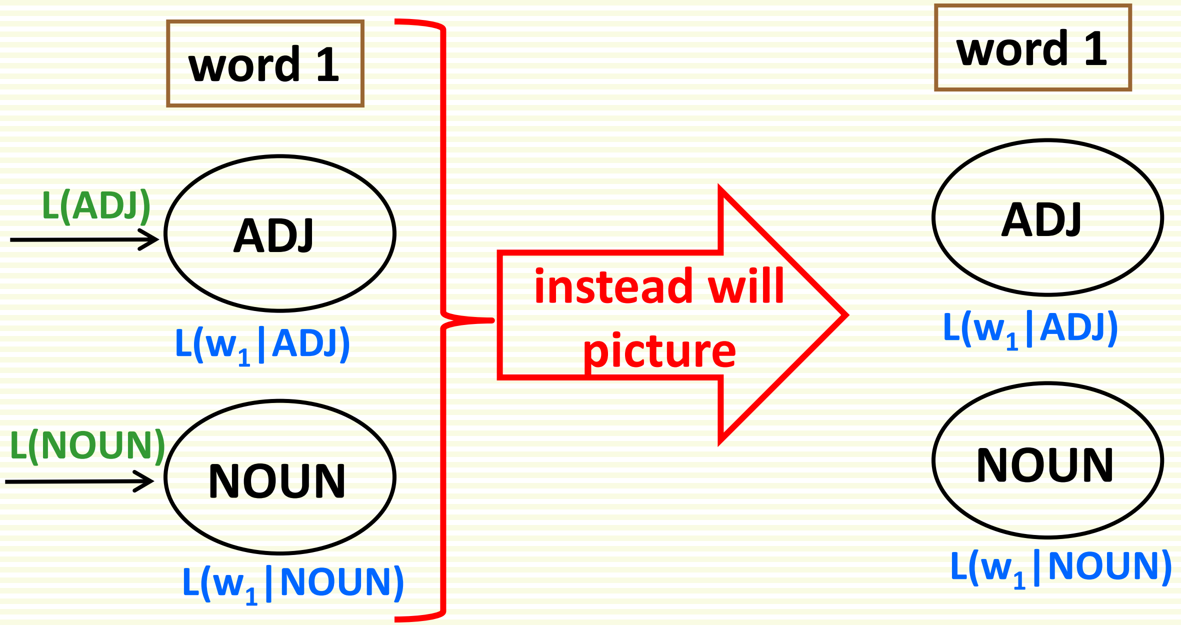


Markov Model Tagger: DP

- Change notation just for the first word:

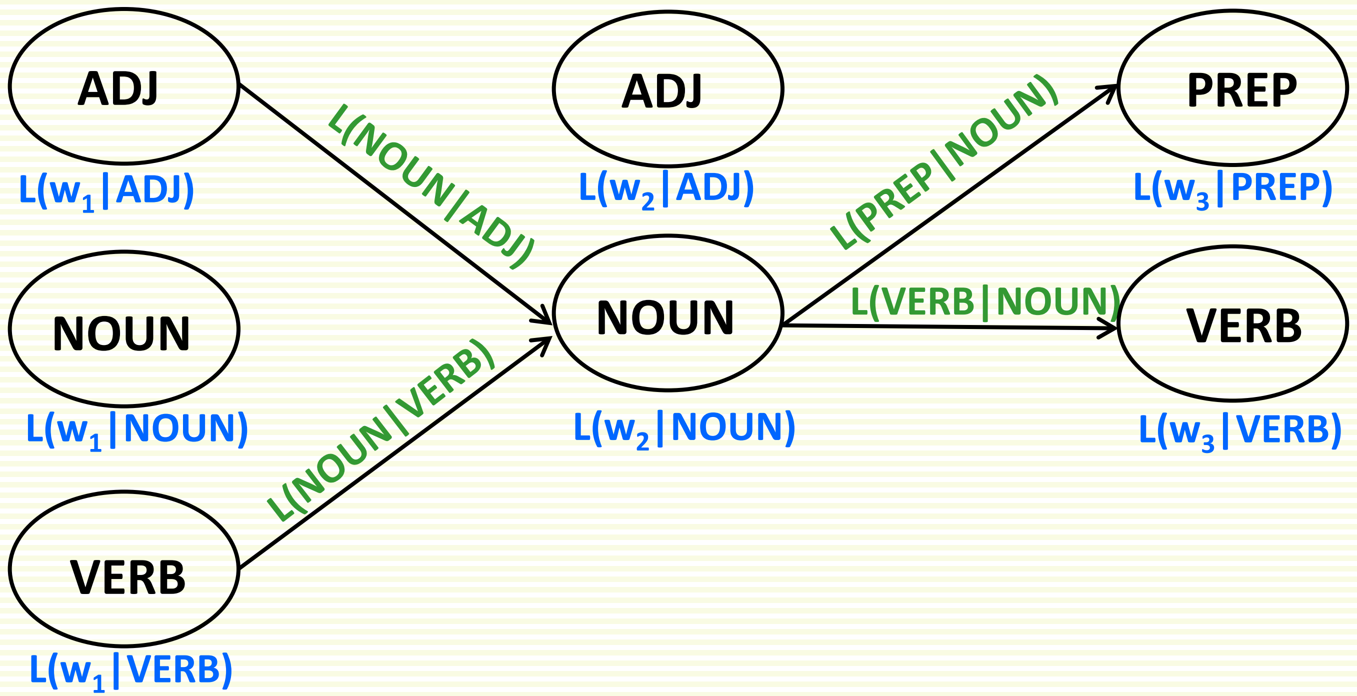
$$\mathbf{L}(\mathbf{w_1}|\mathbf{t_1}) = -\log[\mathbf{P}(\mathbf{w_1}|\mathbf{t_1})] - \log[\mathbf{P}(\mathbf{t_1}|\mathbf{t_0})]$$

$$\sum_{i=1}^{n}\mathbf{L}(\mathbf{w}_i|\mathbf{t}_i) + \sum_{i=1}^{n}\mathbf{L}(\mathbf{t}_i|\mathbf{t}_{i-1}) \Rightarrow \sum_{i=1}^{n}\mathbf{L}(\mathbf{w}_i|\mathbf{t}_i) + \sum_{i=2}^{n}\mathbf{L}(\mathbf{t}_i|\mathbf{t}_{i-1})$$



