CS447: Natural Language Processing

http://courses.engr.illinois.edu/cs447

Lecture 6: Part-of-speech tagging

Julia Hockenmaier

juliahmr@illinois.edu

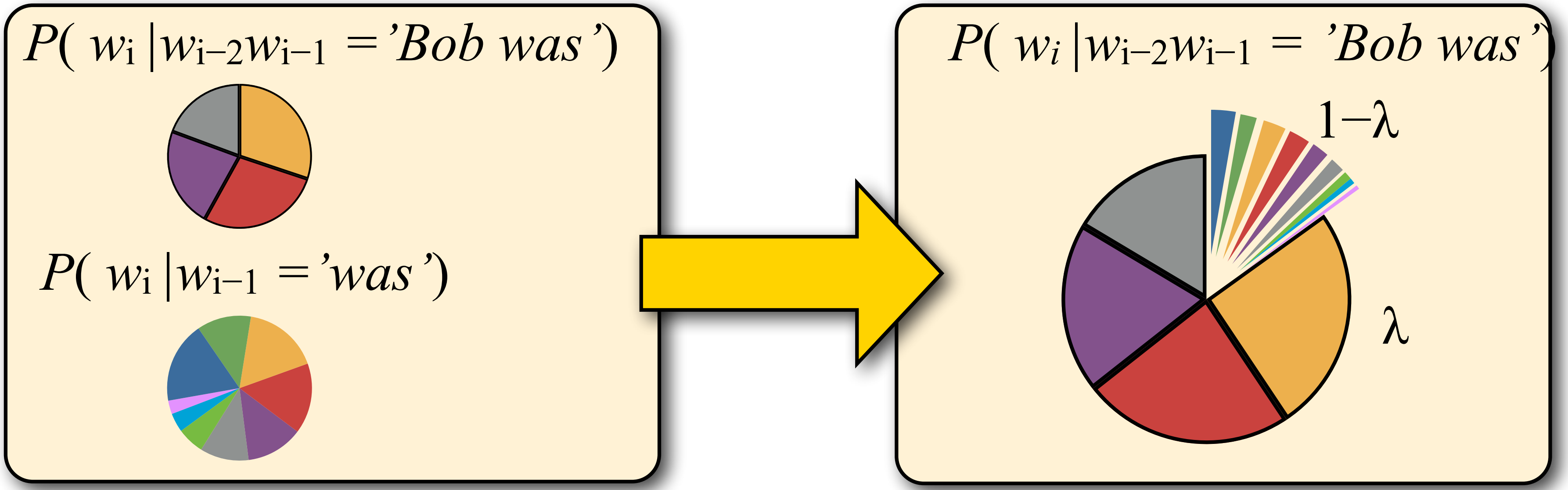
3324 Siebel Center

Smoothing: Reserving mass in $P(X \mid Y)$ for unseen events

Linear Interpolation (Recap)

We don’t see “*Bob was reading*”, but we see “__ *was reading*”.
We estimate $P(reading\,|\,\text{'Bob was'}) = 0$ but $P(reading\,|\,\text{'was'}) > 0$

Use (*n* –1)-gram probabilities to smooth *n*-gram probabilities:

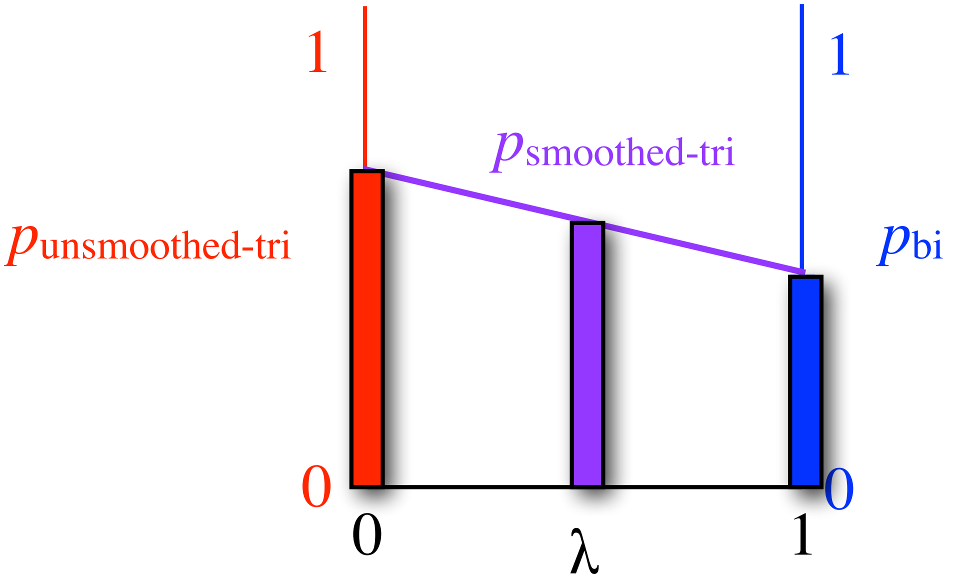

$P(w_i \,|\, w_{i-2}w_{i-1} = \text{'Bob was'})$
$P(w_i \,|\, w_{i-1} = \text{'was'})$
$P(w_i \,|\, w_{i-2}w_{i-1} = \text{'Bob was'})$
$1-\lambda$
λ

$$\underbrace{\tilde{P}_{LI}(w_i|w_{i-n}w_{i-n+1}\ldots w_{i-2}w_{i-1})}_{\text{smoothed n-gram}} = \lambda\,\underbrace{\hat{P}(w_i|w_{i-n}w_{i-n+1}\ldots w_{i-2}w_{i-1})}_{\text{unsmoothed n-gram}}$$
$$+(1-\lambda)\,\underbrace{\tilde{P}_{LI}(w_i|w_{i-n+1}\ldots w_{i-2}w_{i-1})}_{\text{smoothed (n-1)-gram}}$$

What happens to $P(\mathrm{w} \mid \ldots)$?

The smoothed probability $P_{\text{smoothed-tri}}(w_i \mid w_{i-2}\, w_{i-1})$ is a linear combination of $P_{\text{unsmoothed-tri}}(w_i \mid w_{i-2}\, w_{i-1})$ and $P_{\text{bi}}(w_i \mid w_{i-1})$:

$$P_{\text{smoothed-tri}}(w_i \mid w_{i-2}\, w_{i-1}) = (1-\lambda)\, P_{\text{unsmoothed-tri}}(w_i \mid w_{i-2}\, w_{i-1}) + \lambda\, P_{\text{bi}}(w_i \mid w_{i-1})$$



Absolute discounting

Subtract a constant factor $D < 1$ from each nonzero *n*-gram count, and interpolate with $P_{AD}(w_i \mid w_{i-1})$:

non-zero if trigram $w_{i-2}w_{i-1}w_i$ is seen

$$P_{AD}(w_i|w_{i-1}, w_{i-2}) = \frac{\max(C(w_{i-2}w_{i-1}w_i) - D, 0)}{C(w_{i-2}w_{i-1})} + (1-\lambda)P_{AD}(w_i|w_{i-1})$$

If S seen word types occur after $w_{i-2}\ w_{i-1}$ in the training data, this reserves the probability mass $P(U) = (S \times D)/C(w_{i-2}w_{i-1})$ to be computed according to $P(w_i \mid w_{i-1})$. Set:

$$(1-\lambda) = P(U) = \frac{S \cdot D}{C(w_{i-2}w_{i-1})}$$

N.B.: with N_1, N_2 the number of *n*-grams that occur once or twice, $D = N_1/(N_1+2N_2)$ works well in practice

Kneser-Ney smoothing

Observation: *"San Francisco"* is frequent,
but *"Francisco"* only occurs after "*San".*

Solution: the unigram probability $P(w)$ should not depend on the frequency of w, but on the number of contexts in which w appears

$N_{+1}(\bullet w)$: number of contexts in which w appears
= number of word types w' which precede w
$N_{+1}(\bullet\bullet) = \sum_{w'} N_{+1}(\bullet w')$

Kneser-Ney smoothing: Use absolute discounting,
but use $P(w) = N_{+1}(\bullet w)/N_{+1}(\bullet\bullet)$

Modified Kneser-Ney smoothing: Use different *D for bigrams and trigrams (Chen & Goodman '98)*

Class Admin

Homework assignments

Schedule:

Week 1: Friday, 09/25 HW0 out (today!)
Week 3: Friday, 09/15 HW0 due, HW1 out
Week 6: Friday, 10/06 HW1 due, HW2 out
Week 9: Friday, 10/27 HW2 due, HW3 out
Week 12: Friday, 11/17 HW3 due, HW4 out
Week 15: Wednesday, 12/13 HW4 due (last lecture)

Points per assignment:

HW0 = 2 points
(Did you submit (on time)? Was it in the right format?)
HW1,HW2,HW3,HW4 = 10 points per assignment

Homework assignments

HW0 is due **at 10pm** today.

HW1 will go out this evening.
HW1 is due **at 10pm** on Friday, Oct 6.
We use Compass.

Email us ASAP if you cannot access the Compass page for our class.

4th credit hour

Two choices:

- **Research project** (alone, or with one other student)
- **Literature survey** (alone)

Deadlines:

- **Before Oct 20: Check your idea with me**
- **Oct 20 (Wk 8): Proposal due**
  (What topic? What papers will you read?)
- **Nov 15 (Wk 12): Progress report due**
  (Are your experiments on track? Is your paper on track?)
- **Dec 14 (Reading Day): Final report due**
  (Summary of papers, your system)

4th credit hour: Research Projects

What?

You need to read and describe a few (2-3) NLP papers on a particular task, implement an NLP system for this task and describe it in a written report.

Why?

To make sure you get a deeper knowledge of NLP by reading original papers and by building an actual system.

When?

Oct 20 (Wk 8): Proposal due (What topic? What papers will you read?)
Nov 15 (Wk 12): Progress report due (Are your experiments on track?)
Dec 14 (Reading Day): Final report due (Summary of papers, your system)

4th credit hour: Literature Survey

What?

You need to read and describe several (5-7) NLP papers on a particular task or topic, and produce a written report that compares and critiques these approaches.

Why?

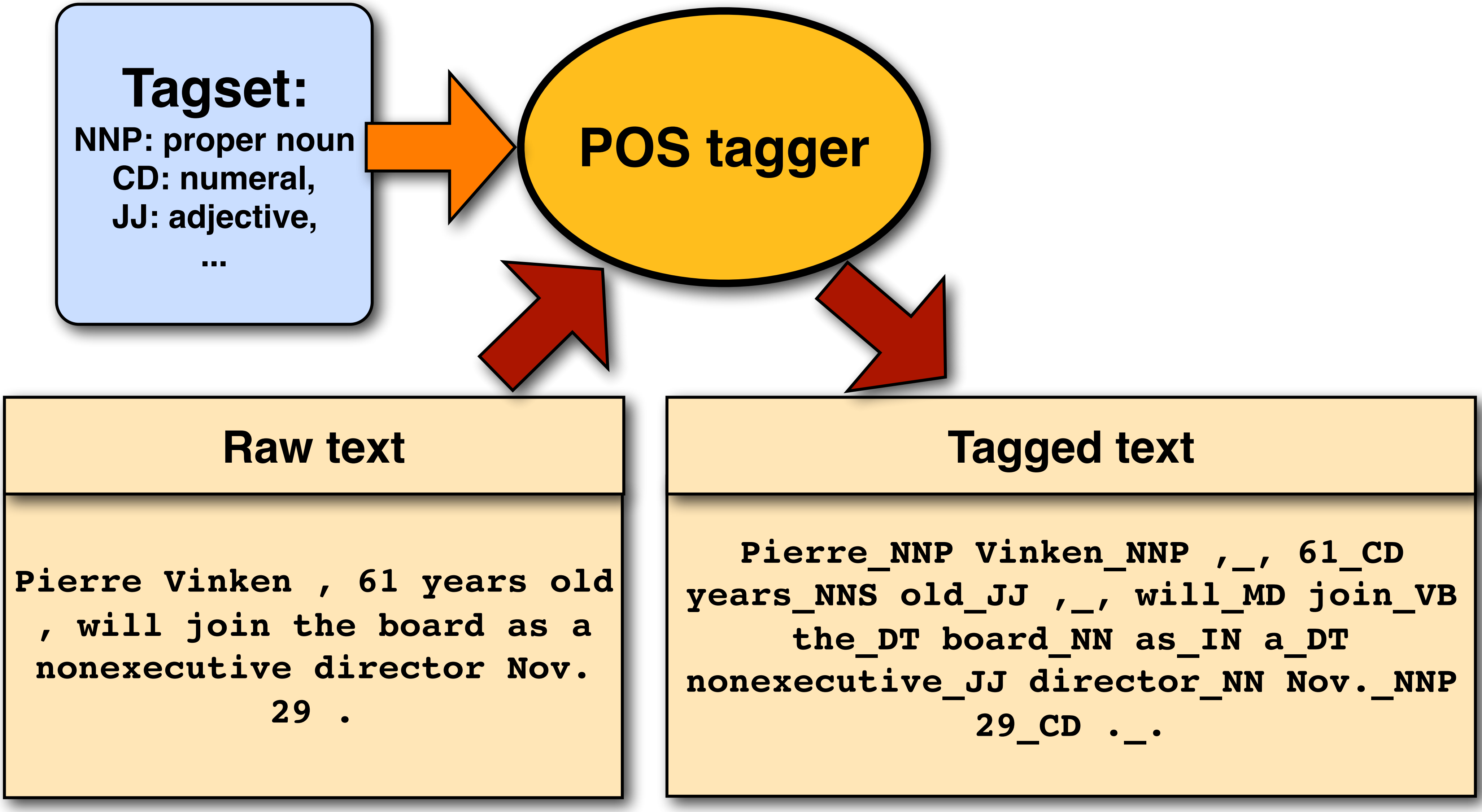
To make sure you get a deeper knowledge of NLP by reading original papers, even if you don't build an actual system.

When?

Oct 20 (Wk 8): Proposal due (What topic? What papers will you read?)
Nov 15 (Wk 12): Progress report due (Is your paper on track?)
Dec 14 (Reading Day): Final report due (Summary of papers)

Part-of-speech (POS) tagging

POS tagging



Tagset:

NNP: proper noun
CD: numeral,
JJ: adjective,
...