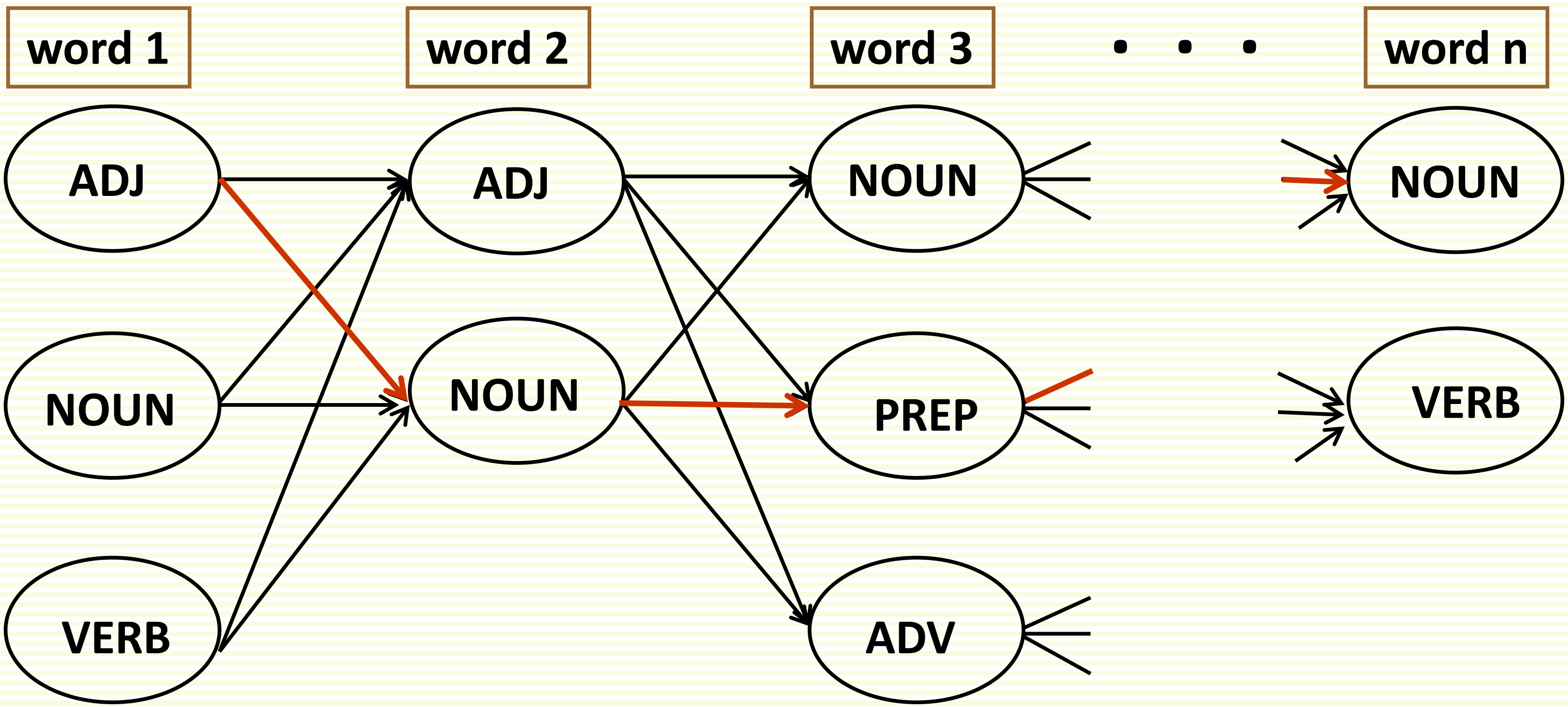
Markov Model Tagger: DP

- Each node has cost $L(w_i | t_i)$
- Each edge has cost $L(t_i | t_{i-1})$
- Cost of a path: $\sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=2}^{n} L(t_i | t_{i-1})$



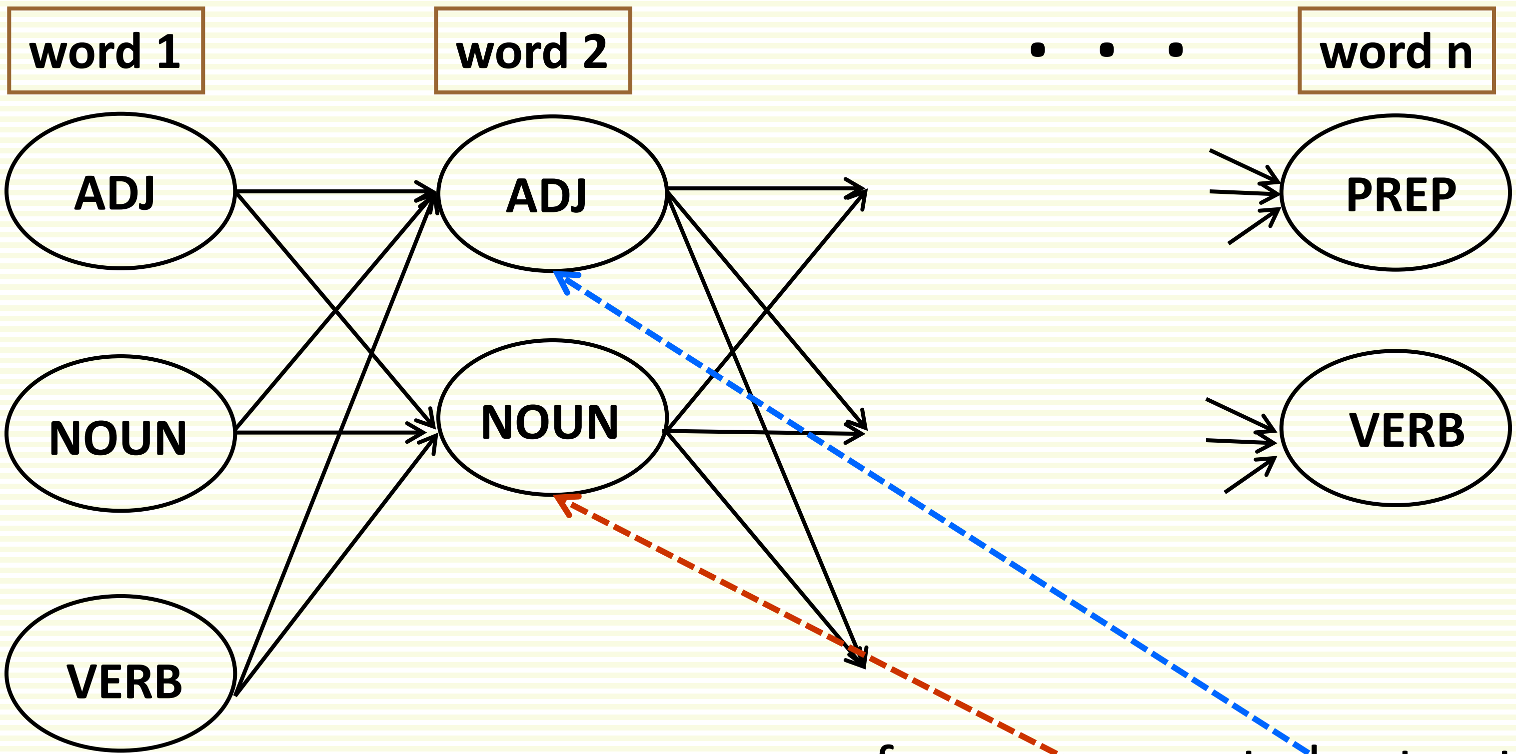
Markov Model Tagger: DP Graph

- Find minimum cost path that starts at some node corresponding to **word 1** and ends at some node corresponding to **word n**