POS tagger

Raw text

`Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .`

Tagged text

`Pierre_NNP Vinken_NNP ,_, 61_CD years_NNS old_JJ ,_, will_MD join_VB the_DT board_NN as_IN a_DT nonexecutive_JJ director_NN Nov._NNP 29_CD ._.`

What is POS tagging?

POS = part-of-speech:

- **Broad word classes:**
  Noun, Verb, Adjective, Adverb, Preposition, …
- **More fine-grained distinctions, e.g.:**
  Common noun, proper noun, pronoun, infinitive, past participle, …

POS tag:

A label for a particular part-of-speech

(e.g. NN = common noun (singular), VB = infinitive verb, …)

POS tagging:

Determine for each word in a sentence which part-of-speech tag it has in that context.

POS Tagging: Ambiguity

Words often have more than one POS:

The **back** door	(adjective)
On my **back**	(noun)
Win the voters **back**	(particle)
Promised to **back** the bill	(verb)

The POS tagging task is to determine the POS tag for a particular instance of a word.

Since there is ambiguity, we cannot simply look up the correct POS in a dictionary.

These examples from Dekang Lin

Why POS tagging?

POS tagging is a prerequisite for further analysis:

- Parsing:

  POS tags give a good indication of possible grammatical analyses.

- Information extraction:

  Finding names, relations, etc.

- Machine Translation:

  The noun “content” may have a different translation from the adjective

- Speech synthesis:
  - How to pronounce “lead”?
  - INsult or inSULT, OBject or obJECT, OVERflow or overFLOW, DIScount or disCOUNT, CONtent or conTENT

Defining a tagset

Word classes

Open classes:

Nouns, Verbs, Adjectives, Adverbs

Closed classes:

Auxiliaries and modal verbs

Prepositions, Conjunctions

Pronouns, Determiners

Particles, Numerals

(see Appendix for details)

Defining a tagset

Tagsets have different granularities:

- Brown corpus (Francis and Kucera 1982): 87 tags
- Penn Treebank (Marcus et al. 1993): 45 tags

Simplified version of Brown tag set; de facto standard for English now:

NN: common noun (singular or mass): water, book
NNS: common noun (plural): books

- Prague Dependency Treebank (Czech): 4452 tags

Complete morphological analysis:

AAFP3----3N----: (nejnezajímavějším) [Hajic 2006, VMC tutorial]
Adjective Regular Feminine Plural Dative….Superlative

Tagsets for English

We have to agree on a standard inventory of word classes.

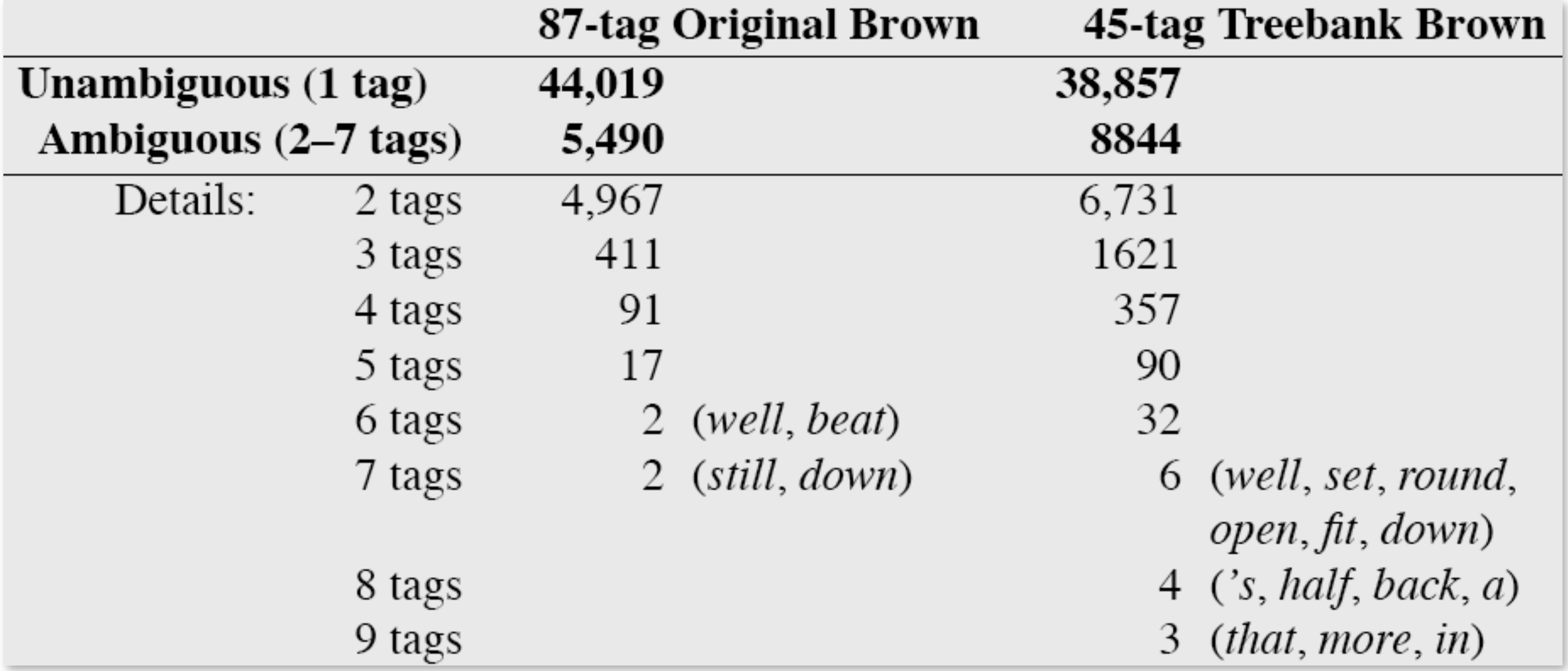
Most taggers rely on statistical models; therefore the tagsets used in large corpora become de facto standard.

Tagsets need to capture semantically or syntactically important distinctions that can easily be made by trained human annotators.

How much ambiguity is there?

How many tags does each word type have?

(Original Brown corpus: 40% of tokens are ambiguous)



		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well*, *beat*)	32	
	7 tags	2	(*still*, *down*)	6	(*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags			4	(*'s*, *half*, *back*, *a*)
	9 tags			3	(*that*, *more*, *in*)

NB: These are just the tags that words appeared with in the corpus.
There are always unseen word/tag combinations.

Evaluating POS taggers

Evaluation metric: accuracy

How many words in the unseen test data can you tag correctly?

State of the art on Penn Treebank: around 97%.

Compare your model against a baseline

Standard: assign to each word its most likely tag (use training corpus to estimate P(t|w))

Baseline performance on Penn Treebank: around 93.7%

… and a (human) ceiling

How often do human annotators agree on the same tag?

Penn Treebank: around 97%

Is POS-tagging a solved task?

Penn Treebank POS-tagging accuracy
≈ human ceiling

Yes, but:

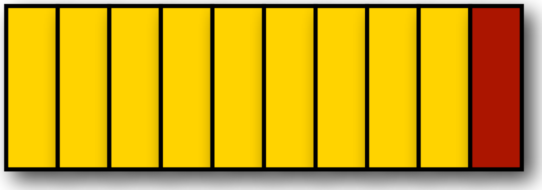
Other languages with more complex morphology need much larger tagsets for tagging to be useful, and will contain many more distinct word forms in corpora of the same size

They often have much lower accuracies

Evaluating POS taggers

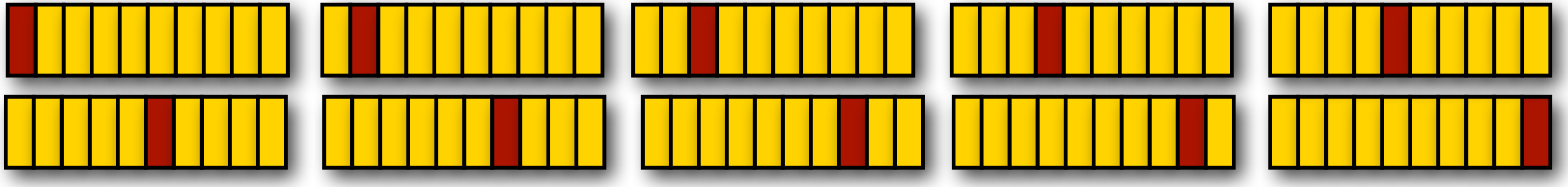
Evaluation setup:

- Split data into training (+development) and separate test sets.



Better setup: *n*-fold cross validation:

- Split data into *n* sets of equal size
- Run *n* experiments, using set *i*=1…*n* to test and remainder to train. This gives average, maximal and minimal accuracies



When comparing two taggers:

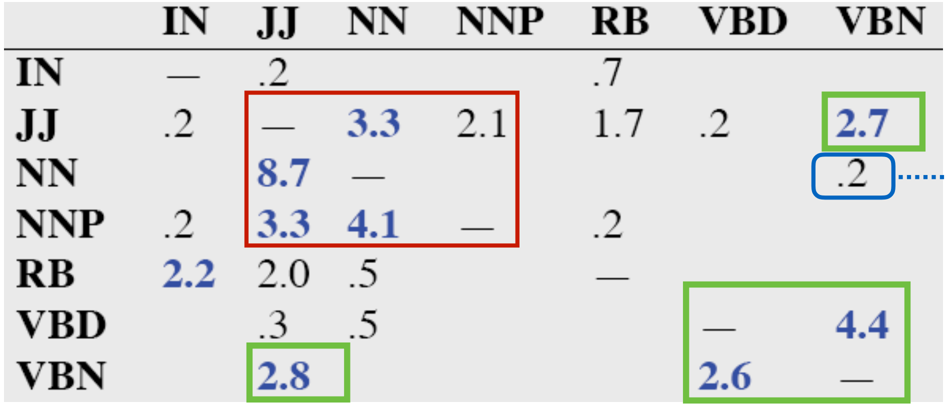
- Use the same test and training data with the same tagset

Qualitative evaluation

Generate a **confusion matrix** (for development data): How often was tag i mistagged as tag j:

Predicted tag

Correct tag



	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

Percent of overall tagging error

See what errors are causing problems:

- Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
- Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

Building a POS tagger

Statistical POS tagging

	she$_1$	promised$_2$	to$_3$	back$_4$	the$_5$	bill$_6$
w =	w_1	w_2	w_3	w_4	w_5	w_6
t =	t_1	t_2	t_3	t_4	t_5	t_6
	PRP$_1$	VBD$_2$	TO$_3$	VB$_4$	DT$_5$	NN$_6$

What is the most likely sequence of tags **t** for the given sequence of words **w** ?

Statistical POS tagging

What is the most likely sequence of tags **t** for the given sequence of words **w** ?

$$
\begin{aligned}
\operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t}|\mathbf{w}) &= \operatorname*{argmax}_{\mathbf{t}} \frac{P(\mathbf{t},\mathbf{w})}{P(\mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t},\mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{t})P(\mathbf{w}|\mathbf{t})
\end{aligned}
$$

$P(\mathbf{t})$: Generating $\mathbf{t}=t_1\ldots t_n$

We make the same Markov assumption as in language modeling:

$$
\begin{aligned}
P(t_1 t_2 \ldots t_{n-1} t_n) &= P(t_1)P(t_2|t_1)\ldots.P(t_n|t_1\ldots t_{n-1}) \\
&= \prod_{i=1}^{n} P(t_i|\ t_1\ldots t_{i\text{-}1}\ ) \\
:=_{def} &\quad \prod_{i=1}^{n} P(t_i|\ t_{i\text{-}1}\ldots t_{i\text{-}1+\mathrm{n}})
\end{aligned}
$$

We define an n-gram model over POS tags

P(**w**|**t**): Generating **w**=w₁...wₙ

We assume that words are independent of each other, and depend only on their own POS-tag:

$$\begin{aligned} P(w_1 w_2 ... w_{n-1} w_n | t_1 t_2 ... t_{n-1} t_n) &= \prod_i P(w_i | w_1 .. w_{i-1}, t_1 ... t_i ... t_n) \\ :=def &\quad \prod_i P(w_i | t_i) \end{aligned}$$

Hidden Markov Models

HMM models are **generative models** of $P(\mathbf{w},\mathbf{t})$

$P(\mathbf{w},\mathbf{t})$ describes a stochastic process that “generates” the data.

HMMs decompose $P(\mathbf{t}, \mathbf{w})$ as $P(\mathbf{t})P(\mathbf{w} \mid \mathbf{t})$

HMMs make **two independence assumptions**:

a) approximate $P(\mathbf{t})$ with an N-gram model

b) assume that each word depends only on its POS tag

An example HMM

Transition Matrix A

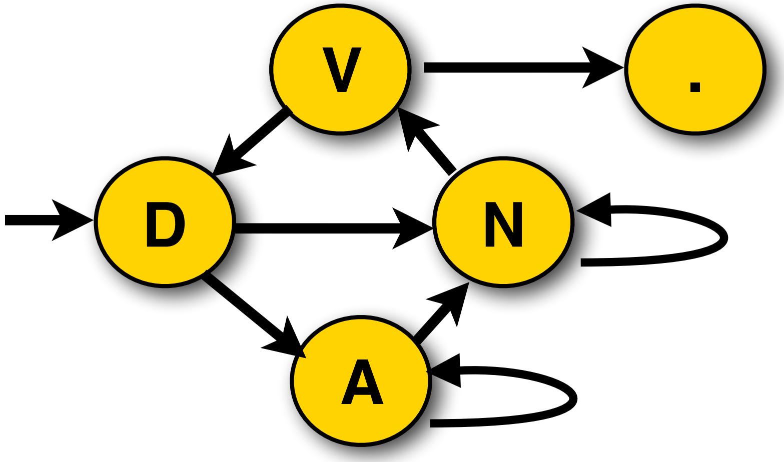
	D	N	V	A	.
D		0.8		0.2	
N		0.7	0.3		
V	0.6				0.4
A		0.8		0.2	
.					