Markov Model Tagger: Main Step of DP

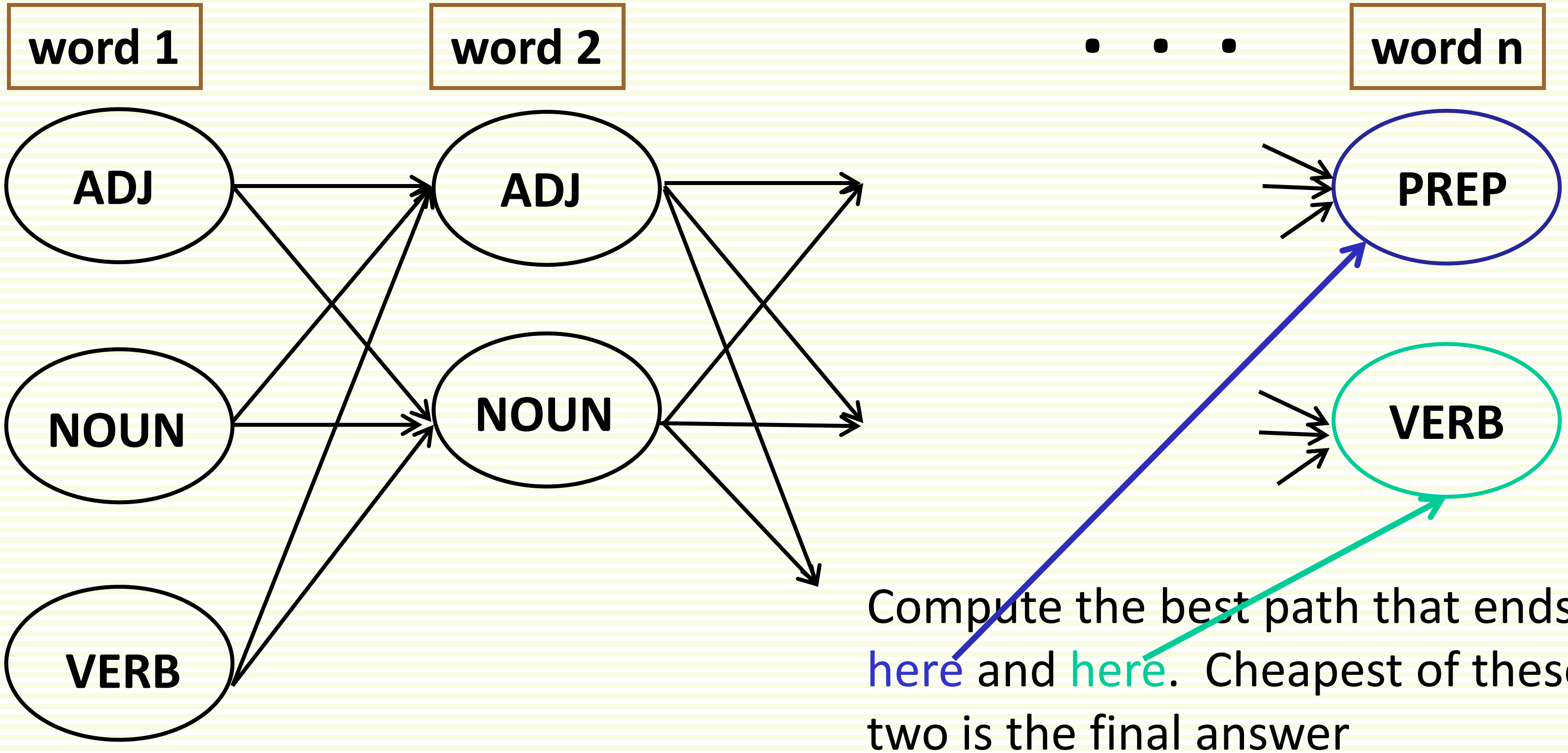
- Main Step: for every node at word $\mathbf{w_i}$, find smallest cost path that leads into it, starting at any node at word $\mathbf{w_1}$



for $\mathbf{w_2}$, compute best path that ends here and here

Markov Model Tagger: DP Overview

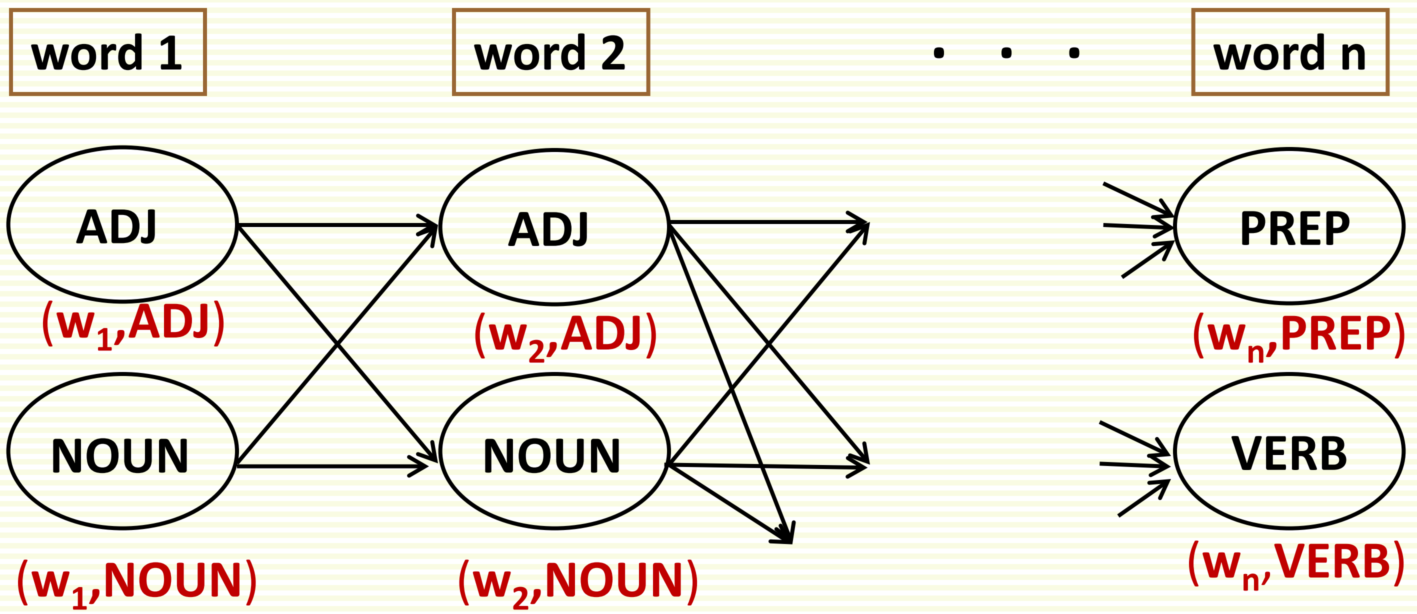
- First compute the best path that ends at any node for $\mathbf{w_1}$
- Then compute the best path that ends at any node for $\mathbf{w_2}$
- …..
- Finally compute the best path that ends at any node for $\mathbf{w_n}$
- The best path overall is smallest cost path that end at $\mathbf{w_n}$



Compute the best path that ends here and here. Cheapest of these two is the final answer