Emission Matrix B

	the	*man*	*ball*	*throws*	*sees*	*red*	*blue*	.
D	1							
N		0.7	0.3					
V				0.6	0.4			
A						0.8	0.2	
.								1

Initial state vector π

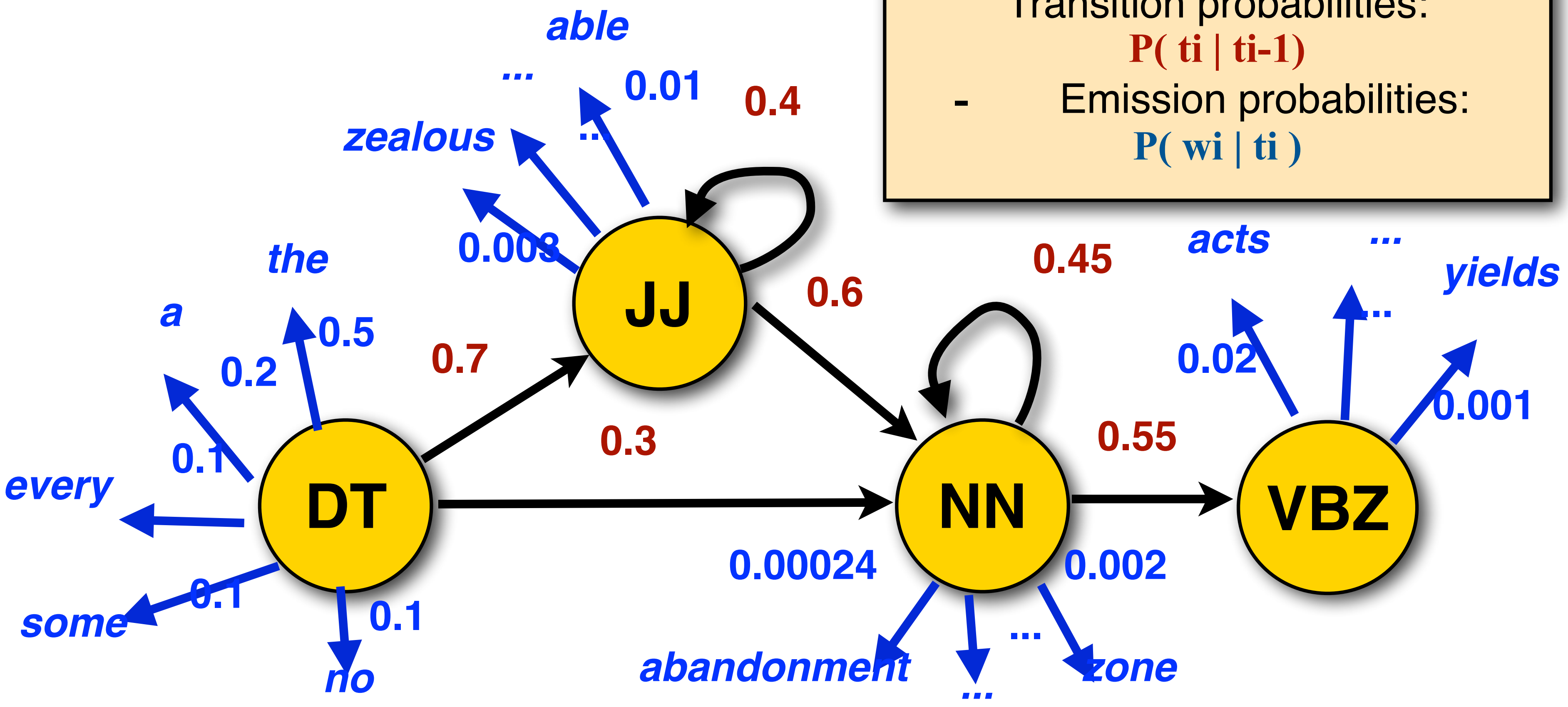
	D	N	V	A	.
π	1				



HMMs as probabilistic automata

An HMM defines
Transition probabilities:
P(ti | ti-1)

- Emission probabilities:
P(wi | ti)



Using HMMs for tagging

- The input to an HMM tagger is a sequence of words, **w**. The output is the most likely sequence of tags, **t**, for **w**.

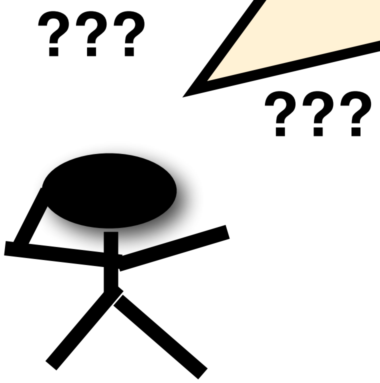
- For the underlying HMM model, **w** is a sequence of output symbols, and **t** is the most likely sequence of states (in the Markov chain) that generated w.

$$\begin{aligned}
\operatorname*{argmax}_{\mathbf{t}} P(\underbrace{\mathbf{t}}_{Output_{tagger}} \mid \underbrace{\mathbf{w}}_{Input_{tagger}}) &= \operatorname*{argmax}_{\mathbf{t}} \frac{P(\mathbf{w},\mathbf{t})}{P(\mathbf{w})} \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\mathbf{w},\mathbf{t}) \\
&= \operatorname*{argmax}_{\mathbf{t}} P(\underbrace{\mathbf{w}}_{Output_{HMM}} \mid \underbrace{\mathbf{t}}_{States_{HMM}}) P(\underbrace{\mathbf{t}}_{States_{HMM}})
\end{aligned}$$

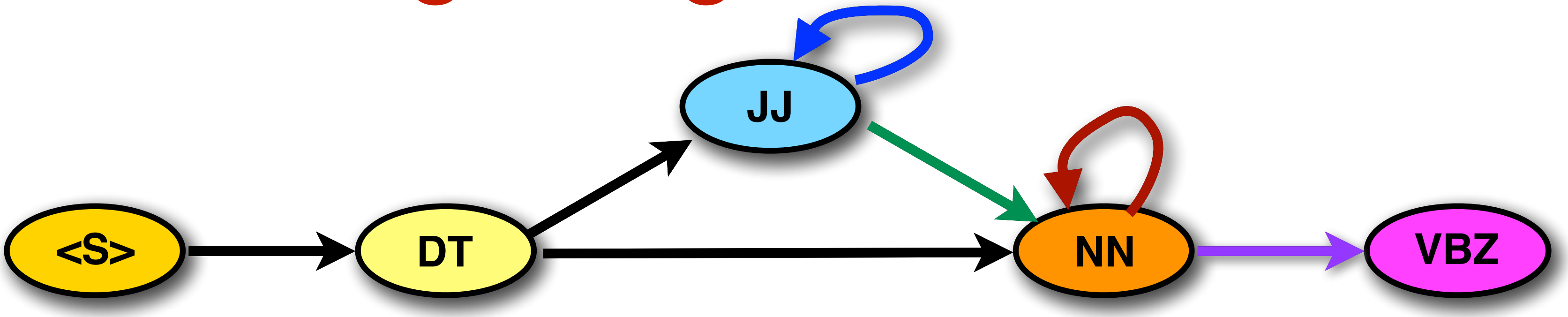
!

How would the automaton for a trigram HMM with transition probabilities $P(t_i \mid t_{i-2}t_{i-1})$ look like?

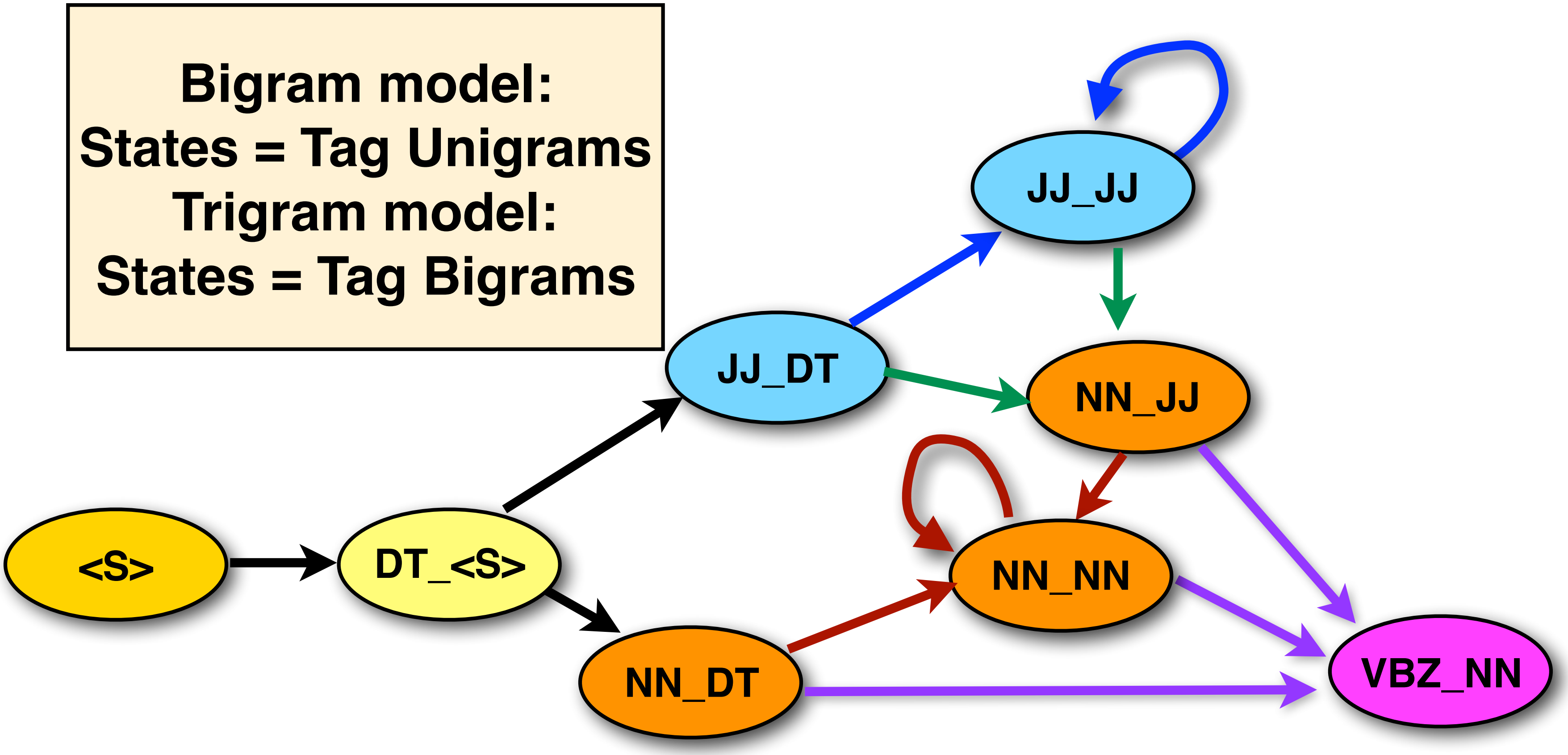
What about unigrams or n-grams?



Encoding a trigram model as FSA



Bigram model:
States = Tag Unigrams
Trigram model:
States = Tag Bigrams



HMM definition

A HMM $\lambda = (A, B, \pi)$ consists of

- a set of N **states** $Q = \{q_1, ...q_N\}$
  with $Q_0 \subseteq Q$ a set of **initial states**
  and $Q_F \subseteq Q$ a set of **final (accepting) states**

- an **output vocabulary** of M items $V = \{v_1, ...v_m\}$

- an $N \times N$ **state transition probability matrix** A
  with a_{ij} the probability of moving from q_i to q_j.
  $(\sum_{j=1}^{N} a_{ij} = 1\ \forall i; \quad 0 \leq a_{ij} \leq 1\ \forall i, j)$

- an $N \times M$ **symbol emission probability matrix** B
  with b_{ij} the probability of emitting symbol v_j in state q_i
  $(\sum_{j=1}^{N} b_{ij} = 1\ \forall i; \quad 0 \leq b_{ij} \leq 1\ \forall i, j)$

- an **initial state distribution vector** $\pi = \langle \pi_1, ..., \pi_N \rangle$
  with π_i the probability of being in state q_i at time $t = 1$.
  $(\sum_{i=1}^{N} \pi_i = 1 \quad 0 \leq \pi_i \leq 1\ \forall i)$

Learning an HMM

Where do we get the transition probabilities $P(t_j | t_i)$ (matrix A) and the emission probabilities $P(w_j | t_i)$ (matrix B) from?

Case 1: We have a POS-tagged corpus.

- This is learning from labeled data, aka “supervised learning”

```
Pierre_NNP Vinken_NNP ,_, 61_CD years_NNS
old_JJ ,_, will_MD join_VB the_DT board_NN
as_IN a_DT nonexecutive_JJ director_NN Nov._NNP
29_CD ._.
```

Case 2: We have a raw (untagged) corpus and a tagset.

- This is learning from unlabeled data, aka “unsupervised learning”

```
Pierre Vinken , 61 years old , will
join the board as a nonexecutive
director Nov. 29 .
```

Tagset:
NNP: proper noun
CD: numeral,
JJ: adjective,...

Learning an HMM from *labeled* data

```
Pierre_NNP Vinken_NNP ,_, 61_CD years_NNS
old_JJ ,_, will_MD join_VB the_DT board_NN
as_IN a_DT nonexecutive_JJ director_NN Nov._NNP
29_CD ._.
```

We count how often we see t_it_j and $w_j\_t_i$ etc. in the data (use relative frequency estimates):

Learning the transition probabilities:

$$P(t_j|t_i) = \frac{C(t_it_j)}{C(t_i)}$$

Learning the emission probabilities:

$$P(w_j|t_i) = \frac{C(w_j\_t_i)}{C(t_i)}$$

We might use some smoothing, but this is pretty trivial…

Outlook: Dynamic Programming for HMMs

The three basic problems for HMMs

We observe an **output sequence** $w = w_1 \ldots w_N$:

$\boldsymbol{w}$=*"she promised to back the bill"*

Problem I (Likelihood): find $P(w \mid \lambda)$

Given an HMM $\lambda = (A, B, \pi)$, compute the likelihood of the observed output, $P(\boldsymbol{w} \mid \lambda)$

Problem II (Decoding): find $Q = q_1 .. q_T$

Given an HMM $\lambda = (A, B, \pi)$, what is the most likely sequence of states $Q = q_1 .. q_N \approx t_1 \ldots t_N$ to generate $\boldsymbol{w}$?