Markov Model Tagger: DP Variables

- For word $\mathbf{w_i}$ tag $\mathbf{t}$ node is $(\mathbf{w_i}, \mathbf{t})$

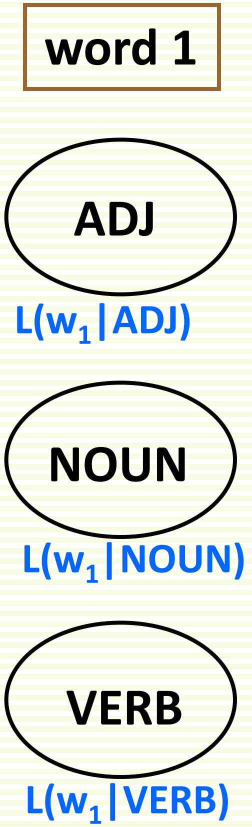


- $\mathbf{C}(\mathbf{w_i}, \mathbf{t})$ cost of best path that starts at any $(\mathbf{w_1}, \mathbf{t})$ and ends at $(\mathbf{w_i}, \mathbf{t})$
- $\mathbf{P}(\mathbf{w_i}, \mathbf{t})$ is parent of node $(\mathbf{w_i}, \mathbf{t})$ on this path
- After all $\mathbf{C}(\mathbf{w_i}, \mathbf{t})$ computed, min of $\mathbf{C}(\mathbf{w_n}, \mathbf{t})$ over all $\mathbf{t}$ gives best path

Markov Model Tagger: DP Initialization

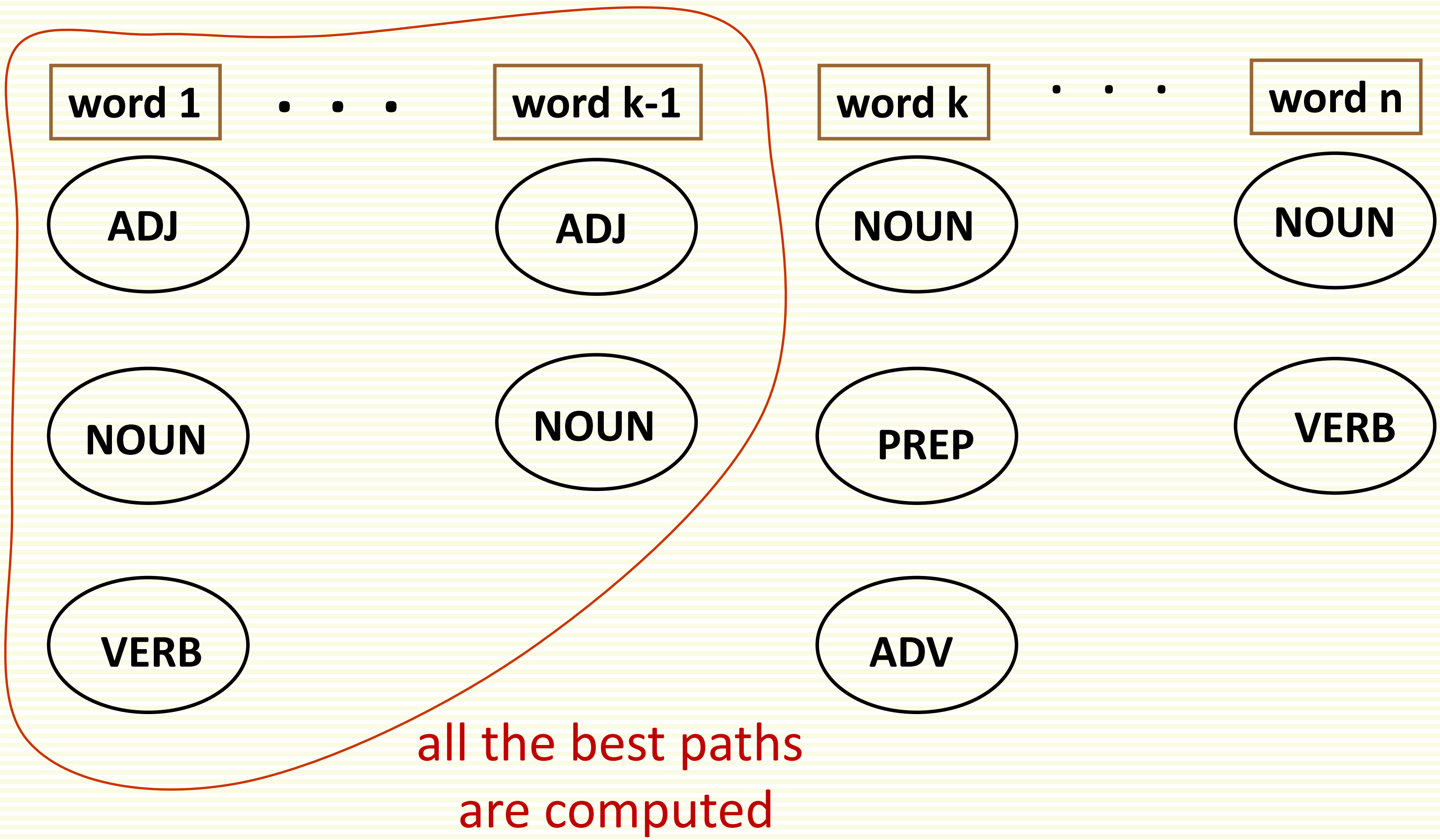
- First compute the best path that ends at any node for $\mathbf{w_1}$
  - trivial, since the path has just one node
- For all tags of the first word **t** :

$$\mathbf{C}(\mathbf{w_1},\mathbf{t}) = \mathbf{L}(\mathbf{w_1}|\mathbf{t}) \qquad \mathbf{P}(\mathbf{w_1},\mathbf{t}) = \text{null}$$



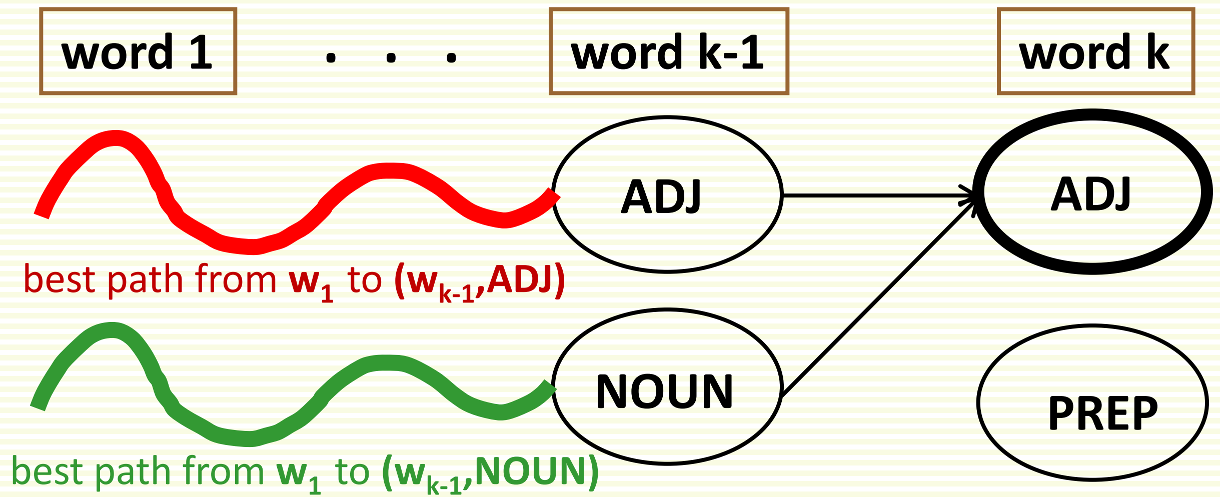
Markov Model Tagger: DP Iteration

- Computed **C(w_i,t)** and **P(w_i,t)** for all tags **t** and **i** < **k**



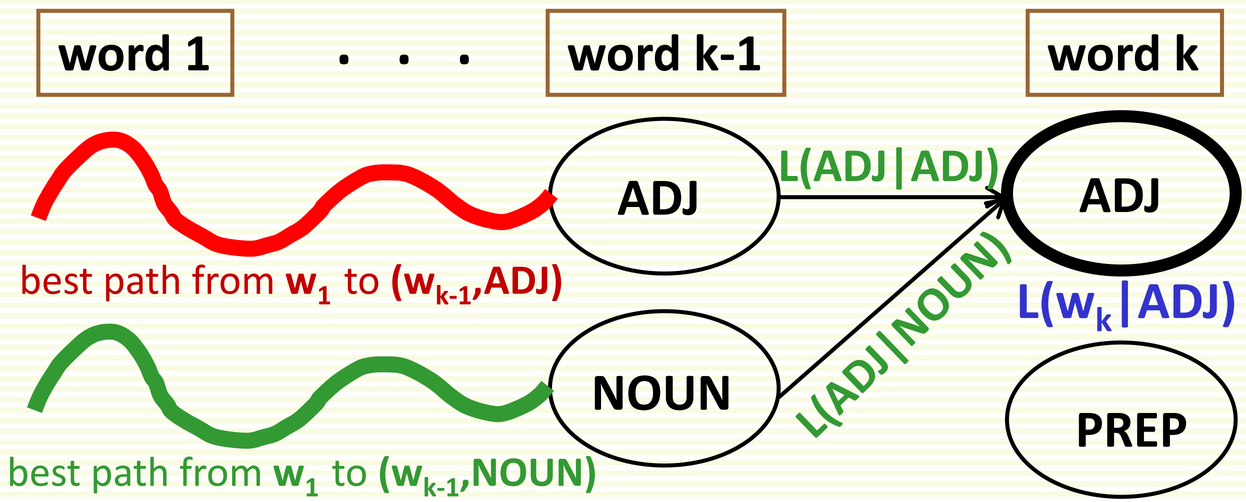
Markov Model Tagger: DP Iteration

- Now compute $\mathbf{C}(\mathbf{w}_k,\mathbf{t})$ and $\mathbf{P}(\mathbf{w}_k,\mathbf{t})$ for **k**
- Consider node ($\mathbf{w}_k$, **ADJ**)



- The best path from $\mathbf{w}_1$ to ($\mathbf{w}_k$, **ADJ**) goes through either
  1. ($\mathbf{w}_{k-1}$,**ADJ**): then it follows best path from $\mathbf{w}_1$ to ($\mathbf{w}_{k-1}$, **ADJ**)
  2. ($\mathbf{w}_{k-1}$,**NOUN**): then it follows best path from $\mathbf{w}_1$ to ($\mathbf{w}_{k-1}$, **NOUN**)
  - because a sub-path of the best path is a best path itself

Markov Model Tagger: DP Iteration



- **C(w_k, ADJ)** is the smaller of two quantities:

1. **C(w_{k-1},ADJ)** + **L(ADJ|ADJ)** + **L(w_k|ADJ)**
   - then **P(w_k, ADJ) = (w_{k-1}, ADJ)**
2. **C(w_{k-1},NOUN)** + **L(ADJ|NOUN)** + **L(w_k|ADJ)**
   - then **P(w_k, ADJ) = (w_{k-1}, NOUN)**

Markov Model Tagger: DP Iteration

- In general, $\mathbf{C}(\mathbf{w_k}, \mathbf{t})$ is computed as follows:

cost of best path from first word to node (word k-1, t')

cost of going through node (w_k, t)

$$\mathbf{C}(\mathbf{w}_\mathbf{k}, \mathbf{t}) = \min_{\mathbf{t}' \in \mathbf{T}(\mathbf{w}_{\mathbf{k}-1})} \{\mathbf{C}(\mathbf{w}_{\mathbf{k}-1}, \mathbf{t}') + \mathbf{L}(\mathbf{t} \mid \mathbf{t}')\} + \mathbf{L}(\mathbf{w}_\mathbf{k} \mid \mathbf{t})$$