Problem III (Estimation): find $argmax_\lambda\, P(w \mid \lambda)$

Find the parameters A, B, π which maximize $P(\boldsymbol{w} \mid \lambda)$

How can we solve these problems?

I. Likelihood of the input:

Compute $P(\boldsymbol{w} \mid \lambda)$ for the input $\boldsymbol{w}$ and HMM λ

II. Decoding (= tagging) the input:

Find the best tags $\boldsymbol{t}^{*}{=}argmax_{\boldsymbol{t}}\, P(\boldsymbol{t} \mid \boldsymbol{w}, \lambda)$ for the input $\boldsymbol{w}$ and HMM λ

III. Estimation (= learning the model):

Find the best model parameters $\lambda^{*}{=}argmax_{\lambda}\, P(\boldsymbol{t}, \boldsymbol{w} \mid \lambda)$
for the training data $\boldsymbol{w}$

These look like hard problems: With T tags, every input string $\boldsymbol{w}_{1...n}$ has T^{n} possible tag sequences

Can we find efficient (polynomial-time) algorithms?

Dynamic programming

We will use a general technique called dynamic programming to solve these problems.

- We will recursively decompose each of these problems into smaller subproblems that can be solved efficiently
- There is only a polynomial number of subproblems.
- We will store the solution of each subproblem in a common data structure
- Processing this data structure takes polynomial time

Solution: Dynamic programming

I. Likelihood of the input:

Compute $P(\boldsymbol{w}| \lambda)$ for the input $\boldsymbol{w}$ and HMM λ

⇒ **Forward algorithm**

II. Decoding (=tagging) the input:

Find best tags $\boldsymbol{t}^{*}=argmax_{\boldsymbol{t}}\, P(\boldsymbol{t} \mid \boldsymbol{w},\lambda)$ for the input $\boldsymbol{w}$ and HMM λ

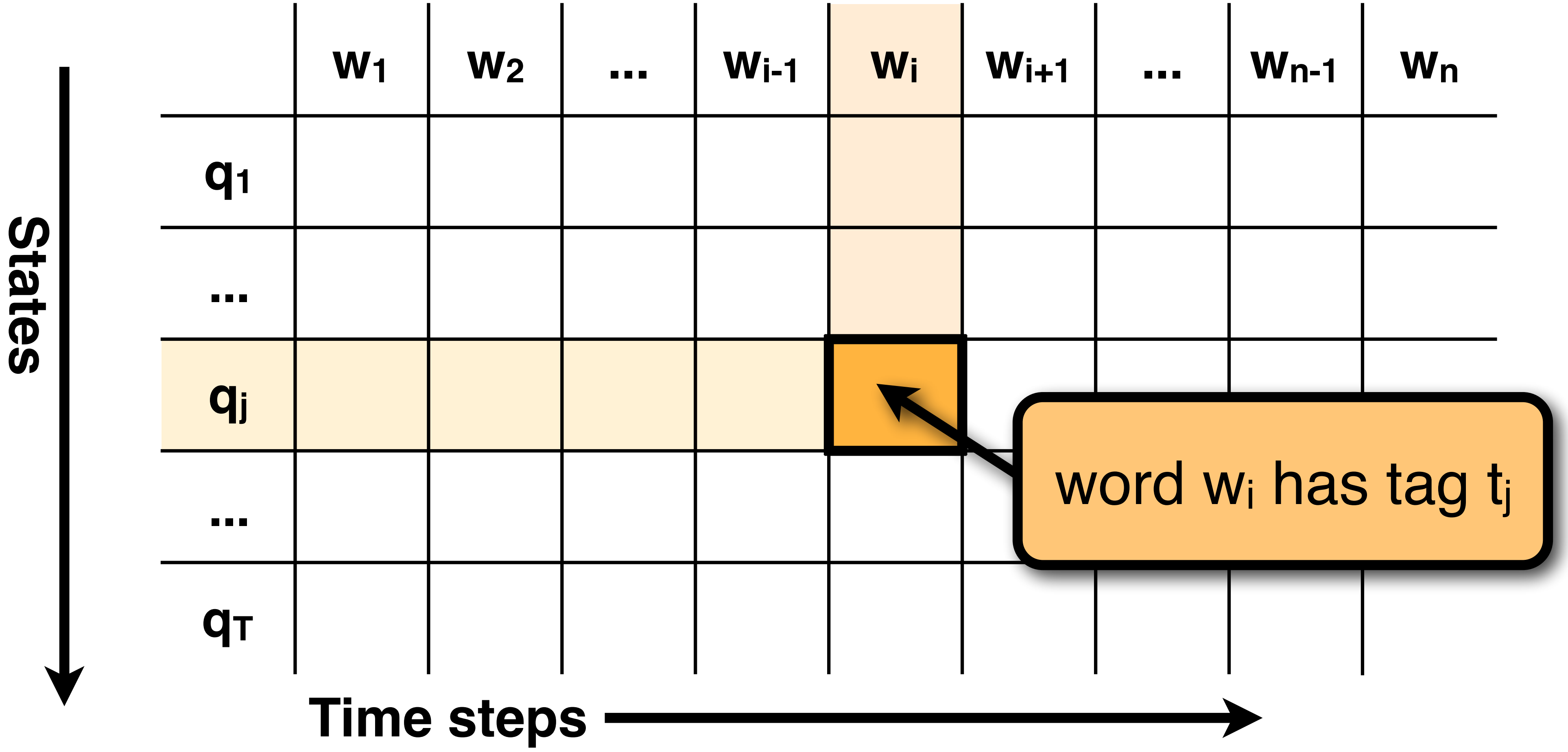
⇒ **Viterbi algorithm**

III. Estimation (=learning the model):

Find best model parameters $\lambda^{*}=argmax_{\lambda}\, P(\boldsymbol{t}, \boldsymbol{w} \mid \lambda)$ for training data $\boldsymbol{w}$

⇒ **Forward-Backward algorithm**

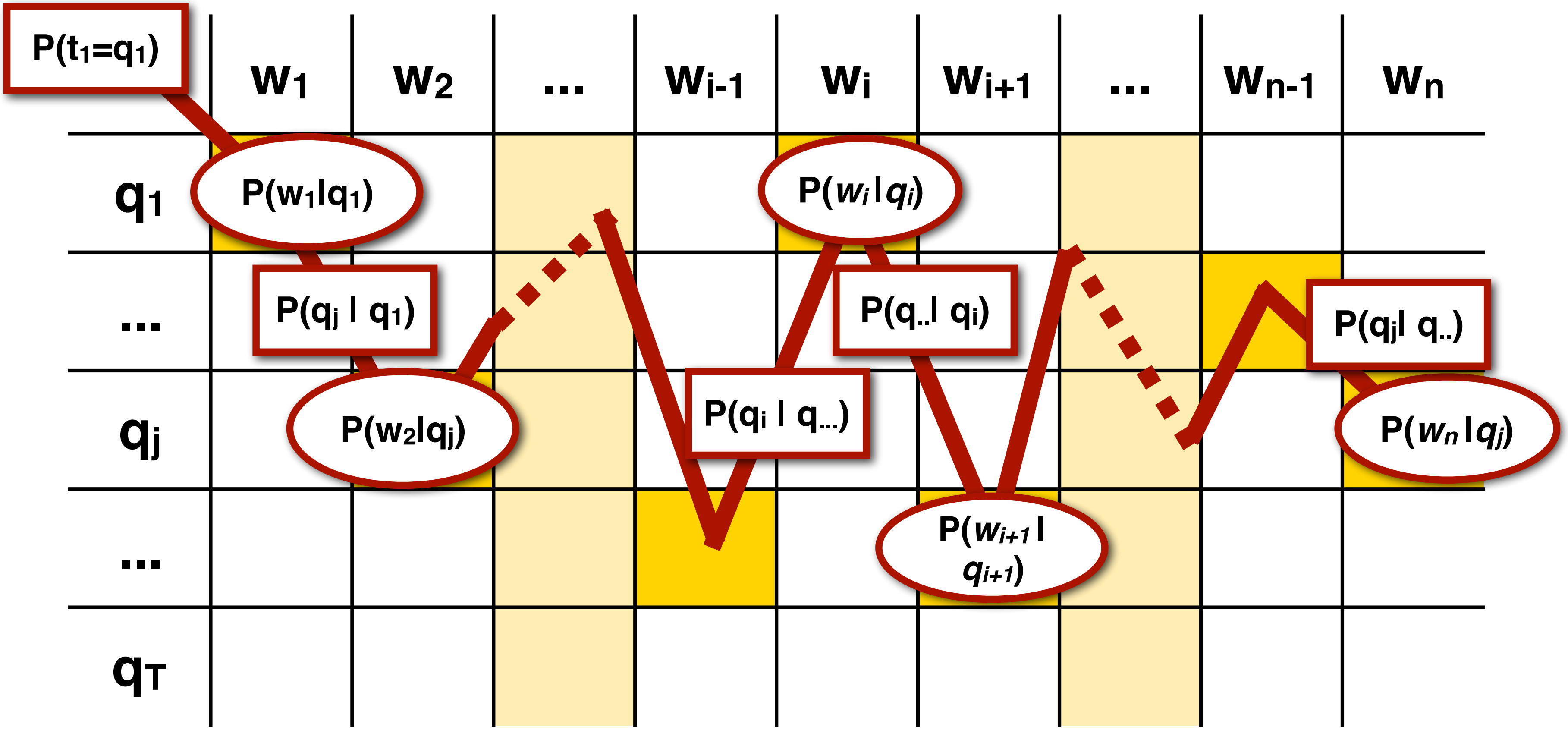
Bookkeeping: the trellis



We use a *n*×*T* table ("**trellis**") to keep track of the HMM.

At every one of the *n* time steps (=words), the HMM can be in one of T states (=tags)

Computing *P*(t,w) for one tag sequence



- One path through the trellis = one tag sequence
- We just multiply the probabilities as before

$$P(\mathbf{t},\mathbf{w}) \quad = \quad P(t_1)P(w_1|t_1)\prod_{i=2}^{T} P(t_i|t_{i-1})P(w_i|t_i)$$

Appendix: English parts of speech

Nouns

Nouns describe entities and concepts:

Common nouns: dog, bandwidth, dog, fire, snow, information

- Count nouns have a plural (dogs) and need an article in the singular (the dog barks)
- Mass nouns don’t have a plural (*snows) and don’t need an article in the singular (snow is cold, metal is expensive). But some mass nouns can also be used as count nouns: Gold and silver are metals.

Proper nouns (Names): Mary, Smith, Illinois, USA, France, IBM

Penn Treebank tags:

NN: singular or mass

NNS: plural

NNP: singular proper noun

NNPS: plural proper noun

(Full) verbs

Verbs describe activities, processes, events:

eat, write, sleep, ….

Verbs have different morphological forms:
infinitive (to eat), present tense (I eat), 3rd pers sg. present tense (he eats), past tense (ate), present participle (eating), past participle (eaten)

Penn Treebank tags:

VB: infinitive (base) form
VBD: past tense
VBG: present participle
VBD: past tense
VBN: past participle
VBP: non-3rd person present tense
VBZ: 3rd person singular present tense

Adjectives

Adjectives describe properties of entities:

blue, hot, old, smelly,…

Adjectives have an...

… attributive use (modifiying a noun):

the blue book

… and a predicative use (e.g. as argument of be):

The book is blue.

Many gradable adjectives also have a…

...comparative form: greater, hotter, better, worse

...superlative form: greatest, hottest, best, worst

Penn Treebank tags:

JJ: adjective JJR: comparative JJS: superlative

Adverbs

Adverbs describe properties of events/states.

- Manner adverbs: slowly (slower, slowest) fast, hesitantly,…
- Degree adverbs: extremely, very, highly….
- Directional and locative adverbs: here, downstairs, left
- Temporal adverbs: yesterday, Monday,…

Adverbs modify verbs, sentences, adjectives or other adverbs:

Apparently, the very ill man walks extremely slowly

NB: certain temporal and locative adverbs (yesterday, here)
can also be classified as nouns

Penn Treebank tags:

RB: adverb RBR: comparative adverb RBS: superlative adverb

Auxiliary and modal verbs

Copula: be with a predicate

She is a student. I am hungry. She was five years old.

Modal verbs: can, may, must, might, shall,…

She can swim. You must come

Auxiliary verbs:

- Be, have, will when used to form complex tenses:

He was being followed. She has seen him. We will have been gone.

- Do in questions, negation:

Don't go. Did you see him?

Penn Treebank tags:

MD: modal verbs

Prepositions

Prepositions occur before noun phrases to form a prepositional phrase (PP):

on/in/under/near/towards the wall,
with(out) milk,
by the author,
despite your protest

PPs can modify nouns, verbs or sentences:

I drink [coffee [with milk]]
I [drink coffee [with my friends]]

Penn Treebank tags:

IN: preposition
TO: ‘to’ (infinitival ‘to eat’ and preposition ‘to you’)

Conjunctions

Coordinating conjunctions conjoin two elements:

X and/or/but X

[[John]NP and [Mary]NP] NP,

[[Snow is cold]S but [fire is hot]S]S.

Subordinating conjunctions introduce a subordinate (embedded) clause:

[He thinks that [snow is cold]S]S

[She wonders whether [it is cold outside]S]S

Penn Treebank tags:

CC: coordinating

IN: subordinating (same as preposition)

Particles

Particles resemble prepositions (but are not followed by a noun phrase) and appear with verbs:

come on
he brushed himself off
turning the paper over
turning the paper down

Phrasal verb: a verb + particle combination that has a different meaning from the verb itself

Penn Treebank tags:

RP: particle

Pronouns

Many pronouns function like noun phrases, and refer to some other entity:

- Personal pronouns: I, you, he, she, it, we, they
- Possessive pronouns: mine, yours, hers, ours
- Demonstrative pronouns: this, that,
- Reflexive pronouns: myself, himself, ourselves
- Wh-pronouns (question words):
  what, who, whom, how, why, whoever, which

Relative pronouns introduce relative clauses

the book that [he wrote]

Penn Treebank tags:

PRP: personal pronoun   PRP$ possessive   WP: wh-pronoun

Determiners

Determiners precede noun phrases:

the/that/a/every book

- Articles: the, an, a
- Demonstratives: this, these, that
- Quantifiers: some, every, few,…

Penn Treebank tags:

DT: determiner