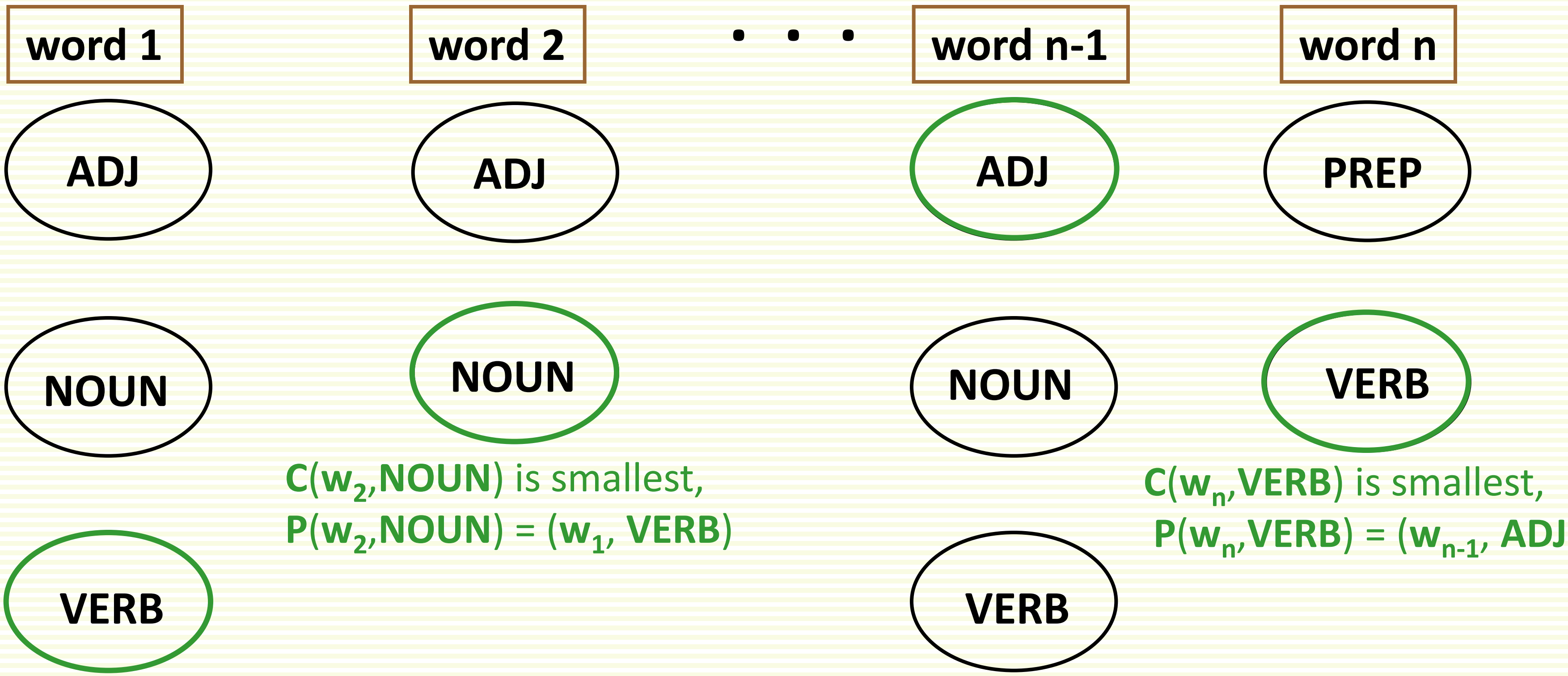
search over all tags t' for word k-1

cost of going between nodes (w_{k-1}, t') and (w_k,t)

- $\mathbf{P}(\mathbf{w_k}, \mathbf{t}) = (\mathbf{w_{k-1}}, \mathbf{t^*})$ where $\mathbf{t^*}$ is the tag for word $\mathbf{w_{k-1}}$ minimizing the expression above

Markov Model Tagger: DP Termination

- After computed all **C(w_i, t)** best cost path is found as the minimum of **C(w_n,t)** over all tags **t**
- Parents on the path traced back using **P(w_i,t)**



- Final tagging is: VERB NOUN … ADJ VERB

MMT Example

L(book|ADJ) = 10

L(book |VERB) = 1

L(book|NOUN) =2

L(that|PRON) = 2

L(that|CONJ) = 4

L(flight|NOUN) = 2

L(flight|VERB) = 1

book

ADJ

VERB

NOUN

L(PRON|VERB) = 3

L(CONJ|VERB)=4

L(PRON|NOUN) =2

L(CONJ|NOUN)= 1

L(PRON|ADJ) =1

L(CONJ|ADJ) =2

that

PRON

CONJ

L(NOUN|PRON) =1

L(VERB|PRON) =10

L(NOUN|CONJ) =4

L(VERB|CONJ) =2

flight

NOUN

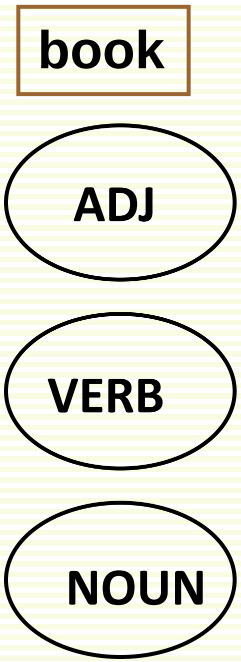
VERB

MMT Example

L(book|ADJ) = 10

L(book|VERB) = 1

L(book|NOUN) = 2



- Iteration 1:
  - **C**(**book**,**ADJ**) = 10, **P**(**book**,**ADJ**) = null
  - **C**(**book**,**VERB**) = 1, **P**(**book**,**VERB**) = null
  - **C**(**book**,**NOUN**) = 2, **P**(**book**,**NOUN**) = null

MMT Example

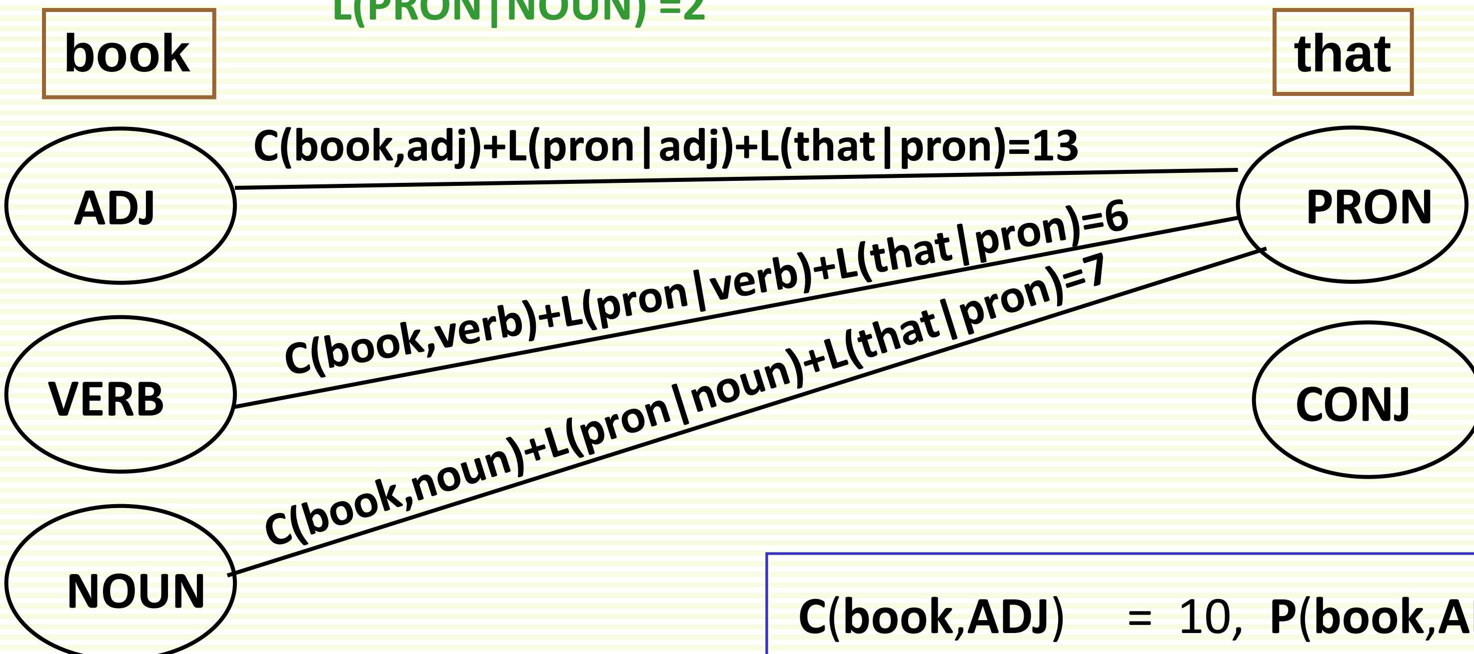
L(PRON|ADJ) =1

L(PRON|VERB) = 3

L(PRON|NOUN) =2

L(that|PRON) = 2

L(that|CONJ) = 4



C(**book**,**ADJ**) = 10, **P**(**book**,**ADJ**) = null
C(**book**,**VERB**) = 1, **P**(**book**,**VERB**) = null
C(**book**,**NOUN**) = 2, **P**(**book**,**NOUN**) = null

- Iteration 2:
  - **C**(**that**, **PRON**) = 6, **P**(**that**,**PRON**) = (**book**,**VERB**)

MMT Example

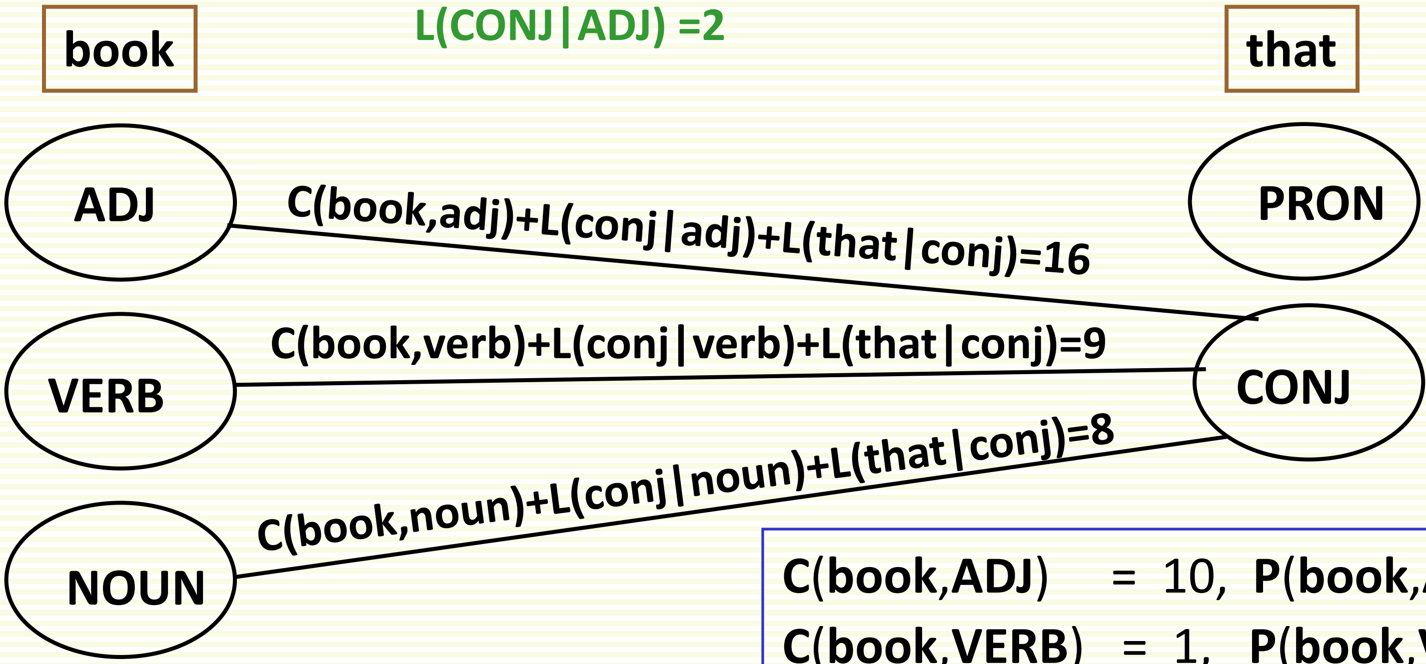
L(CONJ|VERB)=4

L(CONJ|NOUN)= 1

L(CONJ|ADJ) =2

L(that|PRON) = 2

L(that |CONJ) = 4



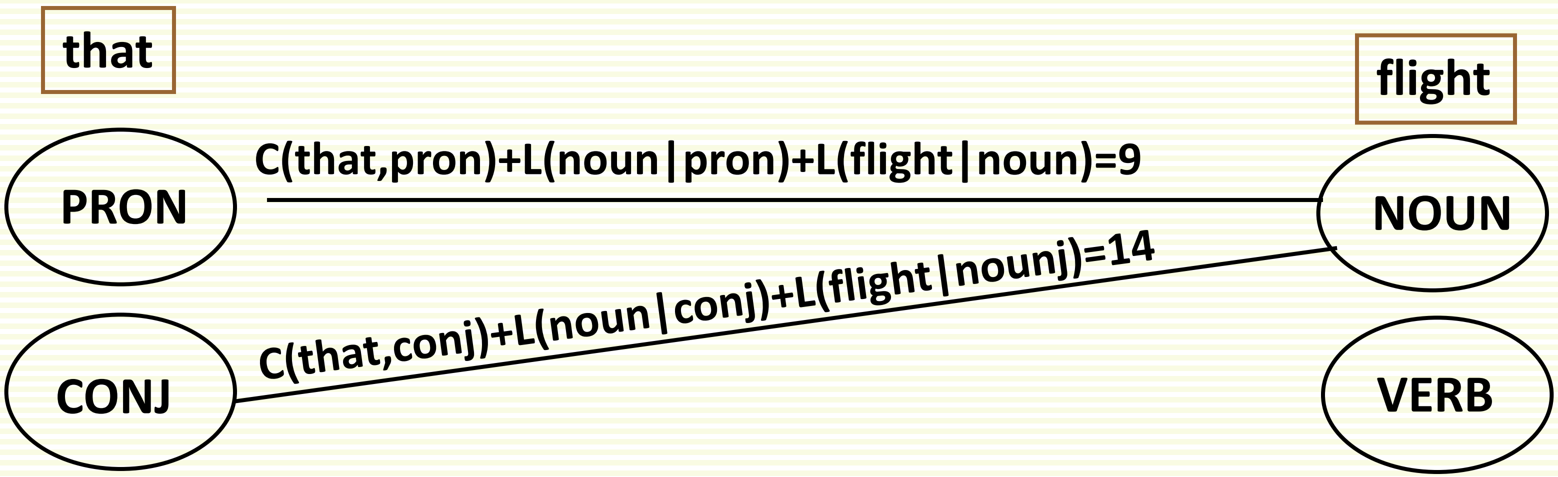
C(**book,ADJ**) = 10, **P**(**book,ADJ**) = null
C(**book,VERB**) = 1, **P**(**book,VERB**) = null
C(**book,NOUN**) = 2, **P**(**book,NOUN**) = null

- Iteration 2:
  - **C**(**that, CONJ**) = 8, **P**(**that, CONJ**) = (**book,NOUN**)

MMT Example

L(NOUN|PRON) =1
L(NOUN|CONJ) =4

L(flight|NOUN) = 2
L(flight|VERB) = 1



C(book,ADJ) = 10, P(book,ADJ) = null
C(book,VERB) = 1, P(book,VERB) = null
C(book,NOUN) = 2, P(book,NOUN) = null
C(that, PRON) = 6, P(that, PRON) = (book,VERB)
C(that, CONJ) = 8, P(that, CONJ) = (book,NOUN)

- Iteration 3:
  - **C(flight, NOUN)** = 9, **P(flight, NOUN) = (that,PRON)**

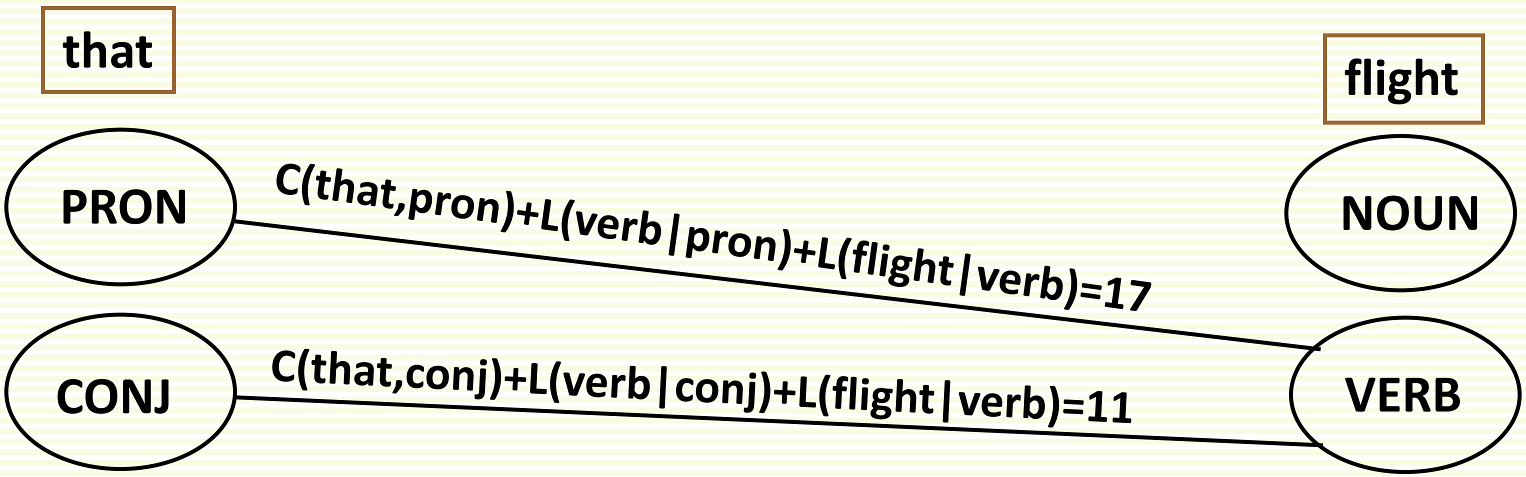
MMT Example

L(VERB|PRON) =10

L(VERB|CONJ) =2

L(flight|NOUN) = 2

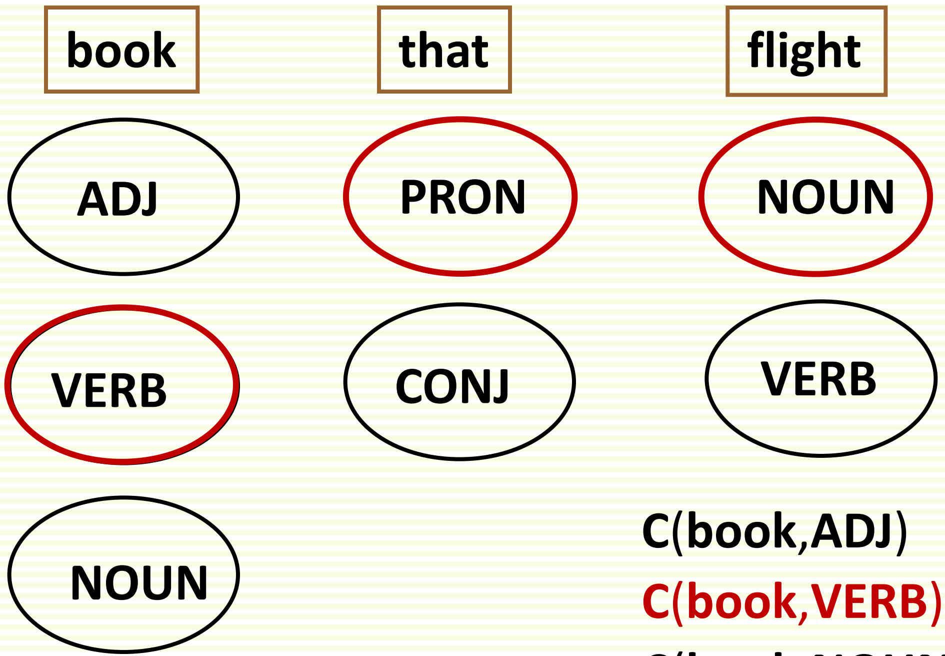
L(flight|VERB) = 1



C(book,ADJ) = 10,	P(book,ADJ) = null
C(book,VERB) = 1,	P(book,VERB) = null
C(book,NOUN) = 2,	P(book,NOUN) = null
C(that, PRON) = 6,	P(that, PRON) = (book,VERB)
C(that, CONJ) = 8,	P(that, CONJ) = (book,NOUN)

- Iteration 3:
  - C(flight, VERB) = 11, P(flight, VERB) = (that,CONJ)

MMT Example



C(book,ADJ) = 10, P(book,ADJ) = null
C(book,VERB) = 1, P(book,VERB) = null
C(book,NOUN) = 2, P(book,NOUN) = null

C(that, PRON) = 6, P(that, PRON) = (book,VERB)
C(that, CONJ) = 8, P(that, CONJ) = (book,NOUN)

C(flight, NOUN) = 9, P(flight, NOUN) = (that,PRON)
C(flight, VERB) = 11, P(flight,VERB) = (that,PRON)

Final Tagging: Book<verb> that <pron> flight<noun>

MMT: Pseudo Code for DP

- Tags($\mathbf{w_i}$) is the set of all possible tags for $\mathbf{w_i}$

for each t $\in$Tags($\mathbf{w_1}$) **do**

  C(w_1, t) = **L**($\mathbf{w_1}$| **t**), **P**($\mathbf{w_1}$, **t**) = null

for i $\leftarrow$ 2 t**o** *n* **do**

  for each t $\in$ Tag($\mathbf{w_i}$) **do**

    C($\mathbf{w_i}$, **t**) = -$\propto$

    for each t' $\in$ Tag($\mathbf{w_{i-1}}$) **do**

      nextCost = **C**($\mathbf{w_{i-1}}$,**t'**) + **L**(**t**|**t'**) + **L**($\mathbf{w_i}$|**t**)

      if nextCost < cost($\mathbf{w_i}$, **t**) **do**

        C($\mathbf{w_i}$,**t**) = nextCost

        P($\mathbf{w_i}$,**t**) = **t'**

Unknown Words

- Simplest method: assume an unknown word could belong to any tag; unknown words are assigned the distribution over POS over the whole lexicon
  - P("karumbula"|verb) = P("karumbula"|noun) = P("karumbula"|adjective) = …. etc
- Some tags are more common than others
  - for example a new word can be most likely a verb, a noun etc. but not a preposition or an article
- Use morphological and other cues
  - for example words ending in *–ed* are likely to be past tense forms or past participles

Tagging Accuracy

- Ranges from 96%-97%
- Depends on:
  - Amount of training data available
  - The tag set
  - Difference between training corpus and dictionary and the corpus of application
  - Unknown words in the corpus of application
- A change in any of these factors can have a dramatic effect on tagging accuracy – often much more stronger than the choice of tagging